Fundamentals in statistical pattern recognition - Mini project

Pen-Digit Character Recognition

Group 7: Tiago Freitas Pereira, Braun Fabian, Marija Nikolić

May, 2015

1 Project description

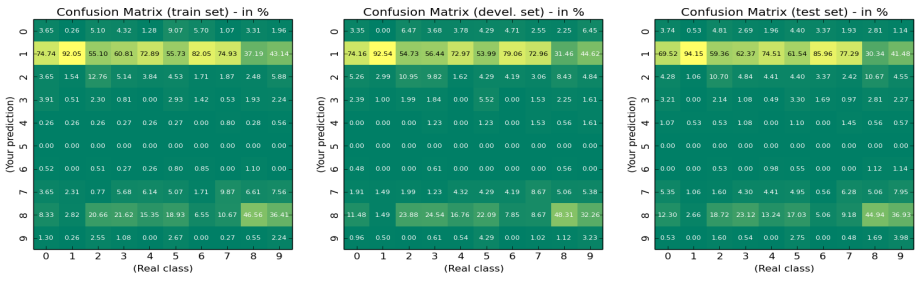
The objective of this project is to create a system for recognizing hand-written digits using statistical pattern recognition techniques. The available data was collected using a track-pad and a stylus. It contains the information of the trace each person did for writing down the numbers. The points are given in drawing order. Although the information about the point-by-point drawing order may potential improve recognition, we have decided not to use it in our project. We reserve 3748 examples for training, 1873 examples for development and 1873 examples for testing.

As a baseline we use the neural networks (Multi-Layer Perception) with hyperbolic tangent activation function and the Mean-Square Error cost. The network has three hidden units, whose weights are randomly initialized (seed=0). In the baseline the regularization parameter of 0.001 is used. The classification error rates (CERs) for training, development and testing data set corresponding to the baseline solution are listed below:

- CER (train set) = 82.8175%
- CER (development set) = 82.5414%
- CER (test set) = 83.6626%

The confusion matrix is shown in Figure 1.

Figure 1: Confusion matrix (Baseline)



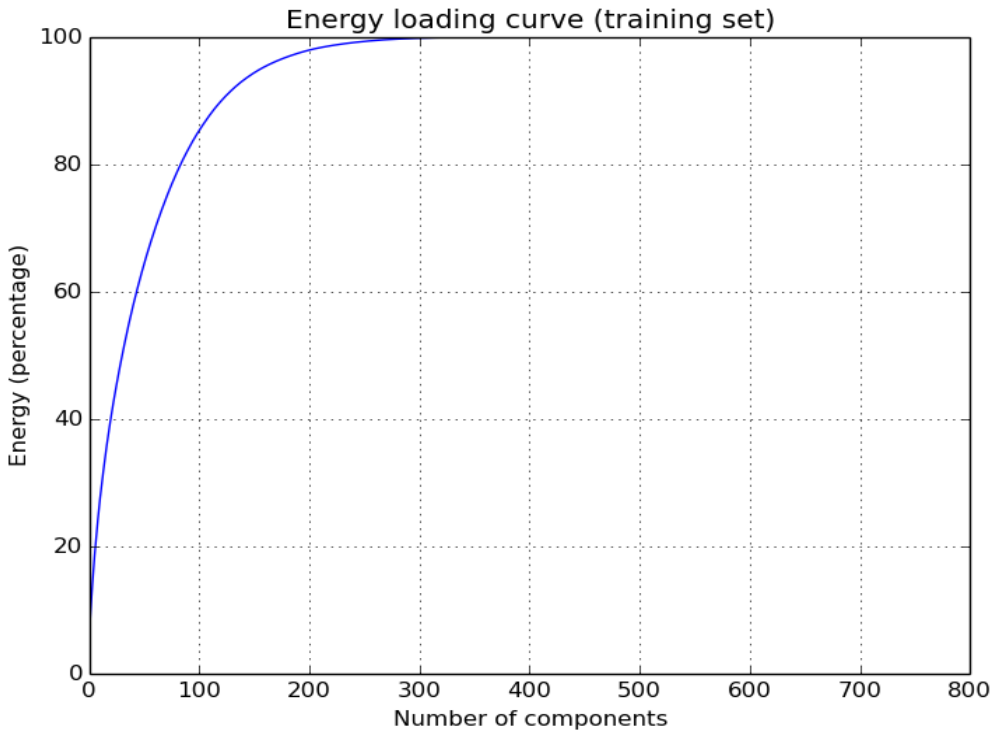
In order to improve the recognition performance we have studied and implemented three different techniques. The first technique (Section 3) makes use of Principal Component Analysis

and Multi-Class Logistic Regression (PCA-MCLR). In the second technique (Section 4) we employ Linear Discriminant Analysis together with Principal Component Analysis (PCA-LDA). The third evaluated technique (Section 5) refers to the Multi-Layer Perception enhanced through the use of Principal Component Analysis (PCA-MLP). Curse of dimensionality represents the main motivation of using Principal Component Analysis (Section 2) in all studied techniques.

2 Principal Component Analysis

The available data set consists of high-dimensional feature vectors (784×1) that are difficult to handle. Moreover, it is not possible to make sense of all variables and to understand the possible correlations. Therefore, we consider data in terms of its principal components, or equivalently, in terms of the directions where the data is most spread out. Dimension reduction is achieved by discarding the directions with the small spread. For these purpose we consider so-called energy load curve (Figure 2), which describes the percentage of the captured data variability with respect to the number of principal components. Figure 2 shows that, for instance, 100, 200 and 300 principal components preserve 85.03%, 97.89% and 99.8% of energy, respectively.

Figure 2: Energy load curve (PCA)



3 Multi-Class Logistic Regression and Principal Component Analysis

We consider a multi-class classification problem where the goal is to assign an input vector X to one of K classes. The number of classes (K) is equal to 10 (10 digits for classification) and input feature vector (X) is rearranged as 784-vector. Here we use MCLR together with PCA (PCA-MCLR), meaning that the input vector (X) is projected to a new space determined using PCA.

Logistic regression (LR) is a classification method with the classifier defined as

$$class = \begin{cases} 0, & h_\theta(x) < 0.5 \\ 1, & h_\theta(x) \geq 0.5, \end{cases} \tag{1}$$

where

$$h_\theta(x) = g(\theta^T x), \theta^T = [\theta_0 \theta_1 ... \theta_d], \tag{2}$$

with $g(x)$ being the logistic function

$$g(x) = \frac{1}{1 + \exp(-x)}. \tag{3}$$

Logistic regression can be easily generalized to multiple classes (MCLR). We train LR classifier $h_\theta^K(x)$ for each class K (K=1,...,10,) to predict the probability that class is equal to K (K vs. the rest). To make a prediction on a new input vector we choose the class such that

$$class = \arg\max_K h_\theta^K(x), K = 1, ..., 10. \tag{4}$$

We have analyzed the performance of the PCA-MCLR using different number of principal components. The best performance (based on the test set) is achieved using 200 principal components and the corresponding classification error rates (CERs) are listed below:

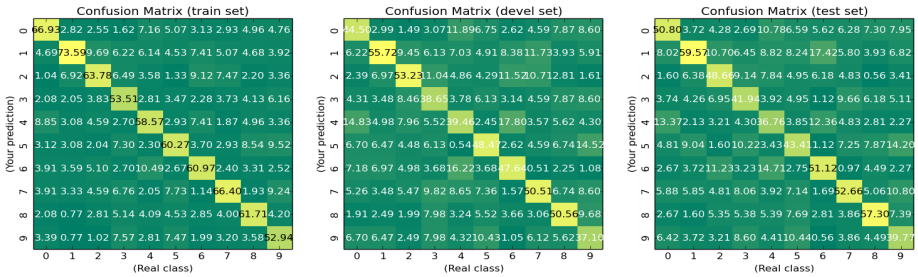
- CER (train set) = 38.0203%
- CER (development set) = 53.2301%
- CER (test set) = 51.8420%

The confusion matrix for training, development and testing data set is shown in Figure 3.

Note that with 200 principal directions, 97.89 % of data variability is preserved.

Compared with the the baseline solution (Section 1), significant improvement is achieved using PCA-MCLR method.

Figure 3: Confusion matrix (PCA-MCLR)



Instructions to reproduce results

$ git clone https://github.com/tiagofrepereira2012/FSPR_miniproject .
$ cd marija/MiniProj_MCLR_PCA/
$ ./mclr_train.py zero -c 200
$ ./mclr_train.py one -c 200
$ ./mclr_train.py two -c 200
$ ./mclr_train.py three -c 200
$ ./mclr_train.py four -c 200
$ ./mclr_train.py five -c 200
$ ./mclr_train.py six -c 200
$ ./mclr_train.py seven -c 200
$ ./mclr_train.py eight -c 200
$ ./mclr_train.py nine -c 200

$ ./mclr_evaluate.py zero.hdf5 one.hdf5 two.hdf5 three.hdf5 four.hdf5 five.hdf5 six.hdf5 seven.hdf5 eight.hdf5 ninie.hdf5 -c 200 --test -p

-c: Number of principal components to keep
--test: Evaluates the performance on test set
-p: Visualizes confusion matrices graphically

4 Linear Discriminant Analysis and Principal Component Analysis

The technique for recognizing hand-written digits described in this section combines LDA and PCA (PCA-LDA). As in the previous technique, PCA is used so as to achieve dimensionality reduction. The assumption that we make for classification is that the projection of the data so that the spread is maximized, leads to better discrimination capabilities. In LDA, we are interested to find a subspace in which the variability is maximized between data from different classes, and at the same time the variability in the same class is minimized. In a new subspace we create average models for each digit from the train data set and evaluate performance of the models on the three

data sets using the cosine similarity. The cosine similarity between two vectors p and q is given as

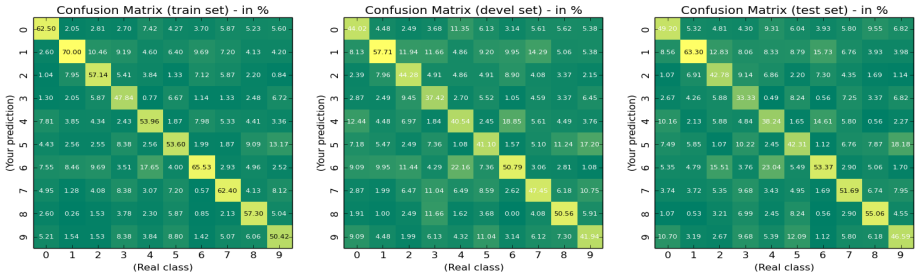
$$s(p, q) = \frac{p \cdot q}{|p||q|}. \tag{5}$$

We have chosen to keep the maximum possible number of LDA components (number of classes - 1 = 9). As for the number of PCA components, different variants were tested. The best performance (based on the test set) is achieved using 200 principal components and the corresponding classification error rates (CERs) are listed below:

- CER (train set) = 41.8890%
- CER (development set) = 54.1911%
- CER (test set) = 52.4826%

The confusion matrix for training, development and testing data set is shown in Figure 4.

Figure 4: Confusion matrix (PCA-LDA)



The results presented for PCA-MCLR and PCA-LDA suggest that better recognizing performance is achieved using the former method.

Instructions to reproduce results

```
$ git clone https://github.com/tiagofrepereira2012/FSPR_miniproject .
$ cd marija/MiniProj_PCA_LDA/
$ ./pca_lda.py -c 200 -l 9 -t -p -e

# -c: Number of principal components to keep
# -l: Number of LDA components to keep
# -t: Evaluates the performance on test set
# -p: Visualizes confusion matrices graphically
# -e: Visualizes energy load curve graphically
```

5 Multi-Layer Perception and Principal Component Analysis

This approach uses a combination of Multi-Layer Perception (MLP) with a preceding Principal Component Analysis (PCA, described in section 2) as illustrated in figure 5. Similar to the approach reported in section 4, the PCA is used to reduce the dimensionality of the input data $\mathbf{X}$ and thus improves the distinguishability of the data fed to the MLP $\mathbf{X}'$.

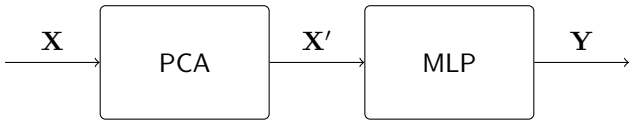


Figure 5: Block diagram of the MLP-PCA approach.

1. **Best results** with stable behaviour over different seeds show CERs of:

$$\text{CER}_{\text{Train}} \approx 65\% \qquad \text{CER}_{\text{Devel}} \approx 66\% \qquad \text{CER}_{\text{Test}} \approx 67\%$$

Instructions to reproduce results
$ git clone https://github.com/tiagofrepereira2012/FSPR_miniproject .
$ cd tiago
$./run_stable.sh

2. **Better results** - *however unstable over different seeds and most probably overfitted* - show CERs of (for one specific seed):

$$\text{CER}_{\text{Train}} = 1.84\% \qquad \text{CER}_{\text{Devel}} = 20.2\% \qquad \text{CER}_{\text{Test}} = 21.0\%$$

As this approach uses 107 principal components and 67 neurons we have 7916 free parameters with 3397 feature vectors in the training set, which is a potential case of overfitting which should better be avoided.

Instructions to reproduce results
$ git clone https://github.com/tiagofrepereira2012/FSPR_miniproject .
$ cd fab
$ ./run_overfit.sh

A remark to this approach:

- As favourable for MLPs, the input data $\mathbf{X}'$ was normalized with the standard deviation of each individual feature. However, this modification did not lead to an improvement of the results and was therefore rejected.