Offline Handritting Word Recognition

Thijs Kooi, Davide Modolo

January 26, 2011

Table of contents

Overview of the Project

Off-line handwriting recognition

- It involves the automatic conversion of text in an image into letter codes which are usable within computer and text-processing applications
- Off-line handwriting recognition is comparatively difficult, as different people have different handwriting styles

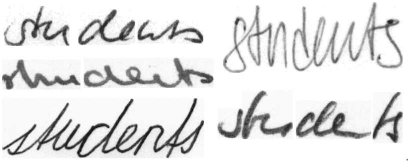


Figure: 'Students' written by different authors

Our AI Project

A lot of research has been done over the past years.

We explored the topic and implemented a full pipeline for the task. The research touched different fields:

- Data Collection
- Image Processing
- Features extraction
- Machine Learning techniques
- Word Recognition using Hidden Markov Models

Dataset

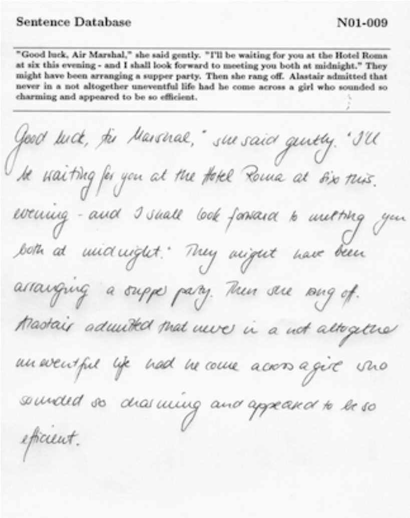
The IAM Handwriting Database[1]

- Unconstrained handwritten text (scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels)
- 1'539 pages of scanned text of 657 writers

- We extracted 35 instances of the 300 most frequent words (20 as training set and 15 as test set)

[1] http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

Example of a page of scanned text



Sentence Database N01-009

"Good luck, Air Marshal," she said gently. "I'll be waiting for you at the Hotel Roma at six this evening - and I shall look forward to meeting you both at midnight." They might have been arranging a supper party. Then she rang off. Alastair admitted that never in a not altogether uneventful life had he come across a girl who sounded so charming and appeared to be so efficient.

Good luck, Air Marshal," she said gently. "I'll
be waiting for you at the Hotel Roma at six this
evening - and I shall look forward to meeting you
both at midnight." They might have been
arranging a supper party. Then she rang off.
Alastair admitted that never in a not altogether
uneventful life had he come across a girl who
sounded so charming and appeared to be so
efficient.

Implementation Details

Pipeline

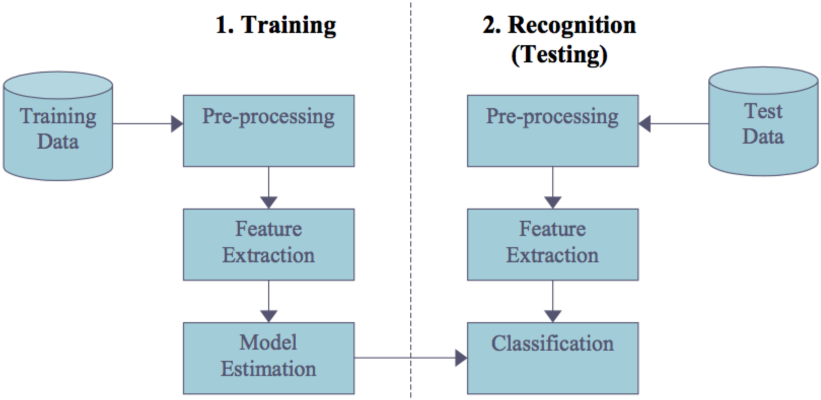


Figure: Pipeline of a word recognition system

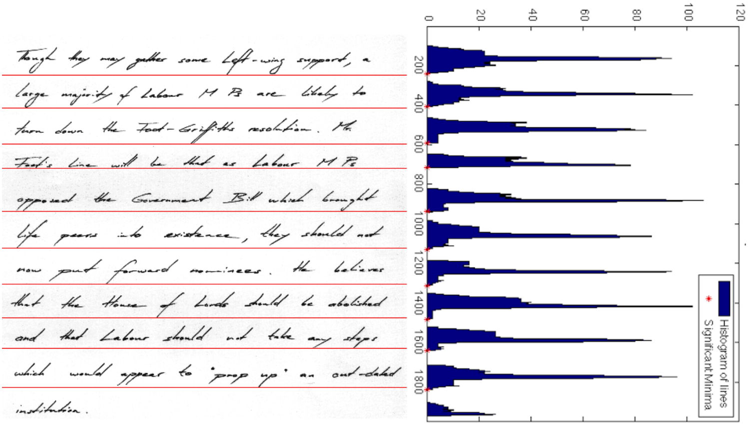
Implementation Details

Pre-Processing

Pre-processing

Images/pipelinePP.png

Line Segmentation



Original image segmented in lines

Vertical histograms and significant minima

Figure: Example of line segmentation

Skew and Slope Correction

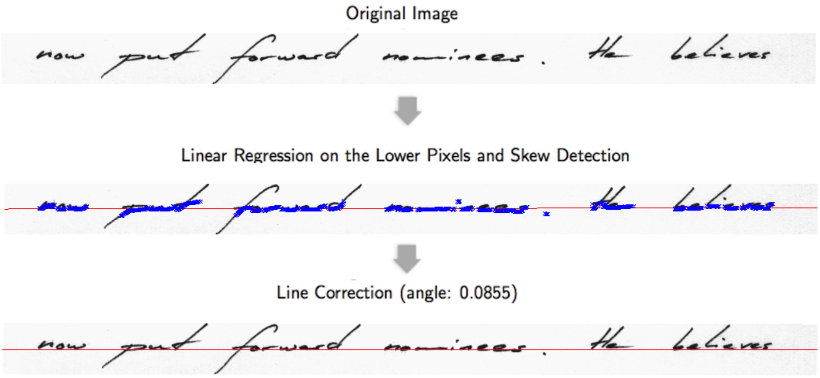


Figure: Skew detection and correction pipeline

Slant detection and Correction

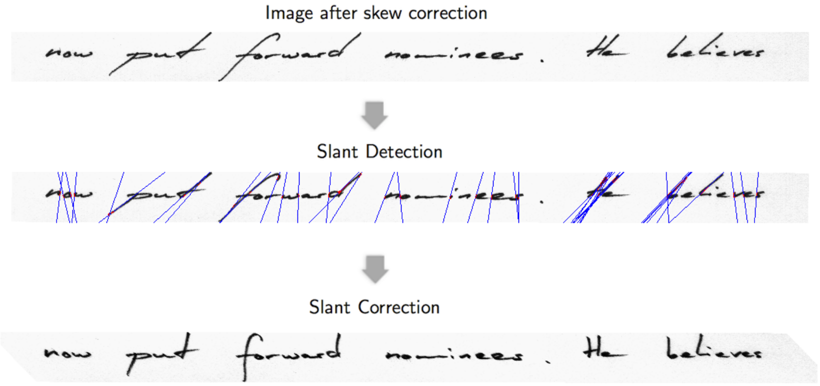


Figure: Slant detection and correction pipeline

Word segmentation

- We first detect white spaces:

Images/words_segm_wp.png

Vertical Scaling

Skeletonization

Remembering the entire pipeline....

Why do we repeat skew and slant corraction twice?

Images/pipelinePPex.png

Because....

Images/pipelinePPex.png

Implementation Details

Feature Extraction

Features

We want to find features that minimise the within-class variability and maximise the between class variability. On top of this, the features should be robust againts distortions caused by different handwriting styles. Moreover, we want to find low dimensional featur vectors and would therefore like features to be highly descriptive. The selection of features depends both on the pre-processing and the classifier to use. If all characters are assumed to have the same oriention, we need rotation variant features to distinguish between for instance a 6 and a 9 and a b and an p, etc.

HMM

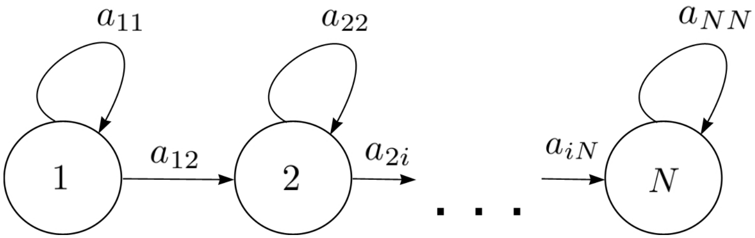


Figure: Left-to-right HMM with N states

Main problems in an HMM

1. The probability of an observation sequence, given the model, $P(\mathbf{O}|\lambda)$.
2. The most likely parameters of the model $\lambda^* = \max P(X|\lambda)$, given a training set of M observation sequences $X = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_M)$.
3. The most likely state sequence, underlying a given observation sequence and the model, $Q^* = \max P(Q|\mathbf{O}, \lambda)$.

Main problems in an HMM

1. The probability of an observation sequence, given the model, $P(\mathbf{O}|\lambda)$.

   Sum-product algorithm: forward-backward algorithm

2. The most likely parameters of the model $\lambda^* = \max P(X|\lambda)$, given a training set of M observation sequences $X = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_M)$.

   EM-algorithm: Baum-Welch reestimation

3. The most likely state sequence, underlying a given observation sequence and the model, $Q^* = \max P(Q|\mathbf{O}, \lambda)$.

   Dynamic programming: Viterbi algorithm

Forward probability

$$\alpha_t(i) \equiv P(o_1, o_2, \ldots, o_t | q_t = s_i, \lambda) = \left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij}\right] b_j(o_t) \quad (1)$$

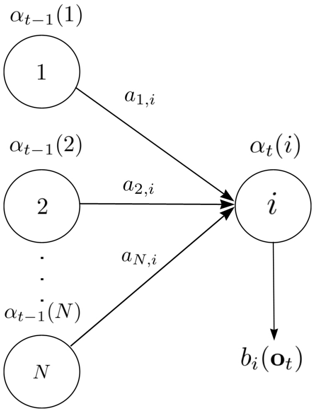


Figure: Computation of forward probability

Forward probability

$$\alpha_t(i) \equiv$$
$$P(o_1, o_2, \ldots, o_t | q_t = s_i, \lambda) =$$
$$\left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij} \right] b_j(o_t) \tag{2}$$

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3}$$

Problem 1 solved.

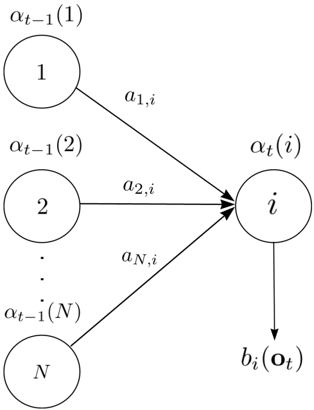


Figure: Computation of forward probability

Updating the parameters

The forward-backward algorithm also commes with a backward probability:

$$\beta_t(j) \equiv P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = s_j, \lambda) = \sum_{i=1}^{N} a_{ij} b_i(o_{t+1}) \beta_i(o_{t+1}) \tag{4}$$

Updating the parameters

Comparison with GMM:

Model:	**GMM**	**HMM**
Model parameters:	$\lambda = \pi, \mu, \Sigma$	$\lambda = \pi, \mathbf{A}, \mathbf{B}$
Hyper parameters:	Number of components	Topology (states, transitions)
Observed variables:	Data points	Observations
Latent variables:	Priors of a component	State sequence

Updating parameters

Model:	**GMM**	**HMM**
Model parameters:	$\lambda = \pi, \mu, \Sigma$	$\lambda = \pi, \mathbf{A}, \mathbf{B}$
Hyper parameters:	Number of components	Topology (states, transitions)
Observed variables:	Data points	Observations
Latent variables:	Priors of a component	State sequence
E-step:	Estimate the probability of a component, given the data and current parameters.	Estimate the probability of being in a state at a timestep and the probability of transfering from a state to another state.
M-step:	Maximise π, μ and Σ.	Maximise π, $\mathbf{A}$ and $\mathbf{B}$

Problems we ran into

Singularities. Consider the following observation:

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

Problems we ran into

Singularities

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 3\frac{1}{3} & 0 \\ 0 & 0 \end{pmatrix}$$

Problems we ran into

Singularities

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

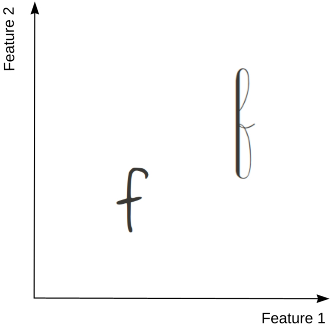
$$\Sigma = \begin{pmatrix} 3\frac{1}{3} & 0 \\ 0 & 0 \end{pmatrix}$$

-¿ Add some random noise, also to prevent variance from collapsing.

Problems we ran into

Short words have less likelihood.
Harder to recognise, more subject to writer variations.
tried to solve this by using MOG.

Within class variations



Using features such as loops, this will give quite different feature vector.

Within class variations

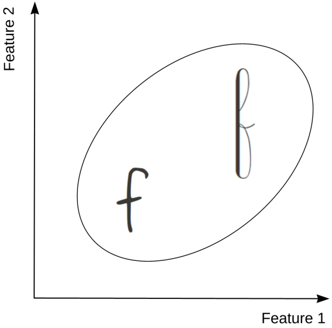


Figure: Fitting a single Gaussian

Within class variations

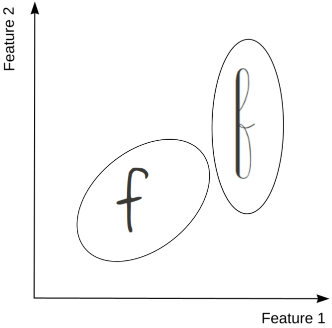


Figure: Fitting a mixture of 2 Gaussians

Results

blabla

Conclusions

blabla

blocs

title of the bloc

bloc text

title of the bloc

bloc text

title of the bloc

bloc text