Optical Character Recognition For Hand Written Digits and Computer Fonts

Himanshu Dahiya
Department Of CSE IIT Ropar

Shreya Dubey
Department Of CSE IIT Ropar

Abstract

Optical character recognition is the electronic conversion of images of handwritten or printed text, i.e. computer font, into machine-encoded text. These images could be from a scanned document, a photo of a document, a scene-photo, for example the text on signs and billboards in a landscape photo.

Optical Character Recognition(OCR) is one of the most useful applications of machine learning. The aim of this project was to learn models which could classify hand written digits and various computer fonts. Various machine learning techniques have shown good results in this field, this project includes three such models. Using these models we were able to achieve accuracy of upto 92.06% and 83.8% for MNIST and notMNIST datasets respectively.

Introduction

The problem that we attempt to solve is to classify the hand-written digits(0 to 9) and computer fonts. That is our model is trying to learn the classification of letters and digits correctly.

The problem that our project aims to solve is the accurate classification of handwritten digits and computer fonts given a new data. There are wide ranging applications of optical character recognition such as it is widely used as a form of information entry from printed paper data records, whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, or any other documentation.

By digitising printed texts we can ensure that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining.

OCR also has its uses in pattern recognition, artificial intelligence and computer vision. OCR can be used for creating reading devices for the blind, automatic number-plate recognition for reading the registration number on the vehicles by using the existing road-rule enforcement cameras, or cameras specifically designed for the task. So, in order to implement this project we decided on three models and observed as to which model was able to classify the letters or letters more accurately.

Related Work

On present day, there are various projects going on in the field of OCR, and various research papers published. We got our idea from a project paper of Stanford University, in which they tried various techniques of Machine Learning to classify given MNIST and notMNIST dataset. They took equal number of instances for each class for training and for testing as well, equal instances of each class were taken. They trained their dataset using different SVM models and L1-regularised logistic regression. The dataset they used consisted of MNIST dataset of hand-written digits and notMNIST dataset of computer fonts, in which each data point was a 28X28 grayscale image. Also, they used PCA(Principal Component Analysis) for dimensionality reduction because there were some redundant features, so removing them leads to increase in accuracy.

DataSets Used

We used two types of datasets, MNIST and notMNIST. The training data contains equal number of examples for each digit and characters respectively in both the datasets.This was done to ensure that the training dataset is not biased to obtain the best results on the test dataset. Similarly, the test dataset consists of equal number of points from each class so that the error that we obtain is generalised error and is not biased towards any class.

The MNIST dataset consists of 60,000 examples of handwritten digits for training and 10,000 examples for testing. This dataset consists of digits only from 0-9, and each datapoint is a 28X28 size image, which in turn leads to 784 features. Out of these, we randomly picked 15,000 training examples, 1,500 of each digit, and 3,000 of test examples, 300 of each digit.

The notMNIST dataset has 18,724 images, each of which were 28X28 size, leading to same kind of dataset as MNIST. We divided this dataset into two parts, training and testing, with 80:20 ratio, which leads to 14,979 training examples and 3,745 testing examples. This dataset had computer fonts form A-J, which makes 10 classes in it also as MNIST dataset.

Methodology

OCR is a very common problem in today's world and there are many solutions available to it, like "Artificial Neural Networks", "Support Vector Machines", "Logistic Regression" and many variations and models of these. Among all the available solutions, we tried implementing Support Vector Machines with 4th degree polynomial kernel, L2 Logistic regression and kNN (k- Nearest Neighbour) for our project.

kNN(k - Nearest Neighbours)

In case of k nearest neighbour algorithm, for each data point, the model assigns the most frequently occurring label of the k nearest training data points to the given test data point where the distance between two points is taken to be the euclidean distance between them.

There is no training in this algorithm, it only takes euclidean distances of the new data point from all the training data points in the original space and classifies the new unclassified data point with respect to the nearest "k" data points and assigns the majority label among them.

K nearest neighbour is a non-linear classifier. The accuracy was relatively high for smaller values of k where k=3 gave the best results.

Logistic Regression with L2 regularisation

In case of L2 Logistic Regression, using Newton-Raphson for convergence, we implemented one versus one classification. This was done because we have 10 classes, for both the MNIST as well as the notMNIST datasets. One versus one was implemented by taking all possible combinations of pair of classes in each iteration and learning the model considering only those two classes, thus now instead of multi-class model, we have a binary model which we can easily trained using the logistic regression with L2 regularisation, which suits binary classification. We would like to maximise the likelihood which is

$$\sum log(p(y^i)\,|\,x^i;\theta) - (\theta^T\theta)^2)$$

with respect to θ where $p(y^i\,|\,x^i;\theta)$ is a sigmoid function:

$$(1/(1+exp(-\theta^T x)))^{y^i} * (1 - 1/(1+exp(-\theta^T x)))^{1-y^i}$$

Logistic regression fits a linear classifier i.e. the decision boundaries are hyperplanes in the original space.

For convergence, we chose Newton-Raphson Method over gradient descent, because it converges faster. In this method, we minimise the loss function,

$$J(\theta) = -\sum_{i=1}^{N} (y^i \log(f(x^i)) + (1-y^i)\log(1-f(x^i)))$$

by finding zero of $J'(\theta)$. Thus, we update θ as,

$$\theta := \theta - \frac{J'(\theta)}{J''(\theta)}$$

We define "Hessian" as $H = J''(\theta)$, so the parameter update equation used becomes,

$$\theta^{t+1} = \theta^t - H^{-1}\nabla J(\theta)$$

Support Vector Machines with 4th Degree Polynomial Kernel

The dataset we are using is 784 dimension data, which might not be linearly separable and therefore we project it to a high dimensional data, where the data might become linearly separable with respect to each pair of classes, and then we fit an optimal separating hyperplane for each pair of class.

In this case, we tried the fourth degree kernel i.e. the model projects the data to a higher dimensional space and tries to fit in a linear decision boundary. We implemented one versus one classification because SVM models binary classification. So, in each iteration we took only those data points which belong to the two classes taken, and the SVM model was fit on this data. The process was repeated for all the combinations of two classes and the labels assigned to the data point was most frequently assigned label in all the possible combinations.

Noise may be present in our dataset, so we are trying to fit "Soft Margin Hyperplane", where C is the penalty factor for the soft error.

In the SVM model, we try to minimise

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\epsilon^i$$

with respect to w and w_0 subject to the constraints:

$$y_i(w^T x^i + w_0) \geq 1 - \epsilon^i$$
$$\epsilon^i \geq 0$$

which is the primal, which on solving results in the dual problem where the objective is to maximise

$$L_d = max_{\alpha^i} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha^i\alpha^j y^i y^j K(i,j) + \sum_{i=1}^{n}\alpha^i$$

with respect to α^i subject to the constraints:

$$\sum_{i=1}^{n}\alpha^i y^i = 0$$
$$0 \leq \alpha^i \leq C\ \forall i$$

Here,K(i , j) is the (i , j) entry of the kernel matrix. We are taking a 4th degree polynomial kernel, where,

$$K(x^i, x^j) = (\gamma {x^i}^T x^j)^4$$

Dual problem was solved using quadratic programming where the box constraint C was set to 2.

Results and Discussion

For the various models, we obtained the following results.

	MNIST	notMNIST
kNN(k=3)	74.3%	69.4%
L2 Logistic Regression(λ = 5)	92.06%	81.23%
SVM with 4th Deg Poly Kernel(C = 2)	80.3%	83.8141%

We took 15000 training examples and 3000 testing examples for both the datasets, MNIST and notMNIST. The above table shows the testing accuracy of various models using each dataset.

kNN(k - Nearest Neighbours)

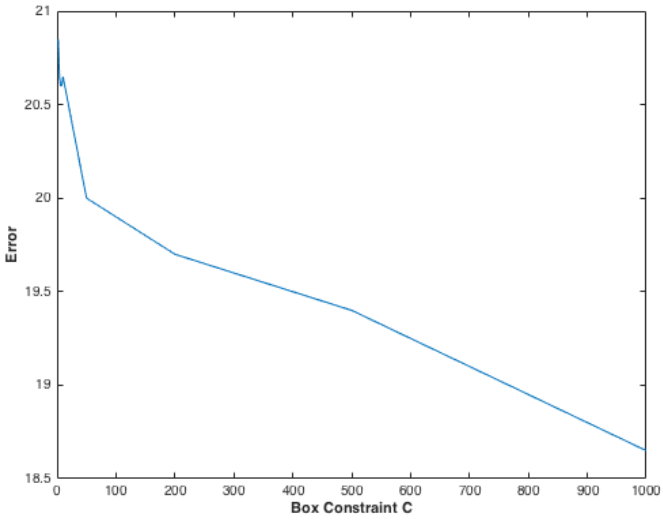
In kNN, we tried to find the optimal value of k to improve our model. Thus, we experimented with smaller values of k i.e. 3 and 5 and for a larger value of k, i.e. 40 and the accuracy obtained for MNIST dataset was 74.3% for k=3, 73.6% for k=5 and 53.8% for k=40. Thus we observed that for smaller values of k, accuracy almost remained the same , but for vary large values of k, the accuracy decreased by a significant amount. Thus, k=3 which gave an accuracy of 74.3% was chosen as the optimal k for the kNN model.

Logistic Regression with L2 regularisation

In case of logistic regression with L2 regularisation, the hyper-parameter λ had to be chosen which best fits the input data. The hyper parameter λ is the regularisation parameter, thus increasing it will lead to less overfitting but it may underfit the data if it is too large, thus we experimented with different values of λ. For smaller values of λ like λ =0.00005, the test accuracy comes out to be 81% for MNIST dataset, for very large values of λ like λ= 50 the accuracy comes out to be 91.66% As we saw, for λ = 5, the test accuracy was 92.06%. Thus, the optimal value of λ was taken to be 5.

Support Vector Machines with 4th Degree Polynomial Kernel

For the SVM model, we chose a fourth degree kernel in order to project the input space to a higher dimensional space and observe if the model is able to fit an optimal separating hyperplane on the data projected onto a higher dimensional space. For SVM, box constraint C = 2, γ = 1.414 which was used in kernel function, we saw that for MNIST dataset, accuracy was 80.3% and for notMNIST dataset, the accuracy was 83.81%. We also observed that notMNIST dataset was not standardised and we were not able to solve the dual problem as quadratic programming was not able to converge because the problem was not convex. Thus, to find the optimal α, the dataset was standardised and then used.



Error VS Box Constraint C on MNIST Dataset with 10,000 training and 2,000 testing examples.

We also tried different values of box constraint with MNIST dataset by taking 10,000 training and 2,000 testing examples because large dataset was taking a lot of time, we observed that on increasing the value of the box constraint, the error obtained was decreasing, as shown in the above graph. This can be attributed to the fact that box constraint is the penalty factor which stresses on minimising the soft error.

The primal problem that we are trying to solve is:

$$\frac{1}{2}||w||^2 + C\sum_{i}\epsilon^i$$

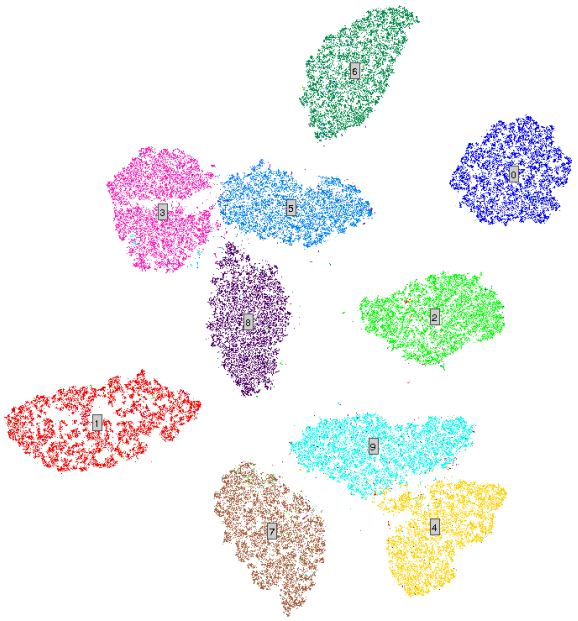
$$=> C(\frac{1}{2C}||w||^2 + \sum_i \epsilon^i)$$

If, C is very large, $\frac{1}{2C} = \lambda$ (regularisation parameter) becomes very small. Which means, $||w||^2$ can be large, therefore, the margin in SVM model = $\frac{2}{||w||^2}$ becomes small.

Thus, as we are increasing the box constraint we are taking into account the noise present in the data and thus accuracy increases.

Below shown is the table for box constraint and error relative to box constraint on MNIST Dataset with 1,0000 training and 2000 testing examples.

C	2	4	6	8	10	50	100	150	200	500	1000
Error	20.85	20.65	20.60	20.60	20.65	20.00	19.90	19.80	19.70	19.40	18.65



Embedding of MNIST into 2D space using a neighbour embedding visualisation technique

Comparing kNN and SVM models, we saw that SVM outperformed kNN by a huge margin. SVM gave an accuracy of 83.8% on the notMNIST dataset and 80.3% on the MNIST dataset. kNN gave an accuracy of 74.3% on MNIST dataset while an accuracy of 69.4% on notMNIST dataset. We can say this because kNN has a very unsophisticated model whereas SVM projects the data to a higher dimensional data and tries to fit the data in a linearly separable boundary assuming that the data is linearly separable in higher dimension. This indicates that notMNIST dataset is not linearly separable as SVM gets the best accuracy by projecting the dataset to higher dimensional space.

In the 2D embedding of the MNIST dataset shown above it is very clear that the MNIST dataset is non-linear thus it turns out that SVM models the data well and achieves a decent accuracy on the same. It performs much better than kNN, by about 6% on MNIST dataset and about 14% on notMNIST dataset.

We also observed that SVM performed well on both the datasets, MNIST and notMNIST but SVM was slightly better than L2 logistic regression on notMNIST dataset. On the other hand L2 logistic regression was better than SVM model in case of MNIST dataset. The reason for this might be the fact that we are implementing one versus one classification, thus in each iteration when we take a pair of classes we are considering only those data points which belong to those two classes which we can see are linearly separable. Thus, while taking pairs of classes since the input space becomes linearly separable, logistic regression might be able to fit the data decently.

Future Work

In this project, the dataset is of very high dimension, i.e. 784 dimension, therefore we can reduce the dimension of the dataset using Principal Component Analysis to reduce the redundancy in dataset. Also, we can implement various machine learning techniques in like Artificial neural network and Convolution Neural Network which have proved to be very efficient in OCR.

This project can be further extended to License Plate Recognition of Vehicles where we are given an image of vehicle and detect the License Number of the vehicle. That will also include various aspects of Image Processing as well. Also, we can can use this project in computer vision and various other fields of Image Processing.

Summary

We performed Logistic regression with L2 regularisation, k-Nearest Neighbours and SVM with 4th Degree Polynomial Kernel on both the MNIST and notMNIST datasets to observe which model best classifies letters or digits in the given data. This project can be used to read scanned documents, License Plate Detection of Vehicles, or convert any text to speech which can be used for the blind. We observed that SVM with 4th degree kernel outperforms other models indicating that the notMNIST dataset is not linearly separable and in the case of MNIST SVM was the second best model.

References

MNIST digit database of handwritten digits. Yann LeCun, Corrina Cortes. *http://yann.lecun.com/exdb/mnist/*

the notMNIST data set. Yaroslav Bulatov. *http://yaroslavvb.blogspot.in/2011/09/notmnist-dataset.html*

http://cs229.stanford.edu/proj2011/Margulis-OpticalCharacterRecognition.pdf

https://in.mathworks.com/help/stats/classificationecoc-class.html

https://in.mathworks.com/help/stats/fitcecoc.html?searchHighlight=fitcecoc&s_tid=doc_srchtitle

https://www.coursera.org/learn/ml-classification/lecture/DBTNt/l2-regularized-logistic-regression

https://cs.stackexchange.com/questions/59597/about-the-mnist-data-set