Graphical Modelling Spring Term 2014
Coursework (due April 28th)

Anagnostopoulos, C., www.canagnos.com/teaching

June 27, 2014

1 Reading characters

We may represent handwritten characters as *bitmaps*, i.e., in the space $\mathcal{M}$ of $p \times q$ matrices with binary entries. We may define the following distance metric in $\mathcal{M}$:

$$d(v, w) = \frac{1}{m} \sum_{i,j} |v_{ij} - w_{ij}| \in [0, 1]$$

where $m = pq$. Assume now that we are given a template w^k for each character in the latin alphabet, including the space character, so that $k \in \{1, \ldots, 27\}$. We can express the probability that a given bitmap v is a 'noisy' version of w^k by:

$$P(v \mid w^k) \propto 1 - d(v, w^k) \tag{1}$$

A.1 Determine the normalisation constant for (1). Is this factor the same if the template w^k has entries in $[0, 1]$ rather than binary?

Download the compressed folder "Data" from the digital blackboard. It contains 55 samples of each lower-case letter in the English alphabet in '.png' format (i.e., 1430 files in total). You may load this data into `R` as follows:

```
> library(png)
> M <- list(a=list(),b=list())
> M$a[[1]] <- readPNG('a_01.png')
> M$a[[2]] <- readPNG('a_02.png')
> M$b[[1]] <- readPNG('b_01.png')
> M$b[[2]] <- readPNG('b_02.png')
```

and so on, creating a list of lists of matrices. These matrices can also displayed as images using the `image` command – see Figure 1. Note that to reproduce these plots you need to rotate the matrices using the function:

```
> my.rotate <- function(m) t(m)[,nrow(m):1]
```

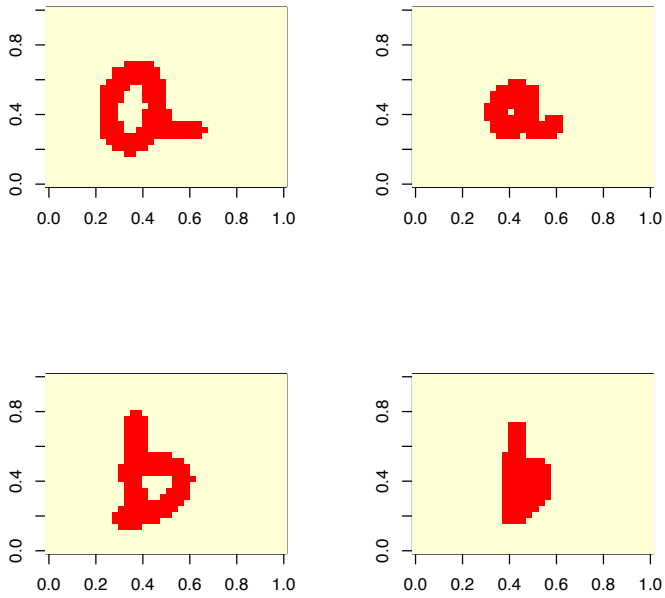

Figure 1: Four example PNG files, two for each writer.

A.2 Henceforth we refer to the 'alphabet' as the collection of all lower-case English letters, as well as the special character 'space'. Let w^k for $k = 1$ to 26 be the average of the training samples for letter k, and w^{27} be the zero matrix. Using the distribution from A.1, and a uniform distribution over all the letters, find the MAP setting for the label of the sample 'a_02.png'. Repeat this exercise over all samples for 'a' and report the average error rate. (Hint: careful not to divide by huge numbers!)

A.3 Now consider the background probabilities of occurrence of each letter in the alphabet, which you can obtain from `first_order.txt` (courtesy of `http://www.data-compression.com/english.shtml`):

```
> first_order <- read.csv('first_order.txt',sep=' ',header=TRUE)
```

Modify your MAP assignment above to incorporate this information. How do the results differ? Briefly comment on potential reasons for this behaviour.

A.4 Fit a Naïve Bayes classifier via ML, and redo the MAP calculations using the non-uniform background-frequency prior. Compare the accuracy to part A.3.

A.5 Modify $v^{a,2}$ by changing its first entry only:

$$\tilde{v}^{a,2} = \begin{cases} 1 - v^{a,2}_{i,j}, & \text{if } i = j = 1, \\ v^{a,2}_{i,j}, & \text{otherwise.} \end{cases}$$

Inspect the estimated probabilities for $P(k \mid \tilde{v}^{a,2})$ (possibly up to a normalising factor, as you have done so far). What is the problem? How can you fix it?

2 Reading words

We now consider sequences of handwritten characters instead. We will make use of second-order statistics to help us revise class allocations that lead to unlikely pairs of consecutive characters (e.g., 'aa'):

```
> second_order <- read.csv('second_order.txt',sep=' ',row.names=1,header=TRUE)
```

Coding hint: to perform simple algebra with `first_order` and `second_order` it might be needed to convert them to a simpler data structure using `as.numeric()`.

B.1 Since we know in advance there will not be any spaces in our data, we need to condition on that fact. Describe what needs to be done and implement this change in your code.

B.2 Consider a Markov Chain taking values in the alphabet indexed by $k \in \{1, \ldots, 26\}$; transition matrix given by the second-order statistics; prior

$P(H_1)$ given by the first-order statistics (suitably constrained to ban spaces). Rank the following three words in order of likelihood under this model:

$$\text{laarning},\ \text{learning},\ \text{learxing}$$

B.3 We now assume the MC of B.2 is hidden, and only observed with 1% probability of error, resulting in a sequence of letters $V_1, \ldots, V_N \in \{1, \ldots, 26\}$:

$$P(V_i \mid H_i = k) = \begin{cases} 0.99, & \text{if } V_i = k, \\ 0.01, & \text{otherwise.} \end{cases}$$

Code up the Viterbi algorithm for identifying the MAP sequence $\hat{h}_1, \ldots, \hat{h}_N$, given a sequence $v_1, \ldots, v_N$. Produce the MAP assignment of the three words from part B.2.

B.4 Select the first sample from the training dataset for each letter appearing in the word `amazed` (you may inspect the bitmaps in question in Figure 2). Produce the MAP estimate of each bitmap (independently) using the Naïve Bayes model of Part A. Input the resulting sequence of letters into your model from B.3, and produce its MAP assignment.

B.5 Consider now an HMM with the same hidden state space, but where the observation at each time is a bitmap, rather than a letter. In B.4 we used the Naïve Bayes model as a preprocessing step. We now incorporate it into the HMM by treating it as the emission probability instead. Draw the respective Bayesian Network. Draw its induced graph.

B.6* For this question, you may use the ML estimates of p_{ij}^k in the Naïve Bayes model without further justification:

$$P(V_t = v \mid H_t = k) = \prod_{i,j} \hat{p}_{ij}^k$$

Modify your code from B.4 to reflect the modified observation space and emission model of B.5. Hence obtain the Viterbi sequence for the sequence you used in B.4.

Attention: question B.6* is a **bonus** coding question, i.e.,

- it receives **no marks** for this assignment
- but it will be used as evidence of applied skill in your reference letter