

Machine learning and case based reasoning

TDT4173- Assignment 5

Øyvind Aukrust Ronnes, NTNU

May 4, 2018

1 Introduction

a) Two optical character recognition programs have been implemented in matlab and python respectively. The matlab program uses maximum likelihood estimation to predict characters while the python program relies on a CNN.

Additionally, the matlab program identifies all the images embedded in the two test images and store them as separate images through the function `process_test_images()`.

While the matlab program does not rely on any external libraries, the python program uses Keras and numpy for the CNN and PIL to read images.

How to run the programs

The matlab program is run through the file `main.m` and the directory of the training images must be changed here. The amount of principal components used in PCA can be set here too.

The python program is run through the only python file in the directory, `keras_network.py`.

2 Feature Engineering

b) The matlab program uses PCA to reduce the dimensions of the problem. As the MLE depends on an *expensive* multivariate gaussian pdf this greatly increases speed by essentially ignoring pixels that normally do not contain vital information. It also subtracts the mean of all images from each images to achieve a zero mean and all pixel values are automatically scaled as floats between 0 and 1. Additionally, when loading the test images, the embedded images are extracted and the white background is discarded by using a sliding window with a stride of one and picking the windows with the highest mean.

The python program only preprocess the data by using a hot one encoding.

c) I would have liked to try using Keras' data generator function as well as the training set was quite small and the network was often stuck in local optima. I would also have liked to use a clustering method to extract the images embedded in the test images, as the method I used was quite simple, albeit working surprisingly well.

3 Character Classification


d) Initially, I thought that a CNN be quick to train and work well on the images because of the images only having a single channel and small dimensions. I expected the MLE to do worse than the CNN because both uppercase and lowercase letters were grouped together in a single class and the predictions are made from a single multivariate gaussian while the network has many more dimensions.


e) The matlab program starts off running PCA on every image in the training set. This was done by first calculating the mean of every single image in the training set, subtracting this from each image, and then finding a covariance matrix. The eigenvectors of the covariance matrix were then found and a subset chosen based on the magnitude of the corresponding eigenvalues. Each image was then transformed by the eigenvector subset to a lower dimension. After running PCA, the mean and covariance of the images in each separate class are found to construct multivariate gaussian distributions for each class. Images can then be predicted by looking at each distribution and picking the class with the biggest likelihood.

The CNN consists of several convolutional and maxpooling layers with dropout functionality. The end of the network uses a series of fully connected *ReLU* layers while the output layer consists of 26 normalizing softmax nodes. The output layer was chosen so that the network would output a likelihood vector. The CNN was trained with the ADAM optimizer and a categorical crossentropy loss function.

I simply chose the CNN model because it is something I am fairly familiar with, while the MLE was chosen as something less familiar, but interesting new approach.

f) The network was run for 100 epochs on 80% of the data and tested on the remaining 20%. It did not perform as well as initially expected by being slow to train and did not correctly classify more than 70% of the test set. The MLE on the other hand did much better than expected, with an accuracy of between 80 and 84%.

The following were classified wrongly by the MLE as an *e* and by the network as an *o* respectively: 

These were classified correctly 

g) I would have liked to try implementing both kNN and decision trees for this task as I am quite curious how well they would do, yet I would have liked to further improve both my network and MLE with additional preprocessing. This was however not the case for my program and training went slow, while only achieving an accuracy of 73% at best.

4 Character Detection

My MLE got the highest accuracy and so I decided to test the detection images on this model.

h) All characters were extracted correctly from the detection images. As an example:



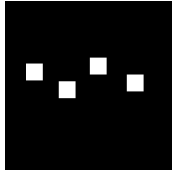
For detection image 1 every single character was correctly classified, while 91.7% were correctly classified in detection image 2.

i) Based on the findings mentioned above, I deem the detection system to perform quite well. The accuracy of 80% on the test images were a bit low however, but some additional preprocessing of the data would probably aid the model well. This could also be applied to remove the need to supplying the model with the amount of characters seen in a detection image.

j) I began implementing additional preprocessing to the detection images through the image segmentation matlab toolbox. I was not able to test it further, but I believe it would have improved the accuracy of the model

T
E S
T

MACHINE LEARNING
AND
CASE BASED
REASONING



(a) binary image



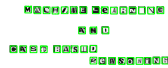
(b) bounding boxes



(c) extracted characters



(d) binary image



(e) bounding boxes



(f) extracted characters

Figure 1: Using MATLAB's Image Processing Toolbox