Faculty of Computer Science, Dalhousie University *9-Oct-2024*

CSCI 4152/6509 — Natural Language Processing

Lecture 10: Basic Probabilistic Models; P0 Topics Discussion

Location: Carleton Tupper Building Theatre C Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lectures

- P0 Topics Discussion: P-01, P-03, P-04, P-05, P-06
- **Probabilistic approach to NLP**
- Logical vs. plausible reasoning
- Probability theory review
- Bayesian inference: generative models
- Probabilistic modeling:
  - random variables, random models
  - full and partial model configurations
  - computational tasks in probabilistic modeling

11.6 Joint Distribution Model

In the Joint Distribution Model, we specify the complete **joint probability distribution,** i.e., the probability of each complete configuration $\mathbf{x} = (x_1, ..., x_n)$:

$$\mathrm{P}(V_1 = x_1, ..., V_n = x_n)$$

In general, we need m^n parameters (minus one constraint) to specify an arbitrary joint distribution on n random variables with m values. One could represent this by a lookup table $p_{\mathbf{x}^{(1)}}, p_{\mathbf{x}^{(2)}}, \ldots, p_{\mathbf{x}^{(m^n)}}$, where $p_{\mathbf{x}^{(\ell)}}$ gives the probability that the random variables jointly take on configuration $\mathbf{x}^{(\ell)}$; that is, $p_{\mathbf{x}^{(\ell)}} = \mathrm{P}(\mathbf{V} = \mathbf{x}^{(\ell)})$. These numbers are positive and satisfy the constraint that $\sum_{\ell=1}^{m^n} p_{\mathbf{x}^{(\ell)}} = 1$.

Example: Spam Detection (continued)

To estimate the joint distribution in our spam detection example, we can simply divide the number of message for each configuration with the total number of messages:

Free	*Caps*	*Spam*	Number of messages	p
Y	Y	Y	20	0.20
Y	Y	N	1	0.01
Y	N	Y	5	0.05
Y	N	N	0	0.00
N	Y	Y	20	0.20
N	Y	N	3	0.03
N	N	Y	2	0.02
N	N	N	49	0.49
		Total:	100	1.00

Estimating probabilities in this way is known as *Maximum Likelihood Estimation* (MLE), since it can be shown that in this way the probability $\mathrm{P}(T|M)$, where T is our training data and M is the model, is maximized in terms of M.

Evaluation (task 1)

As defined earlier, the evaluation task is to evaluate the probability of a complete configuration $\mathbf{x} = (x_1, ..., x_n)$. In the case of joint distribution model, we simply use a table lookup operation:

$$\mathrm{P}(V_1 = x_1, ..., V_n = x_n) = p_{(x_1, x_2, ..., x_n)}$$

Using the spam example, an instance of evaluation task is:

$$\mathrm{P}(\textit{Free} = Y, \textit{Caps} = N, \textit{Spam} = N) = 0.00$$

If we choose some other configuration, we will get a positive probability. This particular configuration has the probability zero due to the fact that it was not seen in the training data, so our estimate based on simple counting is 0. This is a drawback of this model, since the number of possible configurations is typically very large and it is very likely that the training data will not contain some configurations, although any configuration is actually possible. This example is chosen on purpose to show this drawback of the full joint distribution model called the **sparse data problem**.

Simulation (task 2)

Simulation is performed by randomly selecting a full configuration according to the probability distribution in the table. This can be done by dividing the interval $[0, 1)$ into subintervals, whose lengths are p_1, p_2, ..., and p_{m^n}. In most programming languages, there is a random number generator function (a pseudo-random number, to be more precise), which generates random numbers from the interval $[0, 1)$ according to the uniform probability distribution. Based on which interval this random number falls in, we choose the full configuration to generate. This method is known as the "roulette wheel" method, since it can also be represented using a rotating unit disk divided into cut-out segments (like pizza slices) of areas proportional to the table probabilities, and the generation can be visualized as rotating the disk until it randomly stops, while a fixed pointer is used to select a segment. In more details, the following steps can be followed to program this generating procedure:

1. Divide the interval $[0, 1]$ into subintervals of the lengths: $p_1, p_2, \ldots, p_{m^n}$: $I_1 = [0, p_1)$, $I_2 = [p_1, p_1 + p_2)$, $I_3 = [p_1 + p_2, p_1 + p_2 + p_3), \ldots I_{m^n} = [p_1 + p_2 + \ldots + p_{m^n - 1}, 1)$
2. Generate a random number r from the interval $[0, 1)$
3. r will fall exactly into one of the above intervals, e.g.: $I_i = [p_1 + \ldots + p_{i-1}, p_1 + \ldots + p_{i-1} + p_i)$
4. Generate the configuration number i from the table
5. Repeat steps 2–4 for as many times as the number of configurations we need to generate

Inference (task 3)

Marginalization (3.a). The task of marginalization is computing a marginal probability; i.e., the probability of a partial configuration, such as $\mathrm{P}(X_1 = x_1, ..., X_k = x_k)$, where $k < n$:

$$\begin{aligned}
&\mathrm{P}(V_1 = x_1, ..., V_k = x_k) \\
&= \sum_{y_{k+1}} \cdots \sum_{y_n} \mathrm{P}(V_1 = x_1, ..., V_k = x_k, V_{k+1} = y_{k+1}, ..., V_n = y_n) \\
&= \sum_{y_{k+1}} \cdots \sum_{y_n} p_{(x_1, ..., x_k, y_{k+1}, ..., y_n)}
\end{aligned}$$

We need to be able to evaluate complete configurations and then sum over m^{n-k} possible completions, where m is the number of elements in the domain of $y_{k+1}, \ldots, y_n$. This can be implemented by iterating through the model

table and accumulating all probabilities that correspond to the matching configurations; i.e., all full configurations that satisfy the assignments given by the partial configuration for which we calculate the probability.

Conditioning (3.b). Conditioning is the task of computing a conditional probability in the form of probability of assignments of some variables given the assignments of other variables. This probability can be calculated as:

$$\begin{aligned} & \mathrm{P}(V_1 = x_1, \ldots, V_k = x_k | V_{k+1} = y_1, ..., V_{k+l} = y_l) \\ = \quad & \frac{\mathrm{P}(V_1 = x_1, \ldots, V_k = x_k, V_{k+1} = y_1, ..., V_{k+l} = y_l)}{\mathrm{P}(V_{k+1} = y_1, ..., V_{k+l} = y_l)} \end{aligned}$$

so we see that it is reduced to two marginalization tasks. If the configuration in the numerator happens to be a full configuration, that the task is even easier and reduces to one evaluation and one marginalization.

Completion (3.c). Completion is the task of finding the most probable completion $(y^*_{k+1}, ..., y^*_n)$ of a partial configuration $(x_1, ..., x_k)$.

$$\begin{aligned} y^*_{k+1}, ..., y^*_n &= \underset{y_{k+1},\ldots,y_n}{\arg\max}\ \mathrm{P}(V_{k+1} = y_{k+1}, ..., V_n = y_n | V_1 = x_1, ..., V_k = x_k) \\ &= \underset{y_{k+1},\ldots,y_n}{\arg\max}\ \frac{\mathrm{P}(V_1 = x_1, ..., V_k = x_k, V_{k+1} = y_{k+1}, ..., V_n = y_n)}{\mathrm{P}(V_1 = x_1, ..., V_k = x_k)} \\ &= \underset{y_{k+1},\ldots,y_n}{\arg\max}\ \mathrm{P}(V_1 = x_1, \ldots, V_k = x_k, V_{k+1} = y_{k+1}, ..., V_n = y_n) \\ &= \underset{y_{k+1},\ldots,y_n}{\arg\max}\ p_{(x_1,\ldots,x_k,y_{k+1},\ldots,y_n)} \end{aligned}$$

We can implement this by iterating through the model table, and from all configurations match the assignments in the given partial configuration find one with the maximal probability.

Learning (task 4)

Learning is the task of estimating the model parameters based on the given data. We use the **Maximum Likelihood Estimation** (MLE), mentioned before; i.e., for each full configuration we count the number of times this configuration occurred in the data, and divide this number by the total number of the full configurations in the data. This can be expressed using the following formula:

$$p_{(x_1,\ldots,x_n)} = \frac{\#(V_1 = x_1, \ldots, V_n = x_n)}{\#(*, \ldots, *)}$$

We use the hash or number symbol ('#') to denote the number of occurrences of a pattern in a dataset. In the above example, $\#(x_1, \ldots, x_n)$ denotes the number of full configurations $(x_1, \ldots, x_n)$ in the give dataset, and the expression $\#(*, \ldots, *)$ denotes the number of all configurations in the given dataset.

With a large number of variables the data size easily becomes insufficient and we get many zero probabilities — **sparse data problem**

Drawbacks of Joint Distribution Model

- memory cost to store table,
- running-time cost to do summations, and
- the sparse data problem in learning (i.e., training).

Other probability models are found by specifying specialized joint distributions, which satisfy certain independence assumptions.

The goal is to impose structure on joint distribution $\mathrm{P}(V_1 = x_1, ..., V_n = x_n)$. One key tool for imposing structure is variable independence.

11.7 Fully Independent Model

In a fully independent model we assume that all variables are independent, i.e.,

$$\mathrm{P}(V_1 = x_1, ..., V_n = x_n) = \mathrm{P}(V_1 = x_1) \cdots \mathrm{P}(V_n = x_n).$$

It is an efficient model with a small number of parameters: $O(nm)$, where n is the number of variables and m is the number of distinct values of the variables.

The drawback of the model is that the independence assumption is too strong for the model to be useful in any applications.

Fully Independent Model for the Spam Example

If we apply the fully independent model to the spam example, we obtain the following assumption formula:

$$\mathrm{P}(\textit{Free}, \textit{Caps}, \textit{Spam}) = \mathrm{P}(\textit{Free}) \cdot \mathrm{P}(\textit{Caps}) \cdot \mathrm{P}(\textit{Spam})$$

This yields a very restricted form of joint distribution where we can represent each component distribution separately. For a random variable V_j, one can represent $\mathrm{P}(V_j = x)$ by a lookup table with m parameters (minus one constraint). Let $p_{j,x}$ denote the probability V_j takes on value x. That is, $p_{j,x} = \mathrm{P}(V_j = x)$. These numbers are positive and satisfy the constraint $\sum_{x=1}^{m} p_{j,x} = 1$ for each j. Thus, the joint distribution over $V_1, ..., V_n$ can be represented by $n \times m$ positive numbers minus n constraints. The previous tasks (simulation, evaluation, and inference) now become almost trivial. Admittedly this is a silly model as far as real applications go, but it clearly demonstrates the benefits of structure (in its most extreme form).

Example: Spam Detection (continued)

The fully independent model is almost useless in our spam detection example because it assumes that the three random variables: *Caps, Free,* and *Spam* are independent. In other words, its assumption is that knowing whether a message has a capitalized subject or contains the word 'Free' in the subject cannot help us in determining whether the message is spam or not, which is not in accordance with our earlier assumption.

Anyway, let us see what happens when we apply the fully independent model to our example. From the training data:

Free	*Caps*	*Spam*	Number of messages
Y	Y	Y	20
Y	Y	N	1
Y	N	Y	5
Y	N	N	0
N	Y	Y	20
N	Y	N	3
N	N	Y	2
N	N	N	49
		Total:	100

we generate the following probability tables of independent variables:

Free	P(*Free*)
Y	$\frac{20+1+5+0}{100} = 0.26$
N	$\frac{20+3+2+49}{100} = 0.74$

and similarly,

Caps	P(*Caps*)
Y	$\frac{20+1+20+3}{100} = 0.44$
N	$\frac{5+0+2+49}{100} = 0.56$

and

Spam	P(*Spam*)
Y	$\frac{20+5+20+2}{100} = 0.47$
N	$\frac{1+0+3+49}{100} = 0.53$

Hence, in this model any message is a spam with probability 0.47, no matter what the values of *Caps* and *Free* are.

This is example of MLE **Learning** (computational task 4.).

As an example of evaluation, the probability of configuration $(\textit{Caps} = Y, \textit{Free} = N, \textit{Spam} = N)$ in the fully independent model is:

$$\begin{aligned}
&\mathrm{P}(\textit{Free} = Y, \textit{Caps} = N, \textit{Spam} = N) = \\
&\quad = \quad \mathrm{P}(\textit{Free} = Y) \cdot \mathrm{P}(\textit{Caps} = N) \cdot \mathrm{P}(\textit{Spam} = N) = 0.26 \cdot 0.56 \cdot 0.53 \\
&\quad = \quad 0.077168 \approx 0.08
\end{aligned}$$

2. Simulation (Fully Independent Model)

For $j = 1, ..., n$, independently draw x_j according to $\mathrm{P}(V_j = x_j)$ (using the lookup table representation). Conjoin $(x_1, ..., x_n)$ to form a complete configuration.

3. Inference in Fully Independent Model

3.a Marginalization in Fully Independent Model

The probability of a partial configuration $(V_1 = x_1, \ldots, V_k = x_k)$ is

$$P(V_1 = x_1, \ldots, V_k = x_k) = P(V_1 = x_1) \cdot \ldots \cdot P(V_k = x_k)$$

This formula can be obvious, but it can also be derived.

Derivation of Marginalization Formula

$$\begin{aligned}
&\mathrm{P}(V_1 = x_1, ..., V_k = x_k) = \sum_{y_{k+1}} \cdots \sum_{y_n} \mathrm{P}(V_1 = x_1, ..., V_k = x_k, V_{k+1} = y_{k+1}, ..., V_n = y_n) \\
&\quad = \quad \sum_{y_{k+1}} \cdots \sum_{y_n} \mathrm{P}(V_1 = x_1) \cdots \mathrm{P}(V_k = x_k) \mathrm{P}(V_{k+1} = y_{k+1}) \cdots \mathrm{P}(V_n = y_n) \\
&\quad = \quad \mathrm{P}(V_1 = x_1) \cdots \mathrm{P}(V_k = x_k) \left[\sum_{y_{k+1}} \mathrm{P}(V_{k+1} = y_{k+1}) \left[\sum_{y_{k+2}} \cdots \left[\sum_{y_n} \mathrm{P}(V_n = y_n) \right] \right] \right] \\
&\quad = \quad \mathrm{P}(V_1 = x_1) \cdots \mathrm{P}(V_k = x_k) \left[\sum_{y_{k+1}} \mathrm{P}(V_{k+1} = y_{k+1}) \right] \cdots \left[\sum_{y_n} \mathrm{P}(V_n = y_n) \right] \\
&\quad = \quad \mathrm{P}(V_1 = x_1) \cdots \mathrm{P}(V_k = x_k)
\end{aligned}$$

Only have to lookup and multiply k numbers.

Note

It is important to note a general rule which we used to separate summations in the above tasks of Marginalization and Completion: If a and b are two variables, and $f(a)$ and $g(b)$ are two functions, such that $f(a)$ does not depend on b and $g(b)$ does not depend on a, then:

$$\begin{aligned}
\sum_a \sum_b f(a)g(b) &= \sum_a f(a) \left(\sum_b g(b) \right) \\
&\quad \text{(because } f(a) \text{ is a constant for summation over } b\text{)} \\
&= \left(\sum_b g(b) \right) \cdot \left(\sum_a f(a) \right) \\
&\quad \text{(because } \sum_b g(b) \text{ is a constant for sumation over } a\text{)} \\
&= \left(\sum_a f(a) \right) \cdot \left(\sum_b g(b) \right)
\end{aligned}$$

If we assume that $f(a) \geq 0$ and $g(b) \geq 0$, the same rule applies for $\max_a$ and $\max_b$:

$$\begin{aligned}
&\max_a \max_b f(a)g(b) = \\
&= \max_a f(a) \left(\max_b g(b) \right) \\
&\quad \text{(because } f(a) \text{ is a constant for maximization over } b\text{)} \\
&= \left(\max_b g(b) \right) \cdot \left(\max_a f(a) \right) \\
&\quad \text{(because } \max_b g(b) \text{ is a constant for maximization over } a\text{)} \\
&= \left(\max_a f(a) \right) \cdot \left(\max_b g(b) \right)
\end{aligned}$$

P0 Topics Discussion (2)

- Discussion of individual projects as proposed in P0 submissions
- Projects discussed: P-02