Introduction to Probabilistic Graphical Models
Mini Project – Barcode Decoding with HMMs

Umut Şimşekli
Télécom ParisTech, Université Paris-Saclay, Paris, France
umut.simsekli@telecom-paristech.fr

Instructions: (please read carefully)

1. This project can be done in groups of **maximum 4** people. I personally encourage group work, try to form groups.
2. Prepare your report in English by using LaTeXor an ipython (jupyter) notebook. Do not submit scanned papers.
3. Put all your files (code and/or report) in a zip file: *surname_name_miniproject.zip* and upload it to moodle before the deadline (check moodle for the deadline). Late submissions will not be accepted.
4. The initial code is written in Matlab. It should work with Octave as well. You can code in python if you wish, but you would need to write everything from scratch.

1 Introduction & Background

One of the key technologies in closing the producer-consumer gap is product identification and traceability, which is currently achieved by the most prevalent technology: linear (1D) barcode scanning. The mass manufacturing, as well as the consumer goods market, regardless of the sector rely on barcodes, especially linear codes. An example barcode is given in Figure 1.

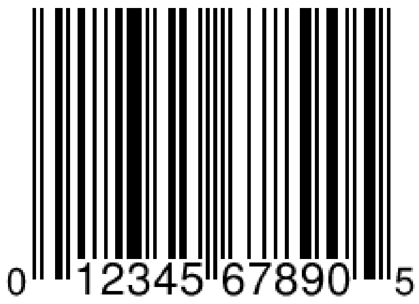

Figure 1: An example 1D (linear) barcode.

In manufacturing systems, high accuracy and speed lead to high throughput and more profit and thus they require robustness, durability, and noise tolerance in barcode scanning systems. At this point conventional laser scanners loose their charm, as they mostly fail to read multiple codes sequentially and necessitate customization of the conveyor

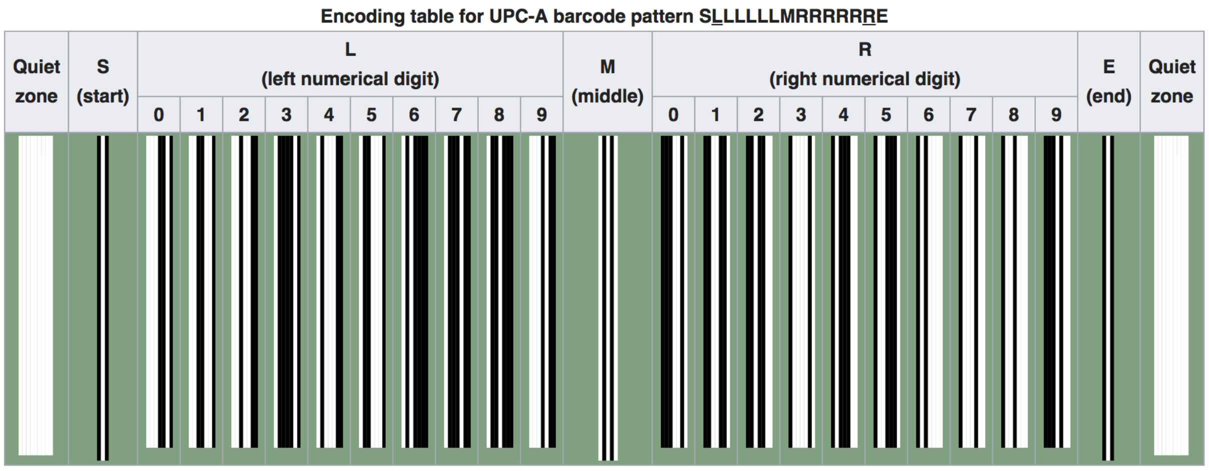
Encoding table for UPC-A barcode pattern SLLLLLLMRRRRRRE
Quiet zone
S (start)
L (left numerical digit)
0 1 2 3 4 5 6 7 8 9
M (middle)
R (right numerical digit)
0 1 2 3 4 5 6 7 8 9
E (end)
Quiet zone

Figure 2: The encoding of the digits in UPC-A.

belts. Instead, machine vision solutions are desired, where the cameras could capture multiple barcodes in arbitrary orientations and decode them. However, the performance requirements of those systems crave for a broad solution, which is able to decode various types of barcodes in difficult scenes, located on different surfaces, and posed under varying constraints.

From the consumers perspective, it is noticeable that in recent years, there has been a huge attempt in empowering end-users with cheap software tools to benefit from barcodes. A customer can now get more information about a product or can experience the entire shopping process without the burden of a shopping cart. It is also worth mentioning that as barcodes implicitly describe products, they are an important source of information for visually impaired people as well. Considering the uncontrolled setting of mobile barcode decoding through low resolution cameras, the performances of available software are far from what is desired.

The aim of this project is to develop a probabilistic model for modeling a specific type linear barcodes, called ‘UPC-A’. Our aim will be to infer the subsequent digits (we will sometimes refer to them as symbols) given a barcode scanline that is obtained from a gray-scale image. This is a challenging task since the observed scanline can be degraded due to noise and blur which often occur in practice.

The main idea in this model will be to incorporate the sequential and hierarchical information of the barcodes into a single dynamic Bayesian network – a hierarchical Hidden Markov Model (H-HMM). You are asked to explicitly model the different layers of the hierarchy of barcodes by using an H-HMM.

2 UPC-A Barcodes

UPC-A is a pervasive progressive barcode symbology with 11 digits (symbols), with an added 12^{th} digit for check-sum. It consists of alternating black-white stripe sequence with 4 possible bar widths. Each digit is encoded by a unique combination of such bars. The scannable area begins with a special start code and ends with the stop code. There are guard bars located in the middle, which optically inverses the encoding scheme to the right of the barcode. The overall length of UPC-A is 95 *base widths*, where a base width is the width of the thinnest bar.

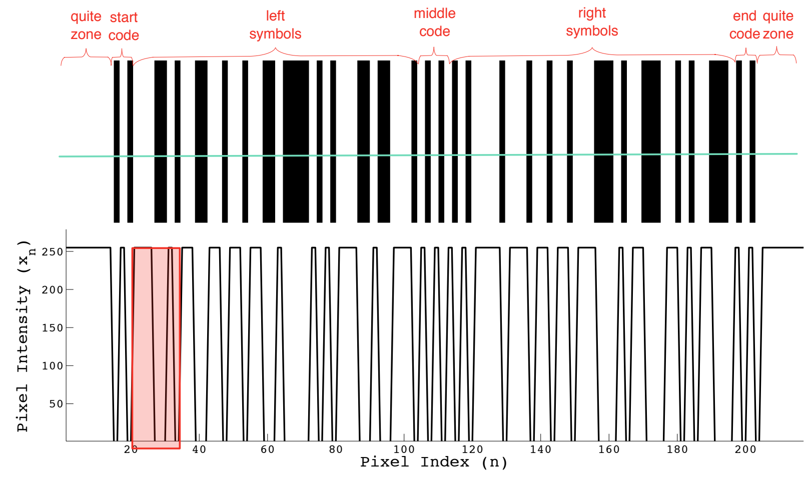

Figure 3: A sample UPC-A barcode (top), its corresponding scanline (bottom), and a symbol in the scanline (shaded).

Here is the Wikipedia definition of the UPC-A barcodes: (`https://en.wikipedia.org/wiki/Universal_Product_Code`)

The UPC-A barcode is visually represented by strips of bars (i.e. *black* bars) and spaces (i.e. *white* bars) that encode the UPC-A 12-digit number. Each digit is represented by a unique pattern of 2 bars and 2 spaces. The bars and spaces are variable width, i.e. 1, 2, 3, or 4 modules wide. The total width for a digit is always 7 modules; consequently, UPC-A 12-digit number requires a total of 712 = 84 modules.

A complete UPC-A is 95 modules wide: 84 modules for the digits (L and R sections) combined with 11 modules for the S (start), M (middle), and E (end) guard patterns. The S (start) and E (end) guard patterns are 3 modules wide and use the pattern bar-space-bar, where each bar and space is one module wide. The M (middle) guard pattern is 5 modules wide and uses the pattern space-bar-space-bar-space, where each bar and space is also one module wide. In addition, a UPC-A symbol requires a quiet zone (extra space of 9 modules wide) before the S (start) and after the E (end) guard patterns. The encodings of the digits are shown in Figure 2.

3 Probabilistic Modeling of UPC-A barcodes

We start by defining the observed scanline x_n, $n = 1, \dots, N$, where n is the pixel index, $x_n \in [0, 255]$ is the pixel intensity, and N is the length of the scanline. We assume that the scanline consists of successive non-overlapping symbols that can be of variable length, i.e., a symbol s will cover a subset of the pixels, say $x_{i:j} \equiv \{x_i, \dots, x_j\}$, and the collection of the symbols will cover the entire scanline $x_{1:N}$. Every pixel x_n belongs to a part of a symbol. An example UPC-A barcode, its corresponding scanline, and a symbol in the scanline are illustrated in Figure 3.

3.1 Random Variables

We model each pixel x_n in a scanline as a noisy observation whose underlying probability distribution is governed by some latent variables. We will now proceed to the definition of these variables. We define the 'symbol' variable $s_n \in D_s = \{1, \dots, S\}$, which denotes the symbol that the pixel x_n belongs to. S represents the number of symbols.

For UPC-A, the set of the symbols is given as $D_s = \{1, \ldots, S\} \equiv \{$*starting quiet zone, ending quiet zone, start code, end code, middle code, left digits (0, ..., 9), right digits (0, ..., 9)*$\}$ and $S = 25$. In this notation, $s_n = 4$ is equivalent to $s_n =$ 'end code'. Our ultimate aim is to find the most likely sequence of these variables.

A UPC-A barcode always has the following sequential form:

1.) Starting quiet zone
2.) Start code
3.) Left symbol 1
4.) Left symbol 2
5.) Left symbol 3
6.) Left symbol 4
7.) Left symbol 5
8.) Left symbol 6
9.) Middle guard
10.) Right symbol 1
11.) Right symbol 2
12.) Right symbol 3
13.) Right symbol 4
14.) Right symbol 5
15.) Right symbol 6 (for the checksum)
16.) End code
17.) Ending quiet zone

In the UPC-A symbology, the number of symbols that can be present in a single barcode is predetermined: there must be 6 digits in between the start code and the middle code, and there must be 6 other digits in between the middle code and the end code. In order to be able to inform our model about the number of symbols that can occur in a scanline, we define the 'index' variable $m_n \in \{1, \ldots, M\}$ that denotes the index of the symbol s_n. For UPC-A, it is sufficient to set $M = 6$ since the left and right digits are separated from each other by the middle code.

The lengths of the symbols differ in UPC-A. For instance the length of the start code is three base-widths (consists of three bars: black-white-black, each bar covering one base-width) whereas the length of a digit symbol is seven base-widths. Therefore, we need to define the mapping $\ell(s_n)$ that determines the length of s_n in pixels. For instance, suppose that the base-width for a particular scanline is 1 pixel (this will always be the setting in this project). Then, the start code would cover 3 pixels.

In order to determine which part of s_n that x_n belongs, we define a 'counter' variable $c_n \in \{1, \ldots, \ell(s_n)\}$, where $c_n = 1$ implies that the symbol s_n starts at the pixel n and $c_n = \ell(s_n)$ implies that s_n is ending at pixel x_n; i.e., there will be a new symbol starting at pixel $n + 1$.

3.2 Transition Model

We assume that within the region of a symbol s_n, the counter variable c_n starts from 1 and increases by one at each pixel until it hits the length of the current symbol $\ell(s_n)$. This deterministic evolution is incorporated in the model by placing a degenerate prior distribution over the counter variables as follows:

$$p(c_n|s_n, c_{n-1}) = \begin{cases} \delta(c_n - c_{n-1} - 1), & c_{n-1} \neq \ell(s_n) \\ \delta(c_n - 1), & c_{n-1} = \ell(s_n) \end{cases} \tag{1}$$

Here, $\delta(\cdot)$ is the Kronecker-delta function where $\delta(s) = 1$ if $s = 0$ and $\delta(s) = 0$ otherwise.

Next, we assume the following prior distribution over the symbol variables:

$$p(s_n|s_{n-1}, m_{n-1}, c_{n-1}) = \begin{cases} \delta(s_n - s_{n-1}), & c_{n-1} \neq \ell(s_n) \\ \tau_s(s_n|s_{n-1}, m_{n-1}), & c_{n-1} = \ell(s_n) \end{cases} \tag{2}$$

where $\tau_s(s_n|\cdot)$ is the transition distribution for the symbol variables that incorporates the rules of a particular barcode symbology to the model. The prior distributions $p(c_n|\cdot)$ and $p(s_n|\cdot)$ together ensure that the symbol variable s_n stays the same within the symbol's region (i.e., for $\ell(s_n)$ pixels) since s_n must be equal to s_{n-1} until c_{n-1} becomes 1. In other words, during the presence of a particular symbol, the counter variable decreasingly counts down to 1. When $c_{n-1} = 1$, the symbol will transition to another symbol where this transition is governed by $\tau_s(s_n|\cdot)$.

Accordingly, we define the transition distribution for the symbol variables as follows:

$$\tau_s(s_n|s_{n-1}, m_{n-1}) = \begin{cases} \textbf{1.)} & \mathcal{U}(\{1, 3\}), & s_{n-1} = 1 \\ \textbf{2.)} & \mathcal{U}(\{6, \dots, 15\}), & s_{n-1} = 3 \\ \textbf{3.)} & \mathcal{U}(\{6, \dots, 15\}), & m_{n-1} \neq 6 \wedge 6 \leq s_{n-1} \leq 15 \\ \textbf{4.)} & \delta(s_n - 5), & m_{n-1} = 6 \wedge 6 \leq s_{n-1} \leq 15 \\ \textbf{5.)} & \mathcal{U}(\{16, \dots, 25\}), & s_{n-1} = 5 \\ \textbf{6.)} & \mathcal{U}(\{16, \dots, 25\}), & m_{n-1} \neq 6 \wedge 16 \leq s_{n-1} \leq 25 \\ \textbf{7.)} & \delta(s_n - 4), & m_{n-1} = 6 \wedge 16 \leq s_{n-1} \leq 25 \\ \textbf{8.)} & \delta(s_n - 2), & s_{n-1} = 4 \\ \textbf{9.)} & \delta(s_n - 2), & s_{n-1} = 2 \end{cases} \tag{3}$$

where the state enumerations are defined in the beginning of the section (D_s) and $\mathcal{U}(\Sigma)$ denotes the discrete uniform distribution with support Σ. The respective verbal descriptions of the cases are given as follows:

1.) If the previous symbol is the starting quiet zone ($s_{n-1} = 1$), then the current symbol can be either the starting quiet zone ($s_n = 1$) or can transition to the starting code ($s_n = 3$).

2.) If the previous symbol is the start code ($s_{n-1} = 3$), then the current symbol must be one of the left digits ($s_n \in \{6, \dots, 15\}$).

3.) If the previous symbol is one of the left symbols ($s_{n-1} \in \{6, \dots, 15\}$) and the previous symbol is not the last digit before the middle guard ($m_{n-1} \neq 6$) (recall that $M = 6$) then the current symbol must be one of the left digits ($s_n \in \{6, \dots, 15\}$).

4.) If the previous symbol is one of the left symbols ($s_{n-1} \in \{6, \dots, 15\}$) and the previous symbol is the last digit before the middle guard ($m_{n-1} = 6$) then the current symbol must be the middle guard ($s_n = 5$).

5.) If the previous symbol is the middle guard ($s_n = 5$), then the current symbol must be one of the right digits ($s_n \in \{16, \dots, 25\}$).

6.) If the previous symbol is one of the right symbols ($s_{n-1} \in \{16, \dots, 25\}$) and the previous symbol is not the last digit before the end code ($m_{n-1} \neq 6$) then the current symbol must be one of the right digits ($s_n \in \{16, \dots, 25\}$).

7.) If the previous symbol is one of the right symbols ($s_{n-1} \in \{16, \dots, 25\}$) and the previous symbol is the last digit before the end code ($m_{n-1} = 6$) then the current symbol must be the end code ($s_n = 4$).

8.) If the previous symbol is the end code ($s_n = 4$), then the current symbol must be the ending quiet zone ($s_n = 2$).

9.) If the previous symbol is the ending quiet zone ($s_n = 2$), then the current symbol must be the ending quiet zone ($s_n = 2$).

We finally define the prior distribution of the index variables m_n as follows:

$$p(m_n|s_n, m_{n-1}, c_{n-1}) = \begin{cases} \delta(m_n - m_{n-1}), & c_{n-1} \neq \ell(s_n) \\ \tau_m(m_n|s_n, m_{n-1}), & c_{n-1} = \ell(s_n) \end{cases} \tag{4}$$

where $\tau_m(m_n|\cdot)$ is the transition distribution for the index variables, for UPC-A it encapsulates a straightforward collection of deterministic relations, given as follows:

$$\tau_m(m_n|s_n, m_{n-1}) = \begin{cases} \textbf{1.)} & \delta(m_n - 1), & s_n \in \{1, \dots, 5\} \\ \textbf{2.)} & \delta(m_n - m_{n-1} - 1), & s_n \in \{6, \dots, 15\} \wedge m_{n-1} \neq 6 \\ \textbf{3.)} & \delta(m_n - m_{n-1} - 1), & s_n \in \{16, \dots, 25\} \wedge m_{n-1} \neq 6 \end{cases} \tag{5}$$

The respective verbal descriptions of the cases are given as follows:

1.) If the current symbol is either the starting quiet zone, ending quiet zone, start code, end code, or the middle guard ($s_n \in \{1, \dots, 5\}$), the index variable must be set to 1 ($m_n = 1$). We only want the index variables to handle left and right digits.

2.) If the current symbol is one of the left symbols ($s_n \in \{6, \dots, 15\}$) and the previous symbol is not the last digit before the middle guard ($m_{n-1} \neq 6$), then we increment the current index variable by one by setting $m_n = m_{n-1} + 1$.

3.) If the current symbol is one of the right symbols ($s_n \in \{16, \dots, 25\}$) and the previous symbol is not the last digit before the end code ($m_{n-1} \neq 6$), then we increment the current index variable by one by setting $m_n = m_{n-1} + 1$.

3.3 Observation Model

After defining the latent variables and their relations by using the prior distributions, we also need to relate them to the observed scanline x_n through an observation model. We define the following observation model:

$$p(x_n|s_n, c_n) = \prod_{i=0}^{1} \mathcal{N}(x_n; \mu_i, \sigma_i^2)^{\mathbb{1}\left(f(s_n, c_n) = i\right)} \tag{6}$$

Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the normal distribution and $\mathbb{1}(\cdot)$ is the indicator function where $\mathbb{1}(x) = 1$ if x is true and $\mathbb{1}(x) = 0$ otherwise. The mapping $f(\cdot)$ determines the color of the bar (black or white) corresponding to a part of a symbol, where the part is determined by the symbol s_n and the pixel index of that particular symbol c_n. Here $f(\cdot) = 0$ if the corresponding bar is white and $f(\cdot) = 1$ if the corresponding bar is black. For instance, consider a UPC-A start code of 3 pixels long (i.e., $s_n = 3$, $\ell(s_n) = 3$). The mapping for this symbol is defined as follows: $f(s_n, c_n = 1:3) \equiv \{1, 0, 1\}$, meaning that the first and the last bars are black and the middle bar is white. The parameters μ_i and σ_i^2 are the mean and the variance for different colors.

If we consider the barcode given in Figure 1, the optimal state path for this particular barcode is given in Figure 4 for illustration purposes.

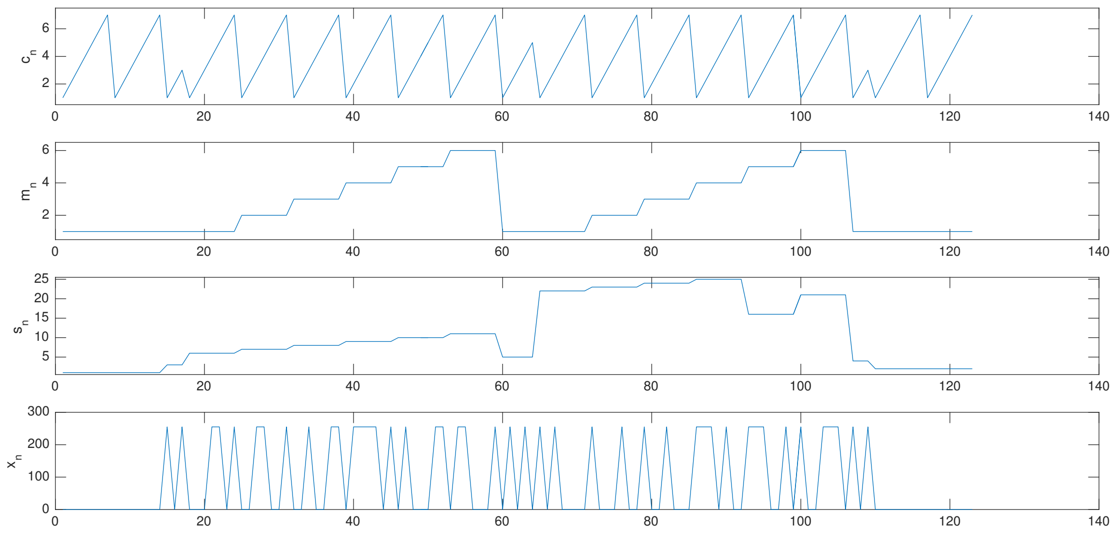

Figure 4: An example optimal state path for a UPC-A barcode. The x-axis is the pixel index (n). After estimating the states s_n and c_n, we can obtain the barcode string as '012345678905'.

Question 1

Draw the directed graphical model for the described model.

Question 2

We are interested in the most likely state trajectory given all the observations. For practical purposes, redefine the described model to an ordinary HMM. How can you construct the transition matrix? What will be the observation model?

Hint: Define the variable $\Psi_n \equiv [s_n, m_n, c_n]$, which encapsulates the state of the model at pixel n. The set of all possible states can be listed in a vector Ω and the state of the system at the pixel n can be represented as $\Psi_n = \Omega(j)$, where $j \in \{1, 2, \dots, (S \times M \times C)\}$ and $C = \max_{s,k} \ell(s)$. The transition matrix will have the following definition.

$$A(i, j) = p(\Psi_{n+1} = \Omega(i) | \Psi_n = \Omega(j)). \tag{7}$$

Try to describe $p(\Psi_{n+1} = \Omega(i) | \Psi_n = \Omega(j))$ by using the distributions defined in Section 3.2.

Question 3

By using the result of the previous question, simulate the HMM and visualize the simulated data. Use $\mu_0 = 250$, $\mu_1 = 20$, and $\sigma_0^2 = \sigma_1^2 = 5$. Does the simulated x_n look like a real scanline?

Question 4

Use the file 'template_code.m'. Fill the parts 1, 2, and 3.

Since we have reduced our model to a standard HMM, we can now make use of the forward-backward algorithm to compute the filtering and smoothing distributions and the Viterbi algorithm in order to find the optimal state path.

Question 5

Compute the filtering distribution $p(\Psi_n|x_{1:n})$ by using the forward recursion (fill part 4). How can you compute the marginals $p(s_n|x_{1:n})$, $p(c_n|x_{1:n})$, $p(m_n|x_{1:n})$?

Question 6

Compute the smoothing distribution $p(\Psi_{1:N}|x_{1:N})$ by using the forward-backward recursions (fill part 5). How can you compute the marginals $p(s_{1:N}|x_{1:N})$, $p(c_{1:N}|x_{1:N})$, $p(m_{1:N}|x_{1:N})$?

Question 7

Compute the most-likely path by using the Viterbi algorithm (fill part 6):

$$\Psi^{\star}_{1:N} = (s^{\star}_{1:N}, m^{\star}_{1:N}, c^{\star}_{1:N}) = \underset{\Psi_{1:N}}{\arg\max}\, p(\Psi_{1:N}|x_{1:N}). \tag{8}$$

By postprocessing this optimal path, decode the barcode string (fill part 7).

Question 8

From now on, generate random barcodes by using 'generate_barcode.m'. Run your algorithms for different random barcodes. Set $\mu_0 = 255$, $\mu_1 = 0$, and $\sigma_0^2 = \sigma_1^2 = 1$.

1. Visualize the (marginal) filtering distributions.
2. Visualize the (marginal) smoothing distributions.
3. Visualize the most-likely path.
4. Try different values for the variable 'obs_noise' in 'generate_barcode.m' and repeat the previous steps. How does the algorithm behave when you increase the value of 'obs_noise'?