Optical Character Recognition

Approach using k-Nearest Neighbors and Support Vector Machines

Alla Marchenko, Olav Markussen and Geir Kulia

April 26, 2017

1 Introduction

This report is presenting two approaches to Optical Character Recognition (OCR). The approaches applied are k-Nearest Neighbor Classifier and Support Vector Machines. A general overview of the implementation is shown in Figure 1. The data is loaded from the Char74k-Lite dataset and diveded, at random, into two databases. 80 % is selected as the training set, and 20 % is the test set. The data is then preprocessed, as described in section 2. A model is trained on the training set, before the model is passed to the classifier. The classifier is described in section 3. The system output is the classifier error, which is given by

$$E = \frac{N_{\text{fail}}}{N_{\text{test}}} \tag{1}$$

where N_{fail} is the number of wrong classifications in the test set and N_{test} is the total number of samples in the test set.

The Python program mainly used two packages: Scikit-image and Scikit-learn. Scikit-image is a opensource image processing library, that was manly used for preprocessing. Scikit-learn is a library that can be used for classification, regression, clustering, dimensional reduction, model selection, and also preprocessing. It was heavely applied during this project. A complete list of all libraries used are shown in Appendix A.

Test this. The software uses UNIX convention for file paths, so this might not work on windows. To start the python program: type

```
$ python3 main.py
```

into your terminal. To run the Matlab program, type

```
$ matlab -nodesktop -nosplash
>> run main
```

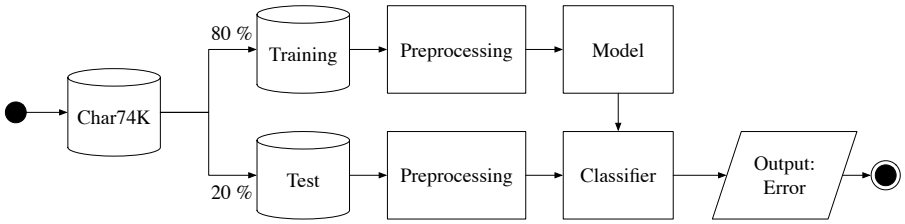

Figure 1: Model of OCR system.

(a) Raw image. (b) Denoised image. (c) Binary image. (d) Histogram of Oriented Gradients.

Figure 2: Example of random selected image before and after preprocessing. The colors has been altered using a color map to visually enhance variance.

2 Feature Engineering

In this section, the preprocessings of the images are shown and discussed. The preprocessing is preformed to enable easier classification for the classifier. Several preprocessing tools were explored to make the character easier to recognize for the classifier. The first preprocessing tool applied was total-variation denoising on n-dimensional images [1]. This technique reduces the total variation of an image. This is useful to avoid classifying noise, as the picture will now contain fewer components with high spacial frequency. An example of this denoising technique is shown in the image Figure 2b.

To increase contrast, we used Otsu's method, which is a way to perform image thresholding based on clustering [6] automatically. After thresholding, the image was morphologically closed. This is a standard technique in image processing to remove small wholes in binary images. The result is shown in Figure 2c. All though the result was visually pleasing; it did not decrease the error rate. Therefore, this method was discarded from further analysis.

The last preprocessing tool applied before dimention reduction was Histogram of Oriented Gradients. It is a feature descriptor uses local object appearences and shapes and describe them as a distribution of intensity gradients. It can be interpreted as the spacial derivative of the intencity of the image. The purpose of this step is to format the image for easier detection later as each element now carry more information than the pixels in the original.

One feature engineering method used and not covered by this section is

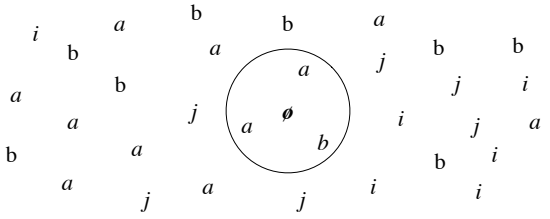

Figure 3: An example of k-nearest neighbors algorithm. An unknown input character ø is decided by comparing it with the $k = 3$ nearest neighbors. The three nearest neighbors to ø are a, a, and b, so in this example ø$= a$.

Principal Component Analysis. Because its application is heavily based on the k-nearest neighbor algorithm, it will be covered in subsubsection 3.1.1.

3 Classifier

In this section, we will discuss two approaches for classifying characters: the k-nearest neighbors' algorithm and linear support vector classifier. We will also discuss task-tailored feature-engineering used to enhance the result, which is not discussed in the previous chapter.

After examining the data set Char74K-lite, the author's initial thought was to use Conventional neural networks (CNN). CNNs have been used to recognize far more complicated objects than characters [4]; hence, it should be a trivial task to apply CNNs to recognize letters [8]. However, due to the limited time for training, lack of heavyweight graphical processing units, and that it is discouraged by the assignment, we choose not to use Conventional neural networks.

The authors had a limited knowledge of Principal Component Analysis and Support Vector Machines and thought this would be an excellent opportunity to gain more theoretical knowledge as well as hands-on experience with these methods. We, therefore, chose to pursuit one application with Principal component analysis and k-nearest neighbors algorithm as a classifier. The other approach was to use linear support vector classifiers. Both methods used the preprocessing described in section 2.

3.1 k-nearest neighbors algorithm with Principal component analysis

The k-nearest neighbors algorithm (k-NN) is a simple non-parametric classification method [5, pp. 231-236]. A training set is embedded in a n-dimensional space and used as a prediction model. The model can then predict new data by estimating the shortest distance to the k nearest classes in the n-dimensional

space, as demonstrated in Figure 3. The unknown character is classified, apparently enough, as the median of the k-nearest neighbors classes.

It is a tangible challenge and computationally heavy to perform classification on high dimensional (i.e. large n) data with k-NN. Therefore, a dimension reduction algorithm was first applied before the k-NN classifier, as described in the next chapter.

3.1.1 Principal component analysis

A dimension reduction tool is preferable before performing a k-Nearest Neighbor Classifier, as discussed in the previous subsection. In this project, the Principal Component Analysis (PCA) procedure was used for reducing the number of correlated dimensions down to a set of the linearly uncorrelated principal component [3]. The principal Component Analysis will use too much space to derive in this report. In short, it decomposes the original n-dimensional space into m orthogonal dimensions by finding the orthogonal dimensions with the highest variance and computing the variance vector in that dimensions direction. The first dimension will have the highest variance; the second will have the second largest and so on. Numerical estimations show that the optimal condition for the k-Nearest Neighbor Classifier is when PCA is used to reduce the dimension down to $m = 60$.

The authors spent a substantial amount of time to make an implementation using t-distributed stochastic neighbor embedding [7] for easier separation of characters. However, this was not possible as the t-distributed stochastic neighbor embedding algorithm can only visualize trained data, and not transform a test set. If a t-distributed stochastic neighbor embedding transform is deactivated, then we believe that the combination of Principal Component analysis, t-distributed stochastic neighbor embedding, and k-nearest neighbors algorithm would be interesting to explore.

3.1.2 Performance of k-nearest neighbors algorithm

To find the best performance of k-NN with PCA, it is useful to implement a search with a nested loop, with a focus on the number k of neighbors and amount of dimension m the data was reduced to. The best performance without implementation was when $k = 5$ and, as mentioned above, $m = 60$. The best performance was calculated using Equation 1 and resulted in an error of

$$E_{k\text{NN}} = 24.4\ \% \tag{2}$$

which was higher than the allowed error of this project. We, therefore, continued with an approach based on Linear Support Vector Classifiers, as described in the next section.

3.2 Linear Support Vector Classifier

A Linear Support Vector Classifier (LSVC) implemented is a supervised learning model for binary classification [2]. A Multiclass LSVC was used in this example,

which means that the 26 classes are reduced to a set of binary classifications.

Each binary LSVC classification maps the training data's separate categories into points in space, and linearly divide them by an apparent gap. The test data is assigned to the same space and classified based on what side of the division line the data appears. Support Vector Classifiers can perform classify non-linear classification by using kernel tricks, i.e. transforming the data. However, kernel tricks were not applied in this project.

4 Character Detection

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

5 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

References

[1] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging*, 2004.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] J. Edward Jackson. *A User's Guide to Principal Components*. John Wiley and Sons, Inc., 2004.

Table 1: Libraries used.

Library	Link
`time`	`https://docs.python.org/3/library/time.html`
`matplotlib`	`https://matplotlib.org/`
`numpy`	`http://www.numpy.org/`
`scipy`	`https://www.scipy.org/`
`skimage`	`http://scikit-image.org/`
`sklearn`	`http://scikit-learn.org/`
`string`	`https://docs.python.org/3/library/string.html`
`os`	`https://docs.python.org/3/library/os.html`
`PIL`	`http://www.pythonware.com/products/pil/`

[4] S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, Jan 1997.

[5] Tom M. Mitchell. *Machine Learning*. Mc Graw Hill - Education, Singapore, 1997.

[6] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1979.

[7] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[8] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. pages 3304–3308, Nov 2012.

A Libraries used

A complete list of all libraries used for the python implementation can be found in Table 1.