Tutorial 8

CS 337 Artificial Intelligence & Machine Learning, Autumn 2021

October-November, 2021

1 Connecting Graphical Models to Propositional Logic

The Indian fruit Jambul[1] is often confused with Blackberries[2]. However, we will use certain regional and seasonal growing patterns of these two fruits along with their physical characteristics to help classify the fruit[3].

Some village folk tell us that the following clauses (together comprising Σ) are good enough for characterizing and discriminating between these fruits.

- blackberry $\leftrightarrow$ north
- blackberry $\leftrightarrow$ winter
- blackberry $\leftrightarrow$ autumn
- blackberry $\leftrightarrow$ medium_in_color
- blackberry $\leftrightarrow$ dark_in_color
- blackberry $\leftrightarrow$ small_in_size
- jambul $\leftrightarrow$ south
- jambul $\leftrightarrow$ spring
- jambul $\leftrightarrow$ summer
- jambul $\leftrightarrow$ light_in_color
- jambul $\leftrightarrow$ big_in_size

[1] `http://en.wikipedia.org/wiki/Jambul`

[2] `http://en.wikipedia.org/wiki/Blackberry`. It is the edible blackberry, not the non-edible smartphone :-)

[3] Note that the data we have is just cooked up.

However, feeling uncomfortable with the fact that blackberries do sometimes grow in the south, that blackberries are sometimes light in color, *etc.*, we decide to probabilistically model[4] the correlations between characteristics and fruits using an undirected[5] graphical model.

We thus start off with the following random variables: $\{S, F, L, C, Z\}$ where S stands for 'seasons' and takes four values, *viz.*, {winter, spring, summer, autumn}, F stands for the fruit and takes on the values {blackberry, jambul}, L stands for the location and assumes two values {north, south}. C stands for the color and takes values {light, medium, dark} while Z stands for the size and takes values {big, small}. We will abbreviate blackberry by b and jambul by j. We consider an undirected graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with the vertex set $\mathcal{V} = \{S, F, L, C, Z\}$ and edge set $\mathcal{E} = \{(S, F), (L, F), (C, F), (Z, F)\}$.

The potential functions over the maximal cliques (*i.e.*, edges) are given in the following tables:

S	$\phi(S, F = b)$	$\phi(S, F = j)$
winter	0.9	0.1
spring	0.3	0.7
summer	0.4	0.6
autumn	0.8	0.2

L	$\phi(L, F = b)$	$\phi(L, F = j)$
north	0.65	0.35
south	0.25	0.75

F	$\phi(C = light, F)$	$\phi(C = medium, F)$	$\phi(C = dark, F)$
b	0.33	0.33	0.34
j	0.8	0.1	0.1

F	$\phi(Z = big, F)$	$\phi(Z = small, F)$
b	0.4	0.6
j	0.95	0.05

- Suppose you are given a piece of fruit and you find that it is small and has medium color. What season is it now, most likely? What is your probability of being correct?
- As against this, what will be the result of (logical) inference from Σ given the evidences? Is there any difference between the results from logical and probabilistic inference? Explain.

2 Autocompletion and HMM

All of us know that Whatsapp and other messaging services have an autocomplete feature, wherein, as we type characters, the different possible word completions are prompted. In

[4]Obviously we needed to put more efforts and engaged a CS337 student in the modeling exercise.

[5]Since the implication in the clauses in Σ is a two-sided $\leftrightarrow$, an undirected graph makes more sense.

Figure 1, you can see the different autocomplete options for a message as I was typing on a whatsapp group involving several farmers.

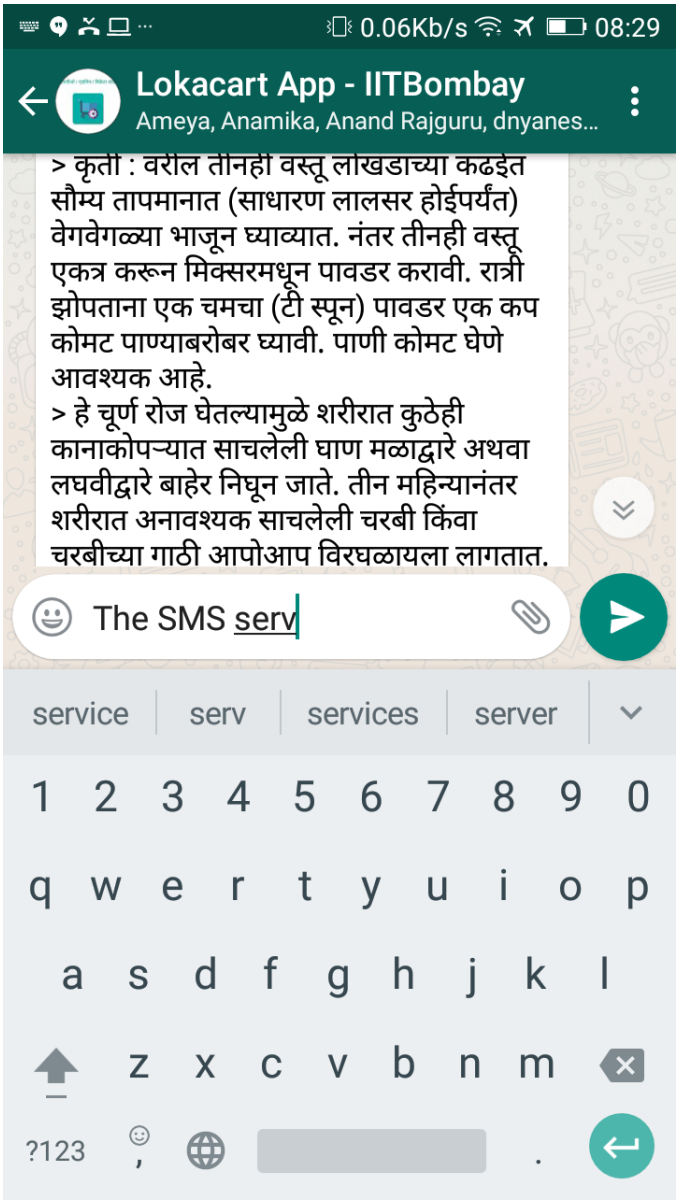

Figure 1: A figure showing the autocomplete feature in Whatsapp

Let us say that traditionally, this autocomplete was enabled by storing a prefix tree over all characters. Each node n_i in the tree represents a prefix (such as node labeled 'ser') and each directed edge e_iij represents the appending of a character to the prefix in n_i (such as appending 'v' to 'ser') to give rise to a longer prefix in n_j (such as 'serv'). We refer to n_j as an immediate **extension** of n_i. Thus, if the characters typed so far are 'serv', the autocompletion options correspond to all the word **extensions** that result from traversing down from the node n_j with prefix 'serv' to every leaf level node (such as 'service', 'services',

'server', etc.). In Figure (2) we have depicted only a small subtree in the larger prefix tree of all strings.

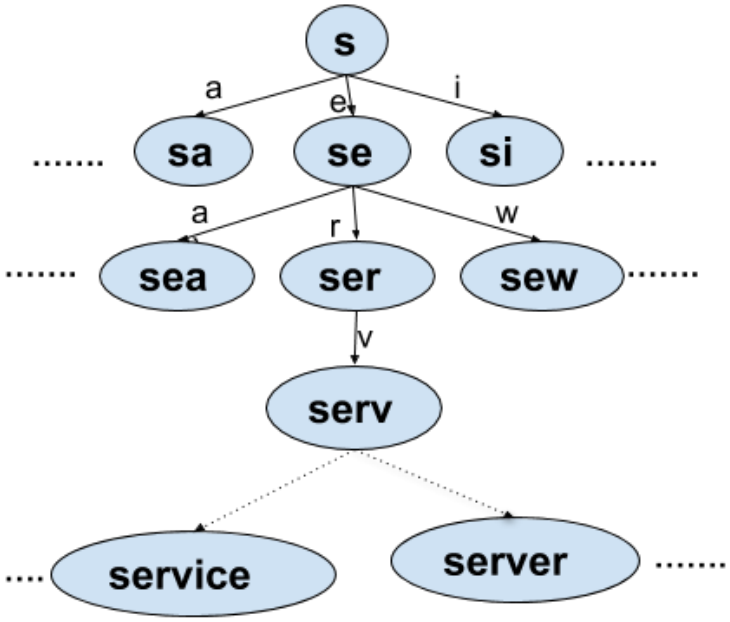

Figure 2: A subtree in the larger prefix tree of all strings

1. Suppose, that in all there are 90 characters that are ever seen on this farmer's Whatsapp group (these include the 26 characters in English and characters in Devanagari etc for also supporting Hindi), what is potentially the number of nodes in this prefix tree? A word can begin with any of the 90 characters. You can answer this question for the average case in which each word is of length 7, and each non-empty prefix has 3 possible immediate **extensions**.

   Solution: Starting from the root, each node has 3 nodes as children. Thus, adding the total number of nodes at each level:$90 \times (1 + 3^2 + ... + 3^6)$.

2. Now you, as our our smart CS337 student believes that prefix trees are outdated and need to be replaced with more relevant and contemporary Machine Learning models for this Autocompletion task. In particular, you need to design an HMM (Hidden Markov Model) for this task. You can assume the same statistics stated in part 1 and design the appropriate number of nodes/parameters in the HMM.

   (a) Describe your HMM and state the number of parameters you chose inspired by part 1 above.

   (b) Explain the correspondence between the nodes/edges in the prefix tree and the nodes/edges in the HMM.

   Solution: A possible answer is: One hot encoding based on 90 characters in the input. 3 nodes in a hidden layer corresponding to the average fanout of 3 mentioned in part

1. And one hot encoding based output which means 90 nodes in the output layer as well. This amounts to a total of 90×3(input to hidden) $+ 90 \times 3$(hidden to output) $+$ 3(last one for the self loop) $= 543$ parameters. There are exponentially many nodes in the tree of all character strings. In an HMM, each node is a hidden state vector. The next character must transform this to a new node.

   (a) If the nodes are implemented as hidden states, different nodes can share structure because they use distributed representations.

   (b) The next hidden representation needs to depend on the conjunction of the current character and the current hidden representation.

3. Discuss advantages and disadvantages of using prefix trees vs. your HMM.

   Solution:

   (a) Advantages of HMM/Disadvantages of prefix-tree:

      i. More compact representation.

      ii. Can be trained and even dynamically updated based on choices made by the user as she/he types and select auto-completions.

      iii. On the other hand, the prefix tree is a bit dumb and is independent of the statistics of the usage of different words. For example, even if 'servable' is a valid word, it is rarely used and the HMM can be eventually trained to reflect this knowledge. But the prefix tree method would treat 'servable' to be as likely an auto-completion option as service.

   (b) Disadvantages of HMM/Advantages of prefix-tree

      i. The prefix tree will be deterministic and fast to infer.

      ii. HMM will incur overheads in training (and also at test time - though students might not be able to guess this aspect). So we will need to decide how often to retrain the HMM model as we go on collecting auto-completion options from the user(s).

      iii. Also, we need to decide when the word will end. So perhaps an end of word character also needs to be included (making the size 91) and predicted.

      iv. Extra/Optional: As for inference, the HMM will need to keep track of all the best/most probably sequences starting from the current input. This will require beam search (exact optimization can be expensive).

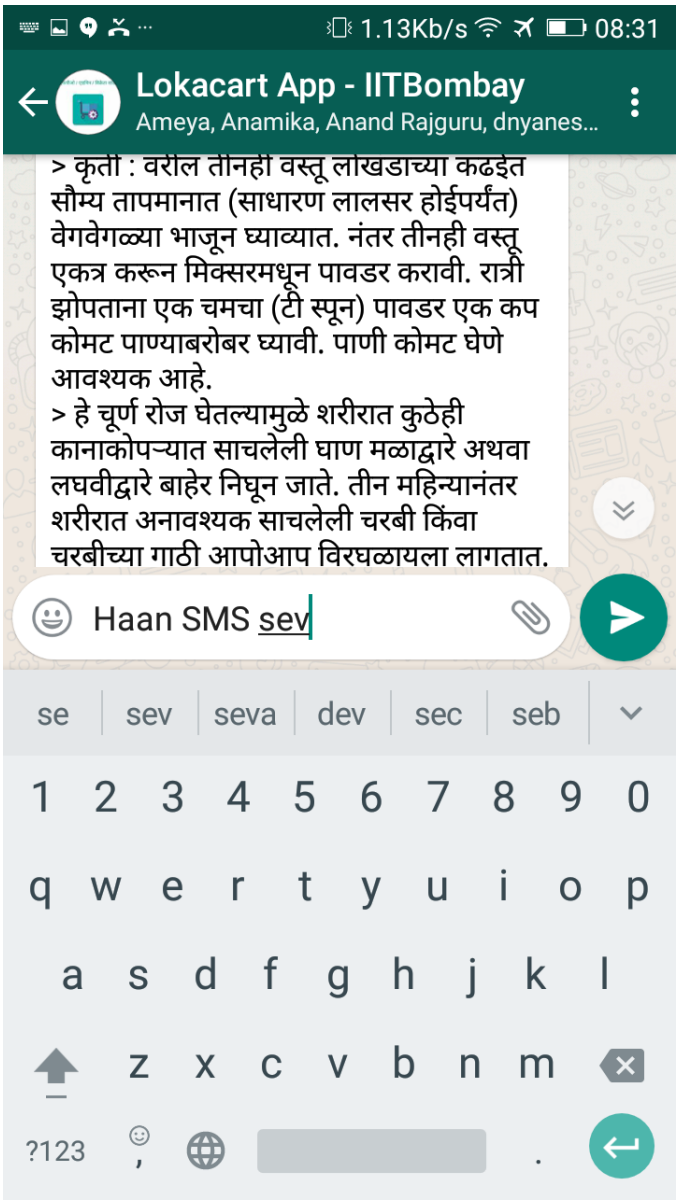

Figure 3: A figure showing the autocomplete feature adopted to Hindi words that might also be typed using the English (latin) alphabet

4. **Slightly Harder - Code mixing:** We realize that in our farmer's WhatsApp group, people often type Hindi words in latin (that is, using English alphabets). This is quite common in India and in (3) we illustrate how the autocomplete feature has been adopted to Hindi words that might also be typed using the English (latin) alphabet. Thus, while 'SMS' is a word purely from English, 'Haan' and 'seva' (as an auto-completion option) are Hindi words typed in English. This is often referred to as **code-mixing**

   (a) What would happen if we simply applied either the Prefix tree model in question

1 or your HMM model in question 2 in this setting? What are limitations?

Solution: In both cases, the vocabulary of English and the other language Hindi would be completely mixed up and that too in the same script. If the scripts were different, atleast language detection could have been enabled using the script. In the absence of that information, the suggestions would (i) dumbly be from a mixture of two different languages expressed in the same script and (ii) the user might get confused/surprised since she/he starts seeing suggestions from a language she/he is typing in!

(b) Our CS337 student is however determined and wants to extend the model from question 2 to also capture this new **code-mixing** phenomenon. Present an extension/modification of your HMM model in question 2 to help capture this phenomenon. Justify your choice and present the number of parameters your model has. Compare your new model with the model in part 2. *As an optional hint, you can draw inspiration from Principal Component Analysis (PCA) while proposing your extension.*

Solution:

i. One option is that you can simply add one more hidden layer with two nodes that connect directly to the 90 input nodes (corresponding to implicit language detection). So in this setting, the answer will be: One hot encoding based on 90 characters in the input. 2 nodes in the first hidden layer corresponding to the choice of the language. 3 nodes in the second hidden layer corresponding to the average fanout of 3 mentioned in part 1. And one hot encoding based output which means 90 nodes in the output layer as well. Also, the input nodes can also be (optionally) connected to the second hidden layer, since determining the code-switching might require access to more context. This amounts to a total of 90×2(input to first hidden) $+$ 2×3(first hidden to second hidden) $+ 90 \times 3$(input to second hideen) $+ 90 \times$ 3(second hidden to output) $+ 3 + 2$(last one for the self loops) $= 543 + 180 +$ $6 + 2 = 733$ parameters. i.e., 188 more parameters than the previous model. Other answers are also possible - for example using LSTMs.

ii. Another option is that one does PCA on the 90 dimensional one-hot encoded input as a **pre-processing step** and takes the smaller dimensional PCA output (say eigenvectors corresponding to the largest two eigenvalues) as input to the HMM. So the new HMM will have everything like the previous HMM in part 2 (a) but the number of input nodes will be different.

iii. Other options can also be entertained - such as use of LSTMS as long as the choice is **somewhat justified** and as long as the number of parameters is presented.

3 Noisy-or representation

Let $X_1, X_2, \ldots X_M$ be parents of Y in a bayesian network. Let $Y, X_1, \ldots, X_M \in \{0, 1\}$. The conditional probability table (CPT) for $p(Y|X_1, \ldots, X_M)$ should ordinarily consist of lot of

entries. How many are they?

Now, here is a representation of the same CPT using just M parameters! This representation, called the noisy-or representation, is as follows:

$$p(Y = 1|X_1, \ldots, X_M) = 1 - \prod_{i=1}^{M}(1 - \mu_i)^{X_i}$$

where the parameters μ_i $(i = 1, 2, \ldots, M)$ represent the probabilities $p(X_i = 1)$. Explain why this representation for the CPT for $p(Y|X_1, \ldots, X_M)$ might be called a 'noisy-or' representation.

4 Gaussian Discriminant Analysis

4.1 Quadratic separating surface

- Consider the following Gaussian Discriminant Classifier discussed in class. Let us say we have K classes with a multivariate Gaussian Model $\mathcal{N}(\mu_i, \Sigma_i)$ fitted for each class:

$$P(\phi(x)|C_1) = \mathcal{N}(\mu_1, \Sigma_1)$$

$$P(\phi(x)|C_i) = \mathcal{N}(\mu_i, \Sigma_i)$$

$$P(\phi(x)|C_K) = \mathcal{N}(\mu_K, \Sigma_i)$$

- Assumption: $\phi(x)$ is generated using **exactly one** $\mathcal{N}(\mu_i, \Sigma_i)$
- In the case of $K = 2$, with $P(C_1)$ and $P(C_2)$ known, separating surface will be $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$.

Prove that the separating surface will be **quadratic**.

SOLUTION:

- If $\phi(x) \sim \mathcal{N}(\mu_i, \Sigma_i)$ (where $\phi(x) \in \Re^m$) then

$$p(\phi(x) \mid C_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma_i|^{\frac{1}{2}}} exp\frac{-(\phi(x) - \mu_i)^T\Sigma_i^{-1}(\phi(x) - \mu_i)}{2}$$

- So, the separating surface is $\phi(x)$ such that $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\} \Longleftrightarrow \{\phi(x) \mid P(\phi(x) \mid C_1)P(C_1) = P(\phi(x) \mid C_2)P(C_2)\} \Longleftrightarrow$ after taking logs, $\phi(x)$ such that

$$-(\phi(x) - \mu_1)^T\Sigma_1^{-1}(\phi(x) - \mu_1) + (\phi(x) - \mu_2)^T\Sigma_2^{-1}(\phi(x) - \mu_2) = b$$

  where b contains terms independent of $\phi(x)$.

- This is indeed a **QUADRATIC equation**!

4.2 Linear Discriminant Analysis

- Now consider the following variant of the above Gaussian Discriminant Classifier. Let us say we have K classes with a multivariate Gaussian Model $\mathcal{N}(\mu_i, \Sigma)$ fitted for each class. That is, the covariance matrix Σ is now shared across the classes:

$$P(\phi(x)|C_1) = \mathcal{N}(\mu_1, \Sigma)$$

$$P(\phi(x)|C_i) = \mathcal{N}(\mu_i, \Sigma)$$

$$P(\phi(x)|C_K) = \mathcal{N}(\mu_K, \Sigma)$$

- Assumption: $\phi(x)$ is generated using **exactly one** $\mathcal{N}(\mu_i, \Sigma)$.

- As before, in the case of $K = 2$, with $P(C_1)$ and $P(C_2)$ known, separating surface will be $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$.

1. Q: Prove that the separating surface will now be **linear**.

   SOLUTION:

   - If $\phi(x) \sim \mathcal{N}(\mu_i, \Sigma)$ (where $\phi(x) \in \Re^m$) then

$$p(\phi(x) \mid C_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}} exp\frac{-(\phi(x) - \mu_i)^T\Sigma^{-1}(\phi(x) - \mu_i)}{2}$$

   - So, the separating surface is $\phi(x)$ such that $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$ $\Longleftrightarrow$ $\{\phi(x) \mid P(\phi(x) \mid C_1)P(C_1) = P(\phi(x) \mid C_2)P(C_2)\}$ $\Longleftrightarrow$ after taking logs, $\phi(x)$ such that

$$-(\phi(x) - \mu_1)^T\Sigma^{-1}(\phi(x) - \mu_1) + (\phi(x) - \mu_2)^T\Sigma^{-1}(\phi(x) - \mu_2) = b$$

   where b contains terms independent of $\phi(x)$
   $\Longleftrightarrow$ ...the above expression can actually be simplified by canceling out he terms involving $\phi^T(x)\Sigma^{-1}\phi(x)$

$$-\phi^T(x)\Sigma^{-1}\phi(x) \ldots - \phi^T(x)\Sigma^{-1}\phi(x) = b$$

   to finally give

$$2\phi^T(x)\Sigma^{-1}\mu_1 + 2\phi^T(x)\Sigma^{-1}\mu_2 - \mu_1^T(x)\Sigma^{-1}\mu_1 - \mu_2^T(x)\Sigma^{-1}\mu_2 = b$$

   that is,

$$2\phi^T(x)\Sigma^{-1}\mu_1 + 2\phi^T(x)\Sigma^{-1}\mu_2 - \mu_1^T(x)\Sigma^{-1}\mu_1 = b'$$

   which is a **LINEAR equation**! Here, $b' = \mu_1^T(x)\Sigma^{-1}\mu_1 + \mu_2^T(x)\Sigma^{-1}\mu_2 + b$ is independent of $\phi(x)$.

2. Q: What will be the maximum likelihood estimates for μ_i and Σ in this new case of different means but shared covariance matrix?

   SOLUTION: The Maximum Likelihood estimate for $\hat{\mu}_i$ will be the same as that for the Quadratic Discriminant Analysis, but that for a shared and single covariance estimate $\hat{\Sigma}$ will correspond to the average of covariance matrix estimates across examples from all the classes

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \phi(x_j^i)$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{K} \sum_{j=1}^{n_i} (\phi(x_i^j) - \mu_i)(\phi(x_i^j) - \mu_i)^T$$