Faculty of Computer Science, Dalhousie University *23-Oct-2025*

CSCI 4152/6509 — Natural Language Processing

Lecture 9: Introduction to Probabilistic Modeling

Location: Studley LSC-Psychology P5260 Instructor: Vlado Keselj
Time: 14:35 – 15:55

Previous Lecture

- Evaluation methods for Text Classification:
  - underfitting and overfitting
  - training error, train and test, n-fold cross-validation
- Text Clustering
- Discussion about evaluation methods for classifiers
- Similarity-based Text Classification
- CNG classification method
- Edit distance:
  - introduction, properties, dynamic programming approach, example, algorithm

Part III

Probabilistic Approach to NLP

11 Introduction to Probabilistic Approach to NLP

11.1 Logical versus Plausible Reasoning

Artificial Intelligence (AI) is an area of Computer Science related to the task of developing software and computers that exhibit some form of intelligent behaviour. NLP can be considered to be a sub-area of AI. Solving problems in AI generally involves implementing in a computer some form of reasoning or inference. The automated inference methods can be divided into two large groups: *logical reasoning*, and *plausible reasoning.* As we will see, this division can be adopted for the NLP area as well.

1. Logical reasoning is known also as classical, symbolic, or knowledge-based AI. The rule-based AI is also a form of logical reasoning. This form of automated inference is based on the principles that we can find in mathematical logic. We start with basic set of premises, known as axioms, and using a fixed set of rules, we derive new conclusions. This type of reasoning is called *monotonic* since with more evidence, i.e., more known facts, we can produce more conclusions. A consequence of this is that once we make conclusions, they cannot be retracted or canceled. In other words, if we conclude that something is true, we cannot conclude that it is wrong based on the new evidence because that would mean that we either had some wrong evidence in the first place, or we made a mistake in reasoning. We also describe this reasoning as *certain*, since the model is designed in such way that we can derive only certain (or definite, guaranteed) conclusions.

2. Plausible reasoning is an automated reasoning in which we typically derive plausible conclusions; i.e., conclusions that may be true, and we usually have some way of rating the truthfulness of such conclusions. There are different approaches to this kind of automated reasoning. The most widely used are the approaches based on the mathematical probability theory, where we try to make a mathematically sound estimate of the probability that our conclusions are true. Other approaches include fuzzy logic and neural networks. Unlike logical reasoning, plausible reasoning is *uncertain*, since the conclusions are not guaranteed to be correct. Logical reasoning is also *non-monotonic* since based on new evidence, we can withdraw some conclusions, and with more evidence we may end up with less conclusions. Plausible reasoning allows for contradicting evidence.

Plausible Reasoning

- How to combine ambiguous, incomplete, and contradicting evidence to draw reasonable conclusions?
- Frequently approached as the task of making plausible inference of some hidden structure from observations.
- examples:

Input (observations)		Hidden Structure
symptoms	$\rightarrow$	illness
pixel matrix	$\rightarrow$	object, relations
speech signal	$\rightarrow$	phonemes, words
word sequence	$\rightarrow$	meaning
sentence	$\rightarrow$	parse tree
word sequence	$\rightarrow$	POS tags, names, entities
words in e-mail Subject:	$\rightarrow$	Is message spam? Yes/No
text	$\rightarrow$	text category (class)

Probabilistic NLP as a Plausible Reasoning Approach

- Regular expressions and finite automata are example of logical or knowledge-based approach to NLP
- Plausible approaches to NLP:
    1. Probabilistic: use of Theory of Probability, also known as stochastic or statistical NLP
    - Alternative plausible approaches, examples:
    2. neural networks,
    3. kernel methods,
    4. fuzzy logic, fuzzy sets,
    5. Dempster-Shafer theory
    6. rough sets,
    7. default logic, ...

11.2 Review of Basics of Probability Theory

This is a brief and intuitive review of some basic notions from the theory of probability.

- You should have this background from previous courses; this is just a review,
    - discussed a bit in the textbook: [JM] 5.5, and [MS] 2.1
- Simple event or basic outcome
    - e.g., rolling a die, choosing a letter
- *Event space:* the set of all outcomes, usually denoted Ω

  The event space is also known as the sample space. For example, the event space for rolling a die has six elements, corresponding to the six different outcomes; or an event space for choosing a letter of the English alphabet has 26 elements if we ignore letter case.
- *Event or outcome* is a set of simple events or basic outcomes
- In other words event is any subset of Ω; i.e., $A \subseteq \Omega$
- Each event is associated with a probability, which is a number between 0 and 1, inclusive: $0 \leq \mathrm{P}(A) \leq 1$

Probability Examples

- P(“rolling a 6 with a die”) $= 1/6$
- Choosing a letter of English alphabet:
  - If we choose uniformly: $\mathrm{P}(\text{‘a’}) = 1/26 \approx 0.04$
  - Choosing from a text: $\mathrm{P}(\text{‘a’}) \approx 0.08$
  - Remember our output from “Tom Sawyer”:

```
 35697 0.1204 e
 28897 0.0974 t
 23528 0.0793 a
 23264 0.0784 o
 20200 0.0681 n
...
```

As we saw from the “Tom Sawyer” example, the probability of the letter ‘a’ in a typical English text is about 0.08.

The probability can be seen as a function that maps a set of possible experiment outcomes to a real number between 0 and 1, including 0 and 1, i.e. to a number from the interval $[0, 1]$. We always have in mind certain space of outcomes Ω and we assume that in each experiment (trial, instance, or model configuration), one of those outcomes will happen. The probability that one of the outcomes from a set A ($A \subset \Omega$) will happen, is denoted $\mathrm{P}(A)$. A set of outcomes A is called an event.

Probability Axioms

- Probability axioms: nonnegativity, additivity, normalization:

  The basic properties of the probability, known as the probability axioms, are:
  - **(Nonnegativity)** $\mathrm{P}(A) \geq 0$, for any event A
  - **(Additivity)** for disjoint events A and B, i.e., if $A, B \subset \Omega$ and $A \cap B = \emptyset$, then
    $$\mathrm{P}(A \cup B) = \mathrm{P}(A) + \mathrm{P}(B).$$
    More generally, for a possibly infinite sequence of disjoint events A_1, A_2, $\ldots$,
    $$\mathrm{P}(A_1 \cup A_2 \cup \ldots) = \mathrm{P}(A_1) + \mathrm{P}(A_2) + \ldots$$
  - **(Normalization)** $\mathrm{P}(\Omega) = 1$, where Ω is the entire sample space.
- Some consequences of the above axioms are: $\mathrm{P}(\emptyset) = 0$ and $\mathrm{P}(\Omega - A) = 1 - \mathrm{P}(A)$

Independent and Dependent Events

- Independent events A and B (definition): $\mathrm{P}(A, B) = \mathrm{P}(A) \cdot \mathrm{P}(B)$
- Use of comma in: $\mathrm{P}(A, B) = \mathrm{P}(A \cap B)$
- Example: choosing two letters in text
  1. Choosing independently: $\mathrm{P}(\text{‘t’}) = 0.1, \mathrm{P}(\text{‘h’}) = 0.07, \mathrm{P}(\text{‘t’}, \text{‘h’}) = 0.007$
  2. Choosing two consecutive letters (dependent events): $\mathrm{P}(\text{‘t’}, \text{‘h’}) = 0.04$

Conditional Probability

- Conditional probability
  $$\mathrm{P}(A|B) = \frac{\mathrm{P}(A, B)}{\mathrm{P}(B)}$$
  The probability $\mathrm{P}(A|B)$ (it is read “probability of A given B”) is the probability of event A happening given that the event B happens.

- Expressing independency using conditional probability
  Two events A are B are independent if and only if:

$$\mathrm{P}(A|B) = \mathrm{P}(A)$$

  This is an alternative definition of independent events.

Annotation with More Events

- There is a bit of flexibility in using notation; e.g.,
- $\mathrm{P}(A, B, C) = \mathrm{P}(A \cap B \cap C)$
- $\mathrm{P}(A|B, C) = \mathrm{P}(A|B \cap C)$
- $\mathrm{P}(A, B, C|D, E, F) = \mathrm{P}(A \cap B \cap C|D \cap E \cap F)$
- and so on.
- Three independent events: $\mathrm{P}(A, B, C) = \mathrm{P}(A)\mathrm{P}(B)\mathrm{P}(C)$
- Conditionally independent events

$$\mathrm{P}(A, B|C) = \mathrm{P}(A|C) \cdot \mathrm{P}(B|C)$$

Bayes' Theorem

- Bayes' theorem (one form):

$$\mathrm{P}(A|B) = \frac{\mathrm{P}(B|A) \cdot \mathrm{P}(A)}{\mathrm{P}(B)}$$

- The second form is based on breaking the set B into disjoint sets $B = A_1 \cup A_2 \cup \ldots$:

$$\mathrm{P}(A_i|B) = \frac{\mathrm{P}(B|A_i) \cdot \mathrm{P}(A_i)}{\mathrm{P}(B)} = \frac{\mathrm{P}(B|A_i) \cdot \mathrm{P}(A_i)}{\sum_i \mathrm{P}(B|A_i)\mathrm{P}(A_i)}$$

11.3 Bayesian Inference and Generative Models

In our further discussion, we will use *Bayesian Inference* applied to the *Generative Models.* These models are called the generative models because they describe how a particular event that we are studying is generated; i.e., what were the causal relations that were involved in producing certain obeservations. We can use the following visual representation to describe this model in general:

$$\begin{tikzcd} {} \arrow[d] & \text{world} \arrow[d] \\ \boxed{\text{truth}} \arrow[d] & \text{sensor} \arrow[d] \\ \boxed{\text{evidence}} & \text{observed measurement} \end{tikzcd}$$

We assume that there is certain true structure or information that caused some events, which can be observed through some sensors. We will simply refer to this true information as *truth* and information that is available to us as *observable evidence,* or simply as *evidence.*

Notation Remark: 'max' and 'arg max' Operators. Before presenting the main use of the Bayes theorem in Bayesian inference, let us first clarify the use of '$\max$' and 'arg max' operators in formulae.

The operator '$\max$' is applied to an expression depending on an index variable x (denoted $\max_x$) and it returns the maximum value of the expression over all possible values of the variable x. On the other hand, the operator 'arg max' gives one value of the index variable x (denoted arg $\max_x$) for which the expression achieves the maximum value. Figure 3 illustrates graphically these operators for a function $y = f(x)$.

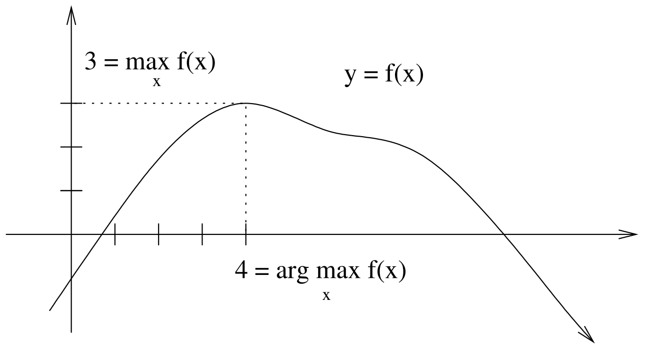

Figure 3: Illustrating operators $\max$ and arg max

Bayesian Inference: Using Bayes' Theorem

– Bayesian inference is a principle of combining evidence

$$\begin{aligned}\text{conclusion} &= \operatorname*{arg\,max}_{\text{possible truth}} P(\text{possible truth}|\text{evidence}) \\ &= \operatorname*{arg\,max}_{\text{possible truth}} \frac{P(\text{evidence}|\text{possible truth})P(\text{possible truth})}{P(\text{evidence})} \\ &= \operatorname*{arg\,max}_{\text{possible truth}} P(\text{evidence}|\text{possible truth})P(\text{possible truth})\end{aligned}$$

– application to speech recognition: acoustic model and language model

The above equations describe Bayesian inference. To understand steps in the equations, let us go over them step by step:

$$\text{conclusion} = \operatorname*{arg\,max}_{\text{possible truth}} P(\text{possible truth}|\text{evidence})$$

The above equation states that we reach the conclusion about the best possible truth by finding the maximal probability of that 'truth' given the evidence, according to our model.

$$\begin{aligned}\text{conclusion} &= \operatorname*{arg\,max}_{\text{possible truth}} P(\text{possible truth}|\text{evidence}) \\ &\boxed{=} \operatorname*{arg\,max}_{\text{possible truth}} \frac{P(\text{evidence}|\text{possible truth})P(\text{possible truth})}{P(\text{evidence})}\end{aligned}$$

In the above equation, we directly apply the Bayes' theorem.

$$\begin{aligned}\text{conclusion} &= \operatorname*{arg\,max}_{\text{possible truth}} P(\text{possible truth}|\text{evidence}) \\ &= \operatorname*{arg\,max}_{\text{possible truth}} \frac{\mathrm{P}(\text{evidence}|\text{possible truth})\mathrm{P}(\text{possible truth})}{\mathrm{P}(\text{evidence})} \\ &\boxed{=} \operatorname*{arg\,max}_{\text{possible truth}} \mathrm{P}(\text{evidence}|\text{possible truth})\mathrm{P}(\text{possible truth})\end{aligned}$$

It is important to note that the reason why the above equation holds is because the denominator $\mathrm{P}(\text{evidence})$ does not depend on 'possible truth'; i.e., it is constant for different values of 'possible truth'. A frequent mistake is to assume that the equation is true because the $\mathrm{P}(\text{possible truth})$ is 1.

If you are not clear why 'evidence' does not depend on different 'possible truths', it will be more clear once we express this in terms of random variables.

Bayesian Inference: Speech Recognition Example

Let us look at the speech recognition as an example of the problem that can be addressed using Bayesian Inference on a generative model.

Slide notes:

Bayesian Inference: Speech Recognition Example

- evidence $\rightarrow$ sound
- possible truth $\rightarrow$ utterance (words spoken)
- our best guess about utterance $\rightarrow$ utterance*

$$\begin{aligned}\text{utterance*} &= \underset{\text{all utterances}}{\text{arg max}}\ P(\text{utterance}|\text{sound}) \\ &= \underset{\text{all utterances}}{\text{arg max}}\ \frac{P(\text{sound}|\text{utterance})P(\text{utterance})}{P(\text{sound})} \\ &= \underset{\text{utterance}}{\text{arg max}}\ P(\text{sound}|\text{utterance})P(\text{utterance})\end{aligned}$$

At the end, we need to estimate two probabilities to recognize the utterance base on the sound: $\mathrm{P}(\text{sound}|\text{utterance})$ which is estimated using an *acoustic model*, and $\mathrm{P}(\text{utterance})$ which is estimated using a *language model.*

11.4 Probabilistic Modeling

Slide notes:

Probabilistic Modeling

- How do we create and use a probabilistic model?
- Model elements:
  - Random variables
  - Model configuration (Random configuration)
  - Variable dependencies
  - Model parameters
- Computational tasks

Random Variables

Slide notes:

Random Variables

- Random variable V, defining an event as $V = x$ for some value x from a domain of values D; i.e., $x \in D$
- $V = x$ is usually not a **basic** event due to having more variables
- An event with two random variables: $V_1 = x_1, V_2 = x_2$
- Multiple random variables: $\mathbf{V} = (V_1, V_2, ..., V_n)$

It is frequently convenient to define events as $V = x$, where V is a variable and x is a value from a domain of values for that variable. This event can be a basic event, but usually it is not since we may have more variables.

Slide notes:

Model Configuration (Random Configuration)

- **Full Configuration:** If a model has n random variables, then a Full Model Configuration is an assignment of all the variables:

$$V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n$$

- **Partial configuration:** only some variables are assigned, e.g.:

$$V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k \quad (k < n)$$

Random Configuration. A **probabilistic model** includes a set of random variables with their domains, i.e., sets of the possible values of those variables. If all variables are assigned some values, we will call it a **random configuration** of the model, or a **full random configuration**. For example, if V_1, V_2, …, V_n are all random variables of a model, then an assignment of those variables, such as:

$$V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n$$

is a **full random configuration**. For each such configuration, the model provides a way of calculating its probability; i.e., calculating the value

$$\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n).$$

From the probability theory perspective, a model defines an event space, where each full random configuration is a simple event.

If we assume that random variables are ordered in a certain way, we do not have to list variable names but just the values, so each full random configuration becomes an n-tuple of values $(x_1, x_2, \ldots, x_n)$, which we also call a **vector**, and use notation $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Vectors are normally n-tuples of numbers, but we generalize this concept here. When creating a probabilistic model for a problem, we assume that a sequence of full configurations $\mathbf{x}^{(1)}$, ..., $\mathbf{x}^{(t)}$ is drawn from some random source:

$$\begin{array}{rcllll}
\mathbf{x}^{(1)} & = & (x_{11}, & x_{12}, & \ldots & x_{1n}) \\
\mathbf{x}^{(2)} & = & (x_{21}, & x_{22}, & \ldots & x_{2n}) \\
 & \vdots & & & & \\
\mathbf{x}^{(t)} & = & (x_{t1}, & x_{t2}, & \ldots & x_{tn}) \\
 & \vdots & & & &
\end{array}$$

Again, we assume a fixed number n of components in each configuration, and assume values x_{ij} are from a finite set of values that can be assigned to the variable V_j. For simplicity reasons, we will frequently assume that all variables get values from the same set $\{x_1, x_2, \ldots, x_m\}$.

If only some variables as assigned, we will call such assignment a **partial configuration**. For example, if only variables V_1, V_2, …, V_k, where $k < n$ are assigned, then we have a partial configuration: $V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k$. In probabilistic terms, a partial configuration defines a random event that consists of all full configurations that satisfy the partial configuration assignment. This event can be described as the following set of full configurations:

$$\{\, (V_1 = y_1, V_2 = y_2, \ldots, V_n = y_n) \quad | \quad y_1 = x_1 \land y_2 = x_2 \land \ldots \land y_k = x_k \,\}$$

or, it is a set of all full configurations $(V_1 = y_1, V_2 = y_2, \ldots, V_n = y_n)$ such that $y_1 = x_1$ and $y_2 = x_2$ and … and $y_k = x_k$.

Probabilistic modelling in NLP can be described as a general framework for modeling NLP problems using random variables, random configurations, and finding effective ways of reasoning about probabilities of these configurations.

Variable Independence and Dependence

- Random variables V_1 and V_2 are *independent* if $\mathrm{P}(V_1{=}x_1, V_2{=}x_2) = \mathrm{P}(V_1{=}x_1)\mathrm{P}(V_2{=}x_2)$ for all x_1, x_2
- or expressed in a different way: $\mathrm{P}(V_1{=}x_1|V_2{=}x_2) = \mathrm{P}(V_1{=}x_1)$ for all x_1, x_2, x_3.
- Random variables V_1 and V_2 are *conditionally independent given* V_3 if, for all x_1, x_2, x_3:
  $\mathrm{P}(V_1{=}x_1, V_2{=}x_2|V_3{=}x_3) =$
  $\mathrm{P}(V_1{=}x_1|V_3{=}x_3)\mathrm{P}(V_2{=}x_2|V_3{=}x_3)$
- or
  $\mathrm{P}(V_1{=}x_1|V_2{=}x_2, V_3{=}x_3) = \mathrm{P}(V_1{=}x_1|V_3{=}x_3)$

As we will see, effective calculation of probabilities and probabilistic inference in general is based on making assumptions about dependence and independence of variables. Two random variables V_1 and V_2 are independent if:

$$\mathrm{P}(V_1 = x_1, V_2 = x_2) = \mathrm{P}(V_1 = x_1) \cdot \mathrm{P}(V_2 = x_2)$$

11.5 Computational Tasks in Probabilistic Modeling

The computational tasks in probabilistic modeling can be divided into the following types of tasks:

1. Evaluation: compute probability of a complete configuration
2. Simulation: (sampling, generation) generate random configurations
3. Inference: has the following sub-tasks:

  3.a Marginalization: computing probability of a partial configuration,
  3.b Conditioning: computing conditional probability of a completion given an observation,
  3.c Completion: finding the most probable completion, given an observation

4. Learning: learning parameters of a model from data.

We have four main tasks, and the task of probabilistic inference is divided further in to three subtasks. A more detailed description of these tasks follows:

1. Evaluation is the task of computing the probability of a full configuration in a probabilistic model; i.e., calculating

$$\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n)$$

A model definition usually provides a straightforward approach to calculate this probability given that we know all parameters of the model.

2. Simulation is the task of producing full configurations according to a given model. Those configurations should satisfy the model conditions, which typically means that probabilities of some events should converge to probabilities specified by the model over many simulated configurations.

3. Inference is divided into three different subtasks: marginalization, conditioning, and completion. **Marginalization (3.a)** is the problem of computing a marginal probability; i.e., the probability of a partial configuration, such as:

$$\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k),$$

where $1 \le k < n$. **Conditioning (3.b)** is the problem of computing a conditional probability, which in terms of variables of a model typically means a conditional probability of the values of some variables, given the values of some other variables. For example, a conditioning task could be expressed as:

$$\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k \,|\, V_{k+1} = x_{k+1}, \ldots, V_n = x_n).$$

Generally, we can do efficiently marginalization, we can also do conditioning by definition, as shown in this formula:

$$\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k \mid V_{k+1} = x_{k+1}, \ldots, V_n = x_n) = \frac{\mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n)}{\mathrm{P}(V_{k+1} = x_{k+1}, \ldots, V_n = x_n)}$$

Completion (3.c) is the problem of finding the most probable assignment of some variables, usually called **hidden variables**, given some other variables, usually called **observed variables**. We approach it by finding the assignments of the hidden variables for which the conditional probability of those assignments given assignments of observed variables is maximized. For example, this task can be described using the formula

$$\operatorname*{arg\,max}_{x_1,\ldots,x_k} \mathrm{P}(V_1 = x_1, V_2 = x_2, \ldots, V_k = x_k \mid V_{k+1} = x_{k+1}, \ldots, V_n = x_n).$$

In the completion formula $\mathrm{P}(\alpha|\beta)$, the variables in the α part are called **hidden variables** and the variables in the β part are called **observed variables**, or **observables**, because normally know, or can observe, variables in the part β, and we do not know the hidden variables in the part α.

4. Learning is the task of determining the parameters of a probabilistic model given the data that the model is supposed to describe. Learning model parameters from incomplete data can be challenging. If we have a dataset with a list of full configurations we can calculate the parameters by counting the occurrences of interesting events. This is called Maximum Likelihood Estimation (MLE), since it can be shown that in this way we obtain a model that gives highest likelihood of data used in learning.

To illustrate probabilistic modelling and the computational tasks, we will use the following illustrative example in spam detection.

Illustrative Example: Spam Detection

The problem of spam detection in e-mail is the problem of automatically detecting whether an arbitrary e-mail message is spam or not. In a toy model, we assume that we can detect whether a message is spam or not relying only on the fact whether the ‘Subject:’ header of the message is capitalized (i.e., completely written in uppercase letters) and whether the ‘Subject:’ header contains the word ‘free’ (uppercase or lowercase). For example, “NEW MORTGAGE RATE” is likely the subject of a spam message, as well as “Money for Free,” “FREE lunch,” etc.

Hence, our model is based on the following three random variables and each of them gets one of two values Y (for Yes) or N (for No):

Caps	=	‘Y’ if the message subject line does not contain lowercase letter, ‘N’ otherwise,
Free	=	‘Y’ if the word ‘free’ appears in the message subject line (letter case is ignored), ‘N’ otherwise, and
Spam	=	‘Y’ if the message is spam, and ‘N’ otherwise.

In order to learn what happens in real-world, we open our mailbox, which serves as our random source, randomly select 100 messages and count how many times each configuration appears.

We might obtain the following table:

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

What are examples of computational tasks in this example?

Let us consider our first, straightforward model, called **Joint Distribution Model.**

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:

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

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

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

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 $p_1, p_2, \dots$, 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: $p_1, p_2, \dots, p_{m^n}$: $I_1 = [0, p_1)$, $I_2 = [p_1, p_1 + p_2)$, $I_3 = [p_1 + p_2, p_1 + p_2 + p_3), \dots I_{m^n} = [p_1 + p_2 + \dots + p_{m^n-1}, 1)$
2. Generate a random number r from the interval $[0, 1)$
3. r will fall exactly into one of the above intervals, e.g.: $I_i = [p_1 + \dots + p_{i-1}, p_1 + \dots + p_{i-1} + p_i)$
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 $y_{k+1}, \dots, y_n$. 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:

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

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 $(x_1, ..., x_k)$.

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

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:

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

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

and similarly,

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

and

Spam	$\mathrm{P}(\mathit{Spam})$
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 $(\mathit{Caps} = Y, \mathit{Free} = N, \mathit{Spam} = N)$ in the fully independent model is:

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

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

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

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:

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

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

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

3.b Conditioning in Fully Independent Model

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

Only have to lookup and multiply $n-k$ numbers.

3.c Completion in Fully Independent Model

$$\begin{aligned}
y^*_{k+1},...,y^*_n &= \operatorname*{arg\,max}_{y_{k+1},\ldots,y_n} \mathrm{P}(V_{k+1}=y_{k+1},...,V_n=y_n|V_1=x_1,...,V_k=x_k) \\
&= \operatorname*{arg\,max}_{y_{k+1},\ldots,y_n} \mathrm{P}(V_{k+1}=y_{k+1})\cdots\mathrm{P}(V_n=y_n) \\
&= \operatorname*{arg\,max}_{y_{k+1}} \mathrm{P}(V_{k+1}=y_{k+1}) \left[\operatorname*{arg\,max}_{y_{k+2}} \cdots \left[\operatorname*{arg\,max}_{y_n} \mathrm{P}(V_n=y_n) \right] \right] \\
&\quad \text{(Since } \max \text{ and arg max distributes over product just like sum.} \\
&\quad \text{That is, } \max_i ax_i = a \max_i x_i \text{ (for } a, x_i \geq 0\text{)} \\
&\quad \text{just like } \textstyle\sum_i ax_i = a \sum_i x_i\text{.)} \\
&= \left[\operatorname*{arg\,max}_{y_{k+1}} \mathrm{P}(V_{k+1}=y_{k+1}) \right] \cdots \left[\operatorname*{arg\,max}_{y_n} \mathrm{P}(V_n=y_n) \right] \\
&= \left[\operatorname*{arg\,max}_{y_{k+1}} p_{k+1,y_{k+1}} \right] \cdots \left[\operatorname*{arg\,max}_{y_n} p_{n,y_n} \right]
\end{aligned}$$

Only have to search through m possible completions for each of the $n-k$ variables separately.

Joint Distribution Model vs. Fully Independent Model

The Fully Independent Model addresses the previous issues with the joint distribution model, but it suffers from a too strong assumption and too little structure, so it usually does not model accurately the real relationships among variables.

Structured probability models are a compromise solution between previous two models. Structured probability models are more efficient than the joint distribution model and they address the issue of the sparse training data, and in the same time they model important dependencies among random variables.

One of the simplest models of this kind is the Naïve Bayes Model.