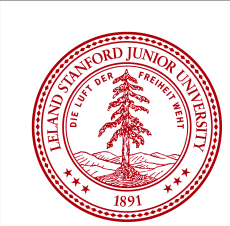
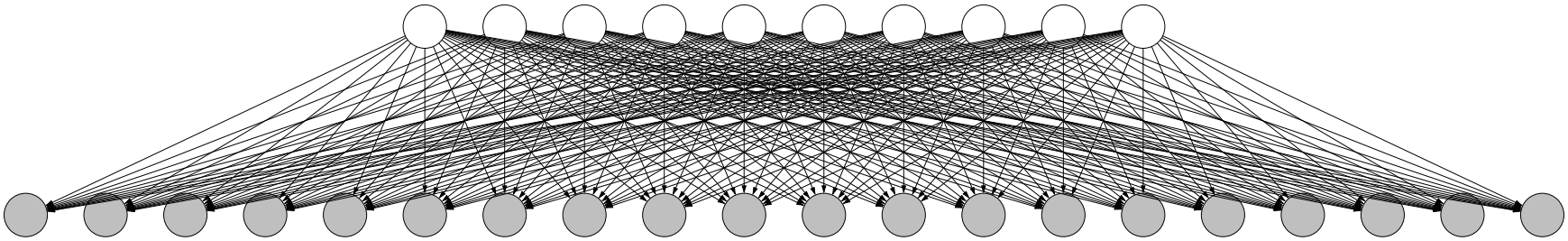


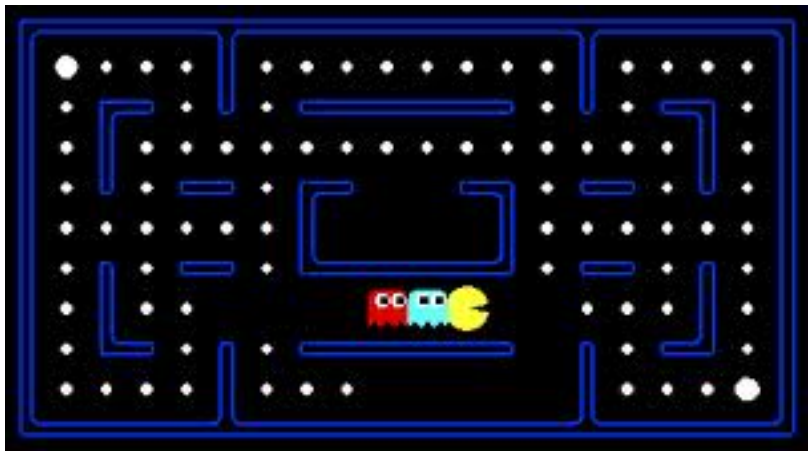
# Lecture 13: Bayesian networks I



# Announcements

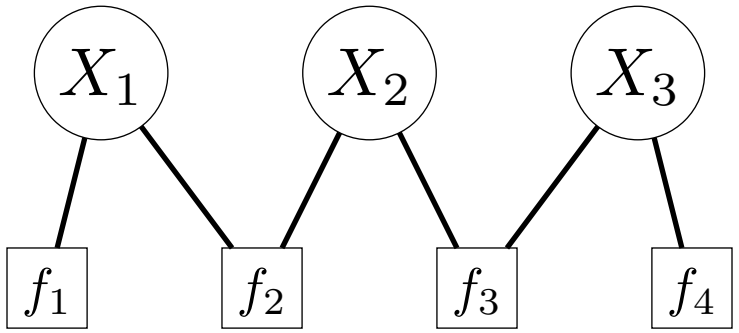
- **scheduling** is due tomorrow
- **car** is due next Tuesday
- **p-progress** is due next Thursday
- **exam** is in two weeks

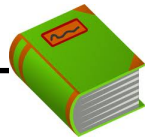
# Pac-Man competition



1. (1765) Adam Keppler
2. (1764) Allan Li
3. (1763.5) Jiahao Zhang

# Review: definition



**Definition: factor graph**

Variables:

$X = (X_1, \ldots, X_n)$, where $X_i \in \text{Domain}_i$

Factors:

$f_1, \ldots, f_m$, with each $f_j(X) \geq 0$

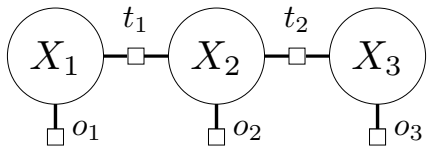
$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

- Last week, we talked about factor graphs, which uses the product of factors to specify a weight Weight$(x)$ for each assignment $x$ in a compact way. The stated objective was to find the maximum weight assignment.
- Given any factor graph, we saw a number of algorithms (backtracking search, beam search, Gibbs sampling, variable elimination) for (approximately) optimizing this objective.

# Review: object tracking

**Problem: object tracking**

Sensors report positions: $0, 2, 2$. Objects don't move very fast and sensors are a bit noisy. What path did the object take?



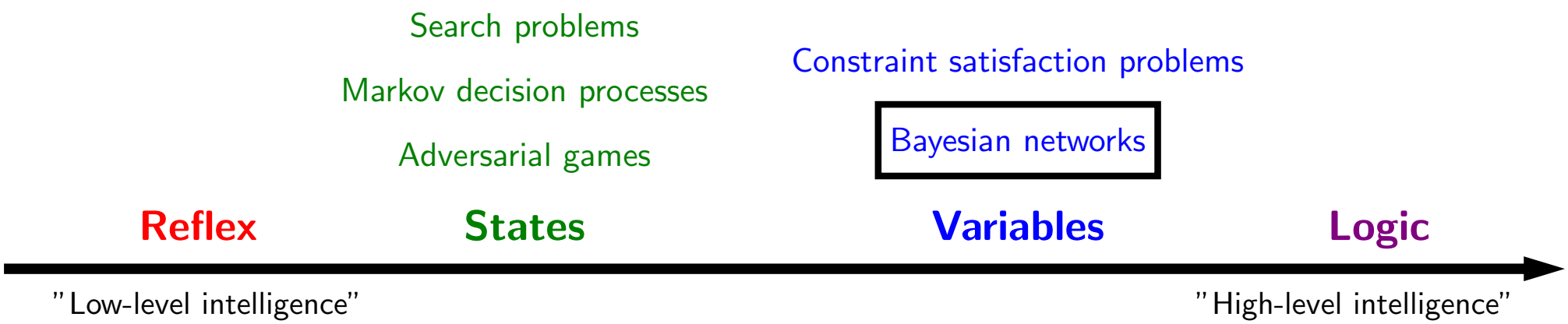

- Variables $X_i$: location of object at time $i$
- Transition factors $t_i(x_i, x_{i+1})$: incorporate physics
- Observation factors $o_i(x_i)$: incorporate sensors

[demo: `maxVariableElimination()`]

How do we **interpet** the factors?

- As an example, recall the object tracking example. We defined observation factors to capture the fact that the true object position is close to the sensor reading, and the transition factors to capture the fact that the true object positions across time are close to each other.
- We just set them rather arbitrarily. Is there a more principled way to think about these factors beyond being non-negative functions?

# Course plan




**Machine learning**

- Much of this class has been on developing modeling frameworks. We started with state-based models, where we cast real-world problems as finding paths or policies through a state graph.
- Then, we saw that for a large class of problems (such as scheduling), it was much more convenient to use the language of factor graphs.
- While factor graphs could be reduced to state-based models by fixing the variable ordering, we saw that they also led to notions of treewidth and variable elimination, which allowed us to understand our models much better.
- In this lecture, we will introduce another modeling framework, Bayesian networks, which are factor graphs imbued with the language of probability. This will give probabilistic life to the factors of factor graphs.

# Roadmap

**Basics**

Probabilistic programs

Inference

- Bayesian networks were popularized in AI by Judea Pearl in the 1980s, who showed that having a coherent probabilistic framework is important for **reasoning under uncertainty**.
- There is a lot to say about the Bayesian networks (CS228 is an entire course about them and their cousins, Markov networks). So we will devote most of this lecture focusing on modeling.

# Review: probability

**Random variables**: sunshine $S \in \{0, 1\}$, rain $R \in \{0, 1\}$

**Joint distribution**:

$\mathbb{P}(S, R) =$

| $s$ | $r$ | $\mathbb{P}(S = s, R = r)$ |
|---|---|---|
| 0 | 0 | 0.20 |
| 0 | 1 | 0.08 |
| 1 | 0 | 0.70 |
| 1 | 1 | 0.02 |

**Marginal distribution**:

$\mathbb{P}(S) =$

| $s$ | $\mathbb{P}(S = s)$ |
|---|---|
| 0 | 0.28 |
| 1 | 0.72 |

(aggregate rows)

**Conditional distribution**:

$\mathbb{P}(S \mid R = 1) =$

| $s$ | $\mathbb{P}(S = s \mid R = 1)$ |
|---|---|
| 0 | 0.8 |
| 1 | 0.2 |

(select rows, normalize)

- Before introducing Bayesian networks, let's review probability (at least the relevant parts). We start with an example about the weather. Suppose we have two boolean random variables, $S$ and $R$ representing whether it is sunny and rainy. Think of an assignment to $(S, R)$ as representing a possible state of the world.
- The **joint distribution** specifies a probability for each assignment to $(S, R)$ (state of the the world). We use lowercase letters (e.g., $s$ and $r$) to denote values and uppercase letters (e.g., $S$ and $R$) to denote random variables. Note that $\mathbb{P}(S = s, R = r)$ is a probability (a number) while $\mathbb{P}(S, R)$ is a distribution (a table of probabilities). We don't know what state of the world we're in, but we know what the probabilities are (there are no unknown unknowns). The joint distribution contains all the information and acts as the central source of truth, like a database.
- From it, we can derive a **marginal distribution** over a subset of the variables. We get this by aggregating the rows that share the same value of $S$. The interpretation is that we are interested in $S$. We don't explicitly care about $R$, but we want to take into account $R$'s effect on $S$. We say that $R$ is **marginalized out**. This is a special form of elimination. In the last lecture, we leveraged max-elimination, where we took the max over the eliminated variables; here, we are taking a sum.
- The **conditional distribution** selects rows of the table matching the condition (right of the bar), and then normalizes the probabilities so that they sum to 1. The interpretation is that we observe the condition $(R = 1)$ and are interested in $S$. This is the conditioning that we saw for factor graphs, but where we normalize the selected rows to get probabilities.

# Probabilistic inference

Joint distribution (probabilistic database):

$$\mathbb{P}(S, R, T, A)$$

Probabilistic inference:

- **Condition** on evidence (traffic, autumn): $T = 1, A = 1$
- Interested in **query** (rain?): $R$

$$\mathbb{P}(\underbrace{R}_{\text{query}} \mid \underbrace{T = 1, A = 1}_{\text{condition}})$$

($S$ is **marginalized out**)

- We should think about each assignment $x$ as a possible state of the world (it's raining, it's not sunny, there is traffic, it is autumn, etc.). Think of the joint distribution as one giant database that contains full information about how the world works.
- In practice, we'd like to ask questions by querying this probabilistic database. First, we observe some evidence, which effectively fixes some of the variables. Second, we are interested in the distribution of some set of variables which we didn't observe. This forms a query, and the process of answering this query (computing the desired distribution) is called **probabilistic inference**.

# Challenges

Modeling: How to specify a joint distribution $\mathbb{P}(X_1, \ldots, X_n)$ **compactly**?

Bayesian networks (factor graphs to specify joint distributions)

Inference: How to compute queries $\mathbb{P}(R \mid T = 1, A = 1)$ **efficiently**?

Variable elimination, Gibbs sampling, particle filtering (analogue of algorithms for finding maximum weight assignment)

- In general, a joint distribution over $n$ variables has size exponential in $n$. From a modeling perspective, how do we even specify an object that large? Here, we will see that Bayesian networks, based on factor graphs, offer an elegant solution.
- From an algorithms perspective, there is still the question of how we perform probabilistic inference efficiently. In the next lecture, we will see how we can adapt all of the algorithms that we saw before for computing maximum weight assignments in factor graphs, essentially by replacing a max with a sum.
- The two desiderata are rather synergistic, and it is the same property — conditional independence — that makes both possible.

cs221.stanford.edu/q

# Question

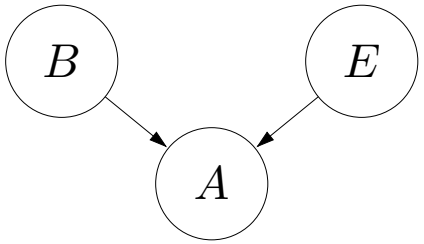
Earthquakes and burglaries are independent events that will cause an alarm to go off. Suppose you hear an alarm. How does hearing on the radio that there's an earthquake change your beliefs about burglary?

| it increases the probability of burglary |
| --- |
| it decreases the probability of burglary |
| it does not change the probability of burglary |

- Situations like these arise all the time in practice: we have a lot of unknowns which are all dependent on one another. If we obtain evidence on some of these unknowns, how does that affect our belief about the other unknowns? This is called **reasoning under uncertainty**.
- In this lecture, we'll see how we can perform this type of reasoning under uncertainty in a principled way using Bayesian networks.

# Bayesian network (alarm)

$$\mathbb{P}(B = b, E = e, A = a) \stackrel{\text{def}}{=} p(b)p(e)p(a \mid b, e)$$




| $b$ | $p(b)$ |
|---|---|
| 1 | $\epsilon$ |
| 0 | $1-\epsilon$ |

| $e$ | $p(e)$ |
|---|---|
| 1 | $\epsilon$ |
| 0 | $1-\epsilon$ |

| $b$ | $e$ | $a$ | $p(a \mid b, e)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

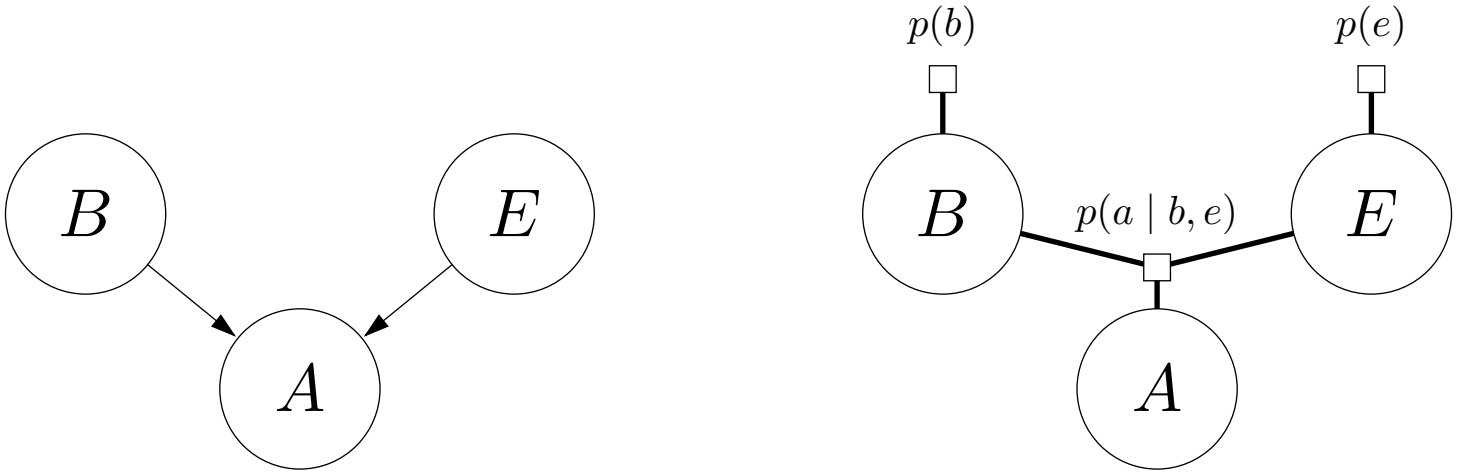
$$p(b) = \epsilon \cdot [b = 1] + (1 - \epsilon) \cdot [b = 0]$$

$$p(e) = \epsilon \cdot [e = 1] + (1 - \epsilon) \cdot [e = 0]$$

$$p(a \mid b, e) = [a = (b \vee e)]$$

- Let us try to model the situation. First, we establish that there are three variables, $B$ (burglary), $E$ (earthquake), and $A$ (alarm).
- Second, we connect up the variables to model the dependencies. Unlike in factor graphs, these dependencies are represented as **directed** edges. You can intuitively think about the directionality as suggesting causality, though what this actually means is a deeper question and beyond the scope of this class.
- Third, for each variable, we specify a **local conditional distribution** (a factor) of that variable given its parent variables. In this example, $B$ and $E$ have no parents while $A$ has two parents, $B$ and $E$. This local conditional distribution is what governs how a variable is generated.
- Fourth, we define the joint distribution over all the random variables as the product of all the local conditional distributions.
- Note that we write the local conditional distributions using $p$, while $\mathbb{P}$ is reserved for the joint distribution over all random variables, which is defined as the product.

# Bayesian network (alarm)




$$\mathbb{P}(B = b, E = e, A = a) = p(b)p(e)p(a \mid b, e)$$

Bayesian networks are a special case of factor graphs!

Note: single factor that connects **all** parents!

- Note that the local conditional distributions (e.g., $p(a \mid b, e)$) are non-negative so they can be thought of simply as factors of a factor graph. The joint probability of an assignment is then the weight of that assignment.
- In this light, Bayesian networks are just a type of factor graph, but with additional structure and interpretation.

# Probabilistic inference (alarm)

Joint distribution:

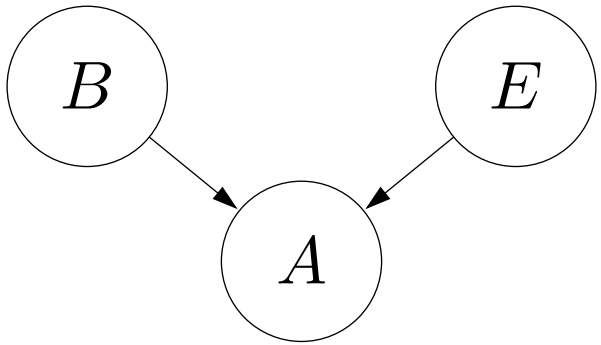
| $b$ | $e$ | $a$ | $\mathbb{P}(B=b, E=e, A=a)$ |
|---|---|---|---|
| 0 | 0 | 0 | $(1-\epsilon)^2$ |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | $(1-\epsilon)\epsilon$ |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | $\epsilon(1-\epsilon)$ |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | $\epsilon^2$ |

Queries: $\mathbb{P}(B)$? $\mathbb{P}(B \mid A=1)$? $\mathbb{P}(B \mid A=1, E=1)$?

[demo: $\epsilon = 0.05$]

- Bayesian networks can be used to capture common reasoning patterns under uncertainty (which was one of their first applications).
- Consider the following model: Suppose the probability of an earthquake is $\epsilon$ and the probability of a burglary is $\epsilon$ and both are independent. Suppose that the alarm always goes off if either an earthquake or a burglary occurs.
- In the prior, we can eliminate $A$ and $E$ and get that the probability of the burglary is $\epsilon$.
- Now suppose we hear the alarm $A = 1$. The probability of burglary is now $\mathbb{P}(B = 1 \mid A = 1) = \frac{1}{2-\epsilon}$.
- Now suppose that you hear on the radio that there was an earthquake ($E = 1$). Then the probability of burglary goes down to $\mathbb{P}(B = 1 \mid A = 1, E = 1) = \epsilon$ again.

# Explaining away

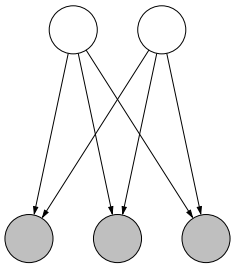



**Key idea: explaining away**

Suppose two causes positively influence an effect. Conditioned on the effect, conditioning on one cause reduces the probability of the other cause.

- This last phenomenon has a special name: **explaining away**. Suppose we have two **cause** variables $B$ and $E$, which are parents of an **effect** variable $A$. Assume the causes influence the effect positively (e.g., through the OR function).
- Conditioned on the effect $A = 1$, there is some posterior probability of $B$. Conditioned on the effect $A = 1$ and the other cause $E = 1$, the new posterior probability is reduced. We then say that the other cause $E$ has explained away $B$.

# Definition



## Definition: Bayesian network

Let $X = (X_1, \ldots, X_n)$ be random variables.

A **Bayesian network** is a directed acyclic graph (DAG) that specifies a joint distribution over $X$ as a product of local conditional distributions, one for each node:

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) \stackrel{\text{def}}{=} \prod_{i=1}^{n} p(x_i \mid x_{\text{Parents}(i)})$$

- Without further ado, let's define a Bayesian network formally. A Bayesian network defines a large joint distribution in a modular way, one variable at a time.
- First, the graph structure captures what other variables a given variable depends on.
- Second, we specify a local conditional distribution for variable $X_i$, which is a function that specifies a distribution over $X_i$ given an assignment $x_{\mathsf{Parents}(i)}$ to its parents in the graph (possibly no parents). The joint distribution is simply **defined** to be the product of all of the local conditional distributions together.
- Notationally, we use lowercase $p$ (in $p(x_i \mid x_{\mathsf{Parents}(i)})$) to denote a local conditional distribution, and uppercase $\mathbb{P}$ to denote the induced joint distribution over all variables. While we will see that the two coincide, it is important to keep these things separate in your head!

# Special properties

Key difference from general factor graphs:

**Key idea: locally normalized**

All factors (local conditional distributions) satisfy:

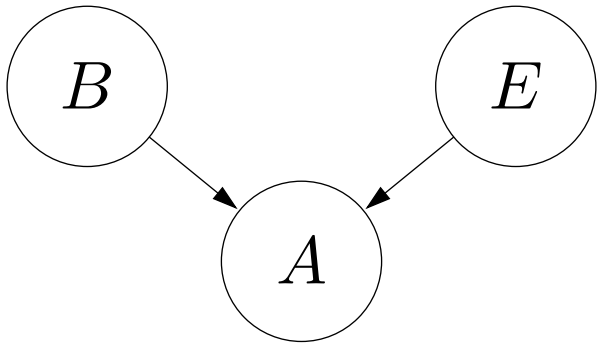
$$\sum_{x_i} p(x_i \mid x_{\text{Parents}(i)}) = 1 \text{ for each } x_{\text{Parents}(i)}$$

Implications:

- Consistency of sub-Bayesian networks
- Consistency of conditional distributions

- But Bayesian networks are more than that. The key property is that all the local conditional distributions, being distributions, sum to 1 over the first argument.
- This simple property results in two important properties of Bayesian networks that are not present in general factor graphs.

# Consistency of sub-Bayesian networks




A short calculation:

$$\mathbb{P}(B = b, E = e) \stackrel{\text{def}}{=} \sum_a \mathbb{P}(B = b, E = e, A = a)$$

$$\stackrel{\text{def}}{=} \sum_a p(b)p(e)p(a \mid b, e)$$

$$= p(b)p(e) \sum_a p(a \mid b, e)$$

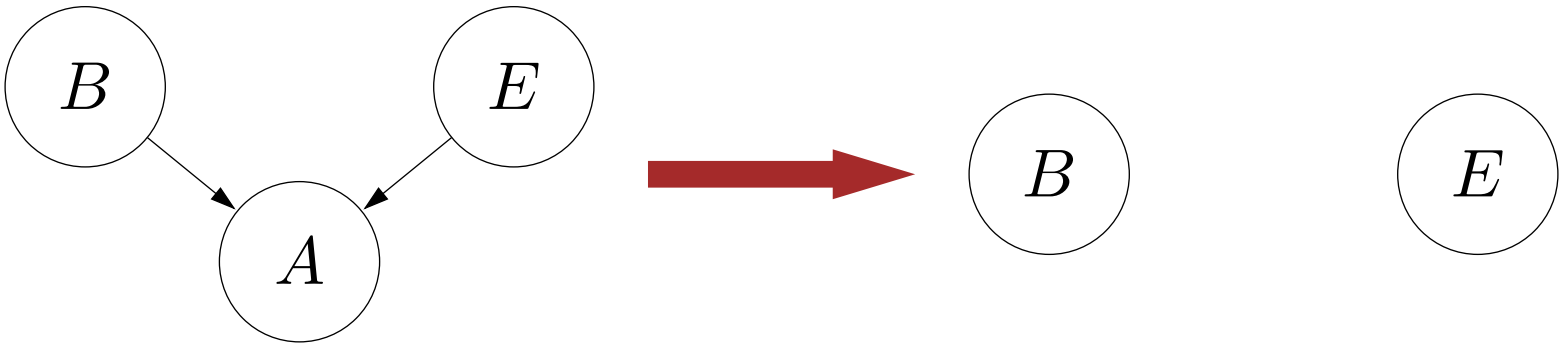
$$= p(b)p(e)$$

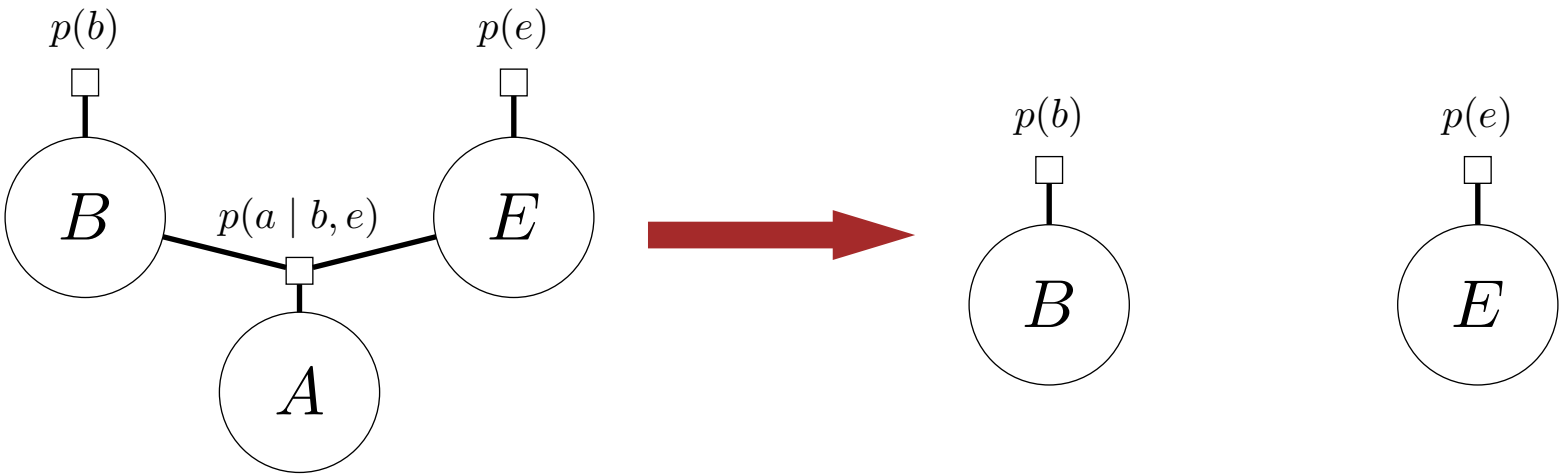
- First, let's see what happens when we marginalize $A$ (by performing algebra on the joint probability). We see that we end up with $p(b)p(e)$, which actually defines a sub-Bayesian network with one fewer variable, and the same local conditional probabilities.
- If one marginalizes out all the variables, then one gets 1, which verifies that a Bayesian network actually defines a probability distribution.
- The philosophical ramification of this property is that there could be many other variables that depend on the variables you've modeled (earthquakes also impact traffic) but as long as you don't observe them, they can be ignored mathematically (ignorance is bliss). Note that this doesn't mean that knowing about the other things isn't useful.

# Consistency of sub-Bayesian networks

**Key idea: marginalization**

Marginalization of a leaf node yields a Bayesian network without the node.



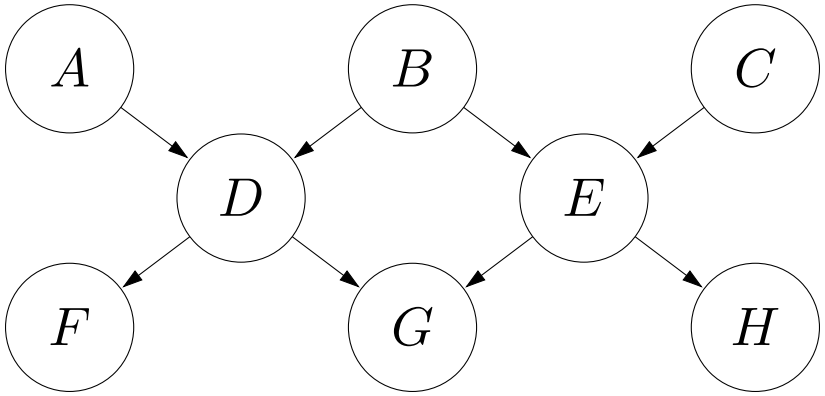



- This property is very attractive, because it means that whenever we have a large Bayesian network, where we don't care about some of the variables, we can just remove them (graph operations), and this encodes the same distribution as we would have gotten from marginalizing out variables (algebraic operations). The former, being visual, can be more intuitive.
- Note that if we marginalized only based on the factor graph representation, we would have kept the factor between $B$ and $E$, which is too conservative. This is because the factor graph representation doesn't "know" about the probabilistic structure of its factors, and the factor has to be kept in general.

# Consistency of local conditionals

**Key idea: local conditional distributions**

Local conditional distributions (factors) are the true conditional distributions.




$$\underbrace{\mathbb{P}(D = d \mid A = a, B = b)}_{\text{from probabilistic inference}} = \underbrace{p(d \mid a, b)}_{\text{by definition}}$$

- Note that the local conditional distributions $p(d \mid a, b)$ are simply defined by the user. On the other hand, the quantity $\mathbb{P}(D = d \mid A = a, B = b)$ is not defined, but follows from probabilistic inference on the joint distribution defined by the Bayesian network.
- It's not clear a priori that the two have anything to do with each other. The second special property that we get from using Bayesian networks is that the two are actually the same.
- To show this, we can remove all non-ancestors of $D$ by the consistency of sub-Bayesian networks, leaving us with the Bayesian network $\mathbb{P}(A = a, B = b, D = d) = p(a)p(b)p(d \mid a, b)$. By the chain rule, $\mathbb{P}(A = a, B = b, D = d) = \mathbb{P}(A = a, B = b)\mathbb{P}(D = d \mid A = a, B = b)$. If we marginalize out $D$, then we are left with the Bayesian network $\mathbb{P}(A = a, B = b) = p(a)p(b)$. From this, we can conclude that $\mathbb{P}(D = d \mid A = a, B = b) = p(d \mid a, b)$.
- This argument generalizes to any Bayesian network and local conditional distribution.

# Medical diagnosis

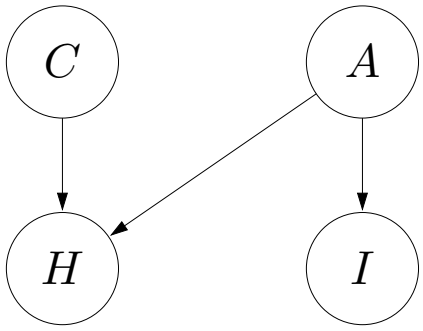
**Problem: cold or allergies?**

You are coughing and have itchy eyes. Do you have a cold or allergies?

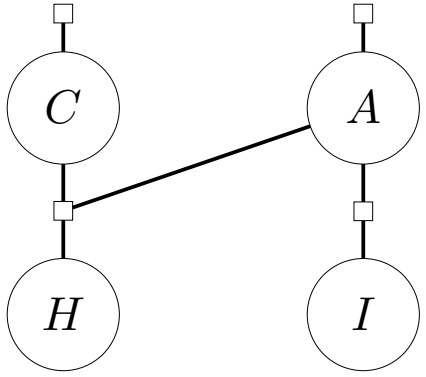
[whiteboard] [demo]

Variables: **C**old, **A**llergies, Coug**h**, **I**tchy eyes

Bayesian network:

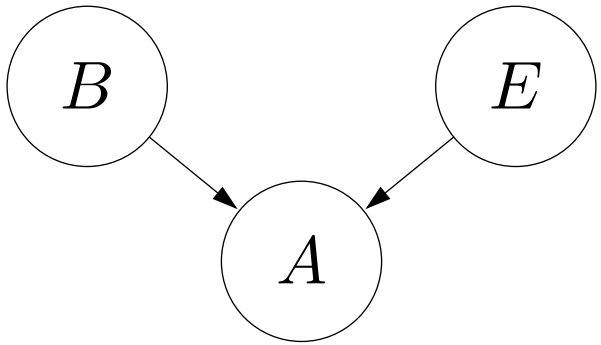



Factor graph:



- Here is another example (a cartoon version of Bayesian networks for medical diagnosis). Allergies and cold are the two hidden variables that we'd like to infer (we have some prior over these two). Cough and itchy eyes are symptoms that we observe as evidence, and we have some likelihood model of these symptoms given the hidden causes.
- Formally, we are interested in $\mathbb{P}(C, A \mid H = 1, I = 1)$.
- We can use the demo to infer the hidden state given the evidence.

# Summary so far




- Random variables capture state of world
- Edges between variables represent dependencies
- Local conditional distributions $\Rightarrow$ joint distribution
- Probabilistic inference: ask questions about world
- Captures reasoning patterns (e.g., explaining away)
- Factor graph interpretation (for inference later)

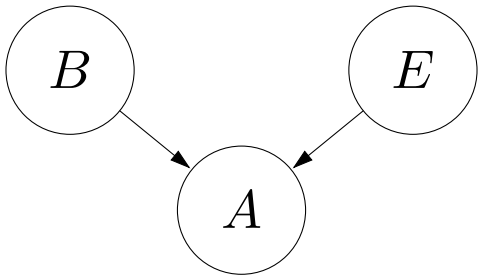
# Roadmap

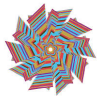
Basics

**Probabilistic programs**

Inference

# Probabilistic programs



**Probabilistic program: alarm**

$$B \sim \text{Bernoulli}(\epsilon)$$
$$E \sim \text{Bernoulli}(\epsilon)$$
$$A = B \vee E$$

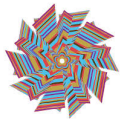
**Key idea: probabilistic program**

A randomized program that sets the random variables.

```
def Bernoulli(epsilon):
    return random.random() < epsilon
```

- There is another way of writing down Bayesian networks other than graphically or mathematically, and that is as a probabilistic program. A **probabilistic program** is a randomized program that invokes a random number generator to make random choices. Executing this program will assign values to a collection of random variables $X_1, \ldots, X_n$; that is, generating an assignment.
- The probability (e.g., fraction of times) that the program generates that assignment is exactly the probability under the joint distribution specified by that program.
- We should think of this program as outputting the state of the world (or at least the part of the world that we care about for our task).
- Note that the probabilistic program is only used to define joint distributions. We usually wouldn't actually run this program directly.
- For example, we show the probabilistic program for alarm. $B \sim$ Bernoulli$(\epsilon)$ simply means that $\mathbb{P}(B = 1) = \epsilon$. Here, we can think about Bernoulli$(\epsilon)$ as a randomized function (`random() < epsilon`) that returns $1$ with probability $\epsilon$ and $0$ with probability $1 - \epsilon$.

# Probabilistic program: example



**Probabilistic program: object tracking**

$X_0 = (0, 0)$

For each time step $i = 1, \ldots, n$:

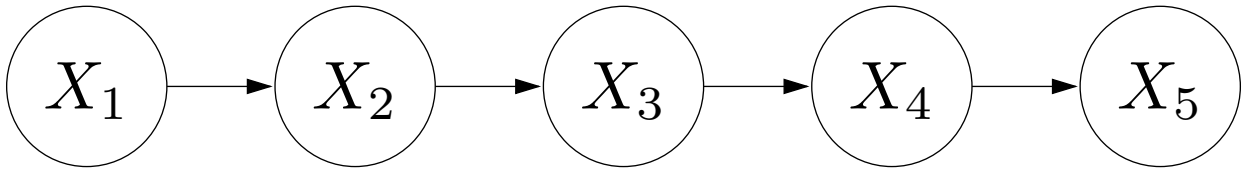
With probability $\alpha$:

$X_i = X_{i-1} + (1, 0)$ [go right]

With probability $1 - \alpha$:

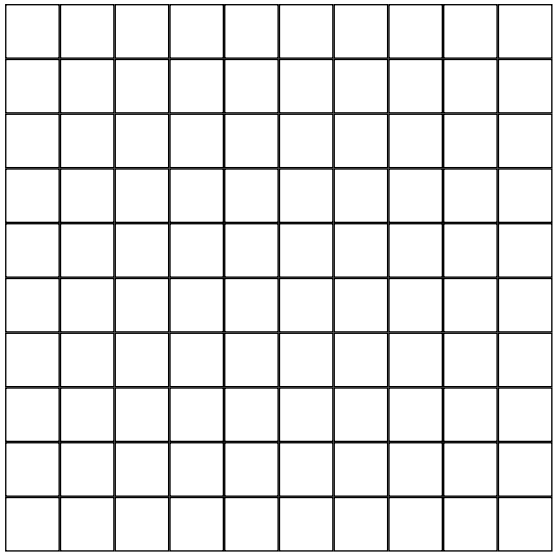
$X_i = X_{i-1} + (0, 1)$ [go down]

Bayesian network structure:



- This is a more interesting generative model since it has a for loop, which allows us to determine the distribution over a templatized set of $n$ variables rather than just $3$ or $4$.
- In these cases, variables are generally indexed by something like time or location.
- We can also draw the Bayesian network. Each $X_i$ only depends on $X_{i-1}$. This is a chain-structured Bayesian network, called a **Markov model**.

# Probabilistic program: example



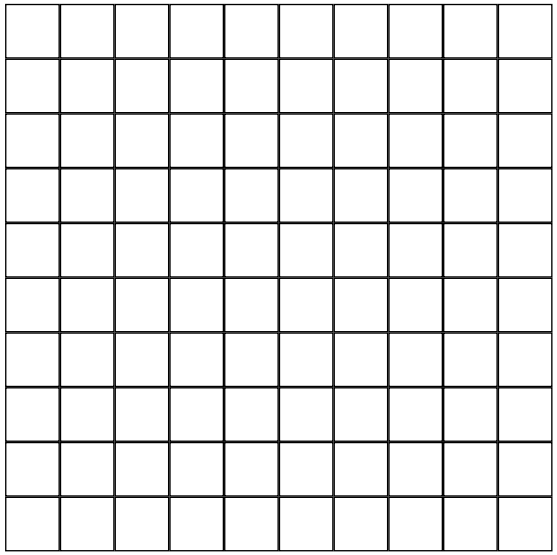
(press ctrl-enter to save)

Run

- Try clicking [Run]. Each time a new assignment of $(X_1, \ldots, X_n)$ is chosen.

# Probabilistic inference: example

**Query**: what are possible trajectories given **evidence** $X_{10} = (8, 2)$?



(press ctrl-enter to save)

Run

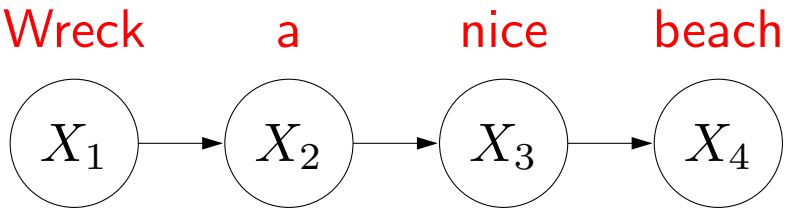
- This program only serves for defining the distribution. Now we can query that distribution and ask the question: suppose the program set $X_{10} = (8, 2)$; what is the distribution over the other variables?
- In the demo, note that all trajectories are constrained to go through $(8, 2)$ at time step $10$.

# Application: language modeling

**Probabilistic program: Markov model**

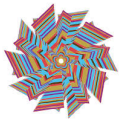
For each position $i = 1, 2, \ldots, n$:

Generate word $X_i \sim p(X_i \mid X_{i-1})$



- In the context of natural language, a Markov model is known as a bigram model. A higher-order generalization of bigram models are $n$-gram models (more generally known as higher-order Markov models).
- Language models are often used to measure the "goodness" of a sentence, mostly within the context of a larger system such as speech recognition or machine translation.

# Application: object tracking

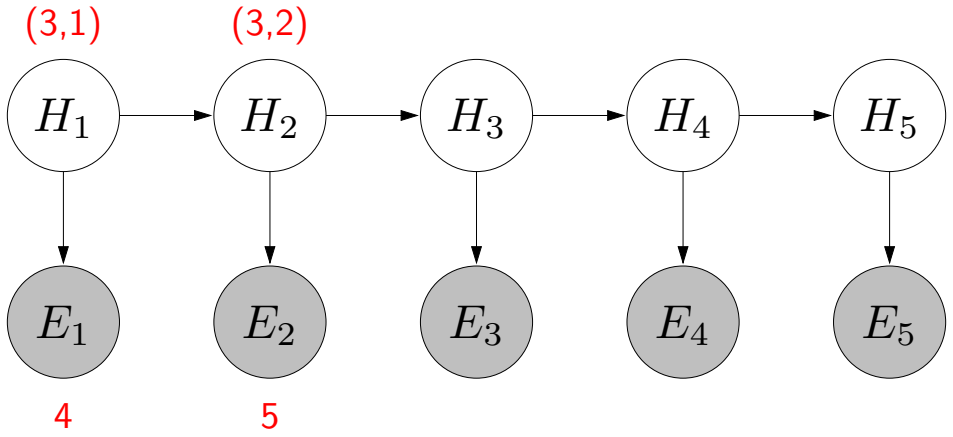


**Probabilistic program: hidden Markov model (HMM)**

For each time step $t = 1, \ldots, T$:

Generate object location $H_t \sim p(H_t \mid H_{t-1})$

Generate sensor reading $E_t \sim p(E_t \mid H_t)$




Inference: given sensor readings, where is the object?

- Markov models are limiting because they do not have a way of talking about noisy evidence (sensor readings). They can be extended quite easily to hidden Markov models, which introduce a parallel sequence of observation variables.
- For example, in object tracking, $H_t$ denotes the true object location, and $E_t$ denotes the noisy sensor reading, which might be (i) the location $H_t$ plus noise, or (ii) the distance from $H_t$ plus noise, depending on the type of sensor.
- In speech recognition, $H_t$ would be the phonemes or words and $E_t$ would be the raw acoustic signal.

# Application: multiple object tracking

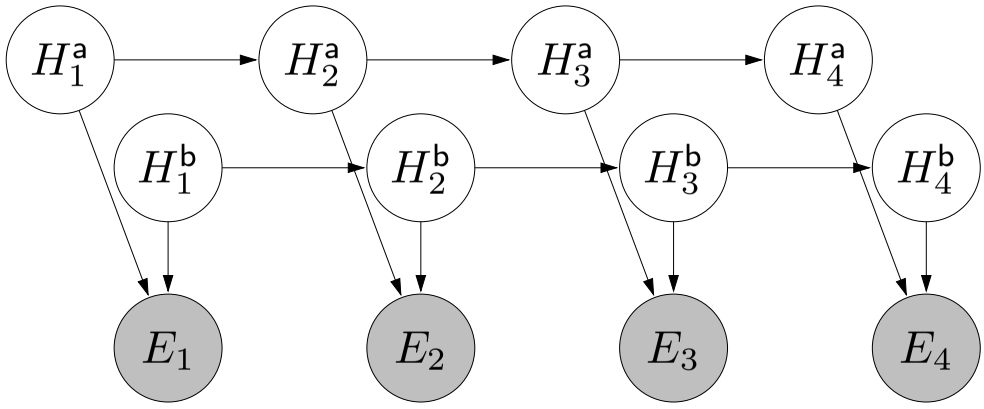
**Probabilistic program: factorial HMM**

For each time step $t = 1, \ldots, T$:

For each object $o \in \{\mathsf{a}, \mathsf{b}\}$:

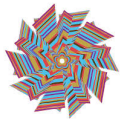
Generate location $H_t^o \sim p(H_t^o \mid H_{t-1}^o)$

Generate sensor reading $E_t \sim p(E_t \mid H_t^{\mathsf{a}}, H_t^{\mathsf{b}})$



- An extension of an HMM, called a **factorial HMM**, can be used to track multiple objects. We assume that each object moves independently according to a Markov model, but that we get one sensor reading which is some noisy aggregated function of the true positions.
- For example, $E_t$ could be the set $\{H_t^{\mathsf{a}}, H_t^{\mathsf{b}}\}$, which reveals where the objects are, but doesn't say which object is responsible for which element in the set.

# Application: document classification

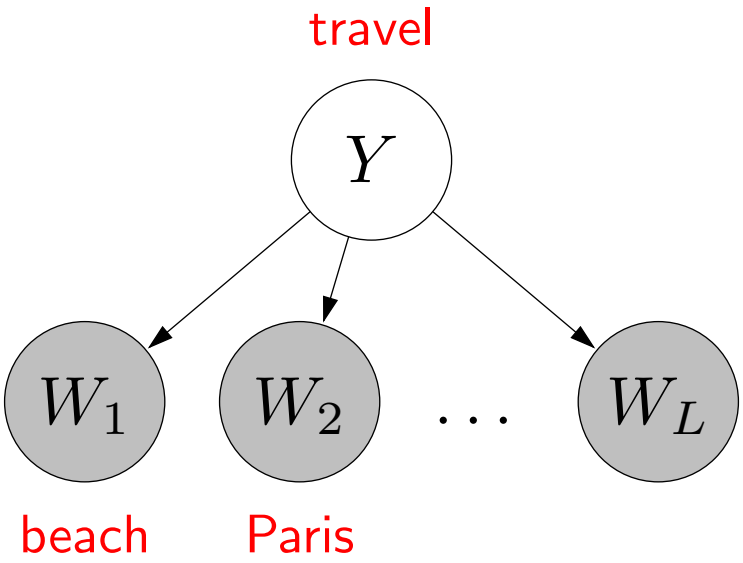


**Probabilistic program: naive Bayes**

Generate label $Y \sim p(Y)$

For each position $i = 1, \ldots, L$:

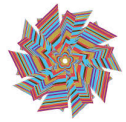
Generate word $W_i \sim p(W_i \mid Y)$




Inference: given a text document, what is it about?

- Naive Bayes is a very simple model which can be used for classification. For document classification, we generate a label and all the words in the document given that label.
- Note that the words are all generated independently, which is not a very realistic model of language, but naive Bayes models are surprisingly effective for tasks such as document classification.
- These types of models are traditionally called generative models as opposed to discriminative models for classification. Rather than thinking about how you take the input and produce the output label (e.g., using a neural network), you go the other way around: think about how the input is generated from the output (which is usually the purer, more structured form of the input).
- One advantage of using Naive Bayes for classification is that "training" is extremely easy and fast and just requires counting (as opposed to performing gradient descent).

# Application: topic modeling



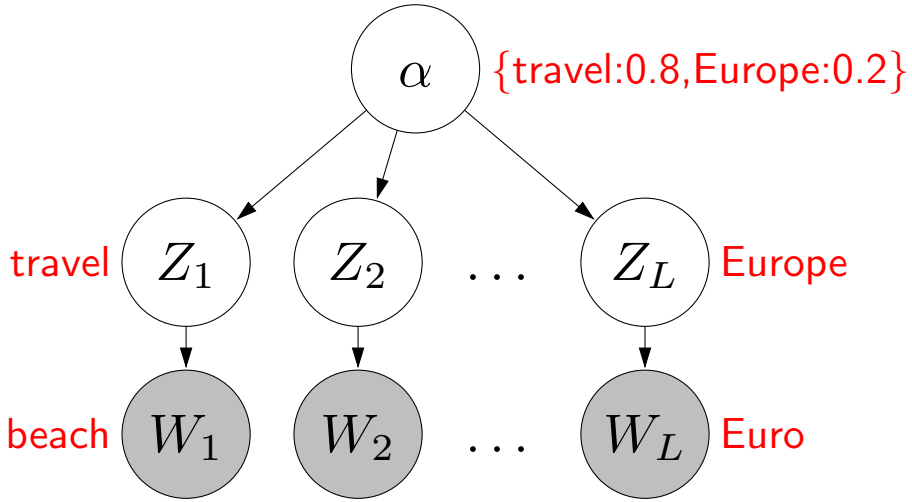
**Probabilistic program: latent Dirichlet allocation**

Generate a distribution over topics $\alpha \in \mathbb{R}^K$

For each position $i = 1, \ldots, L$:

Generate a topic $Z_i \sim p(Z_i \mid \alpha)$

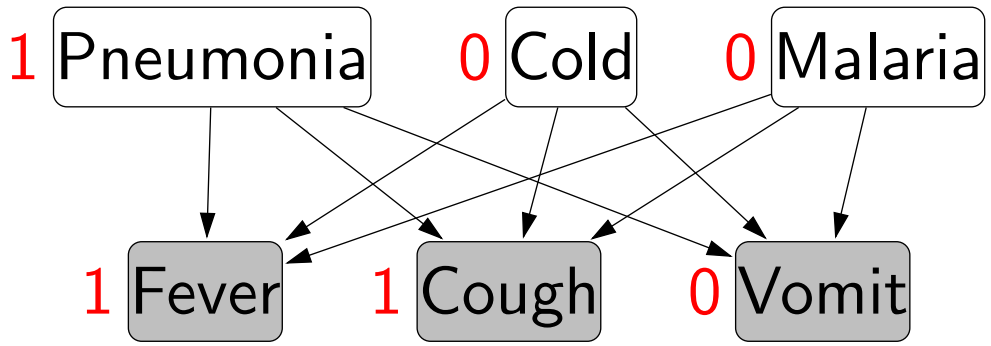
Generate a word $W_i \sim p(W_i \mid Z_i)$

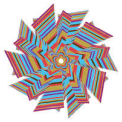



Inference: given a text document, what topics is it about?

- A more sophisticated model of text is latent Dirichlet Allocation (LDA), which allows a document to not just be about one topic (which was true in naive Bayes), but about multiple topics.
- Here, the distribution over topics $\alpha$ is chosen per document from a Dirichlet distribution. Note that $\alpha$ is a continuous-valued random variable. For each position, we choose a topic according to that per-document distribution and generate a word given that topic.
- Latent Dirichlet Alloction (LDA) has been very infuential for modeling not only text but images, videos, music, etc.; any sort of data with hidden structure. It is very related to matrix factorization.

# Application: medical diagnostics



**Probabilistic program: diseases and symptoms**

For each disease $i = 1, \ldots, m$:

Generate activity of disease $D_i \sim p(D_i)$

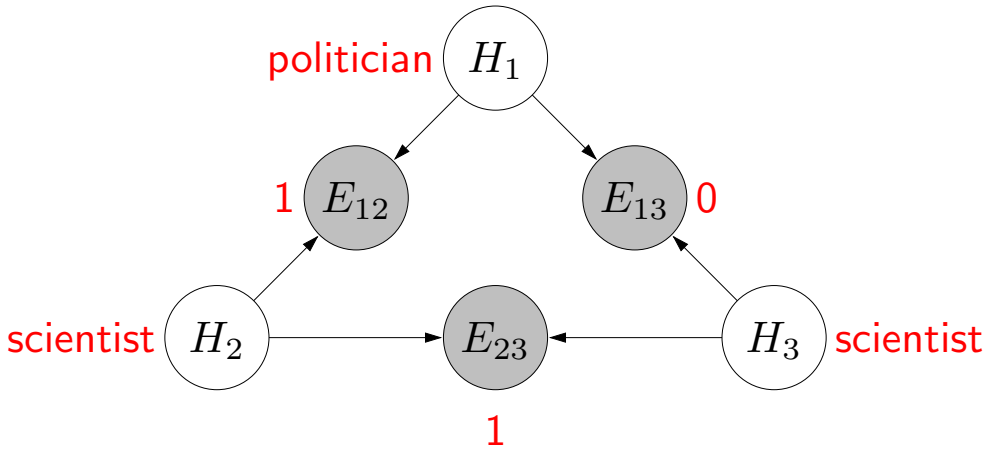
For each symptom $j = 1, \ldots, n$:

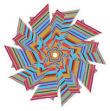
Generate activity of symptom $S_j \sim p(S_j \mid D_{1:m})$

Inference: If a patient has has a cough and fever, what disease(s) does he/she have?

- We already saw a special case of this model. In general, we would like to diagnose many diseases and might have measured many symptoms and vitals.

# Application: social network analysis



**Probabilistic program: stochastic block model**

For each person $i = 1, \ldots, n$:

Generate person type $H_i \sim p(H_i)$

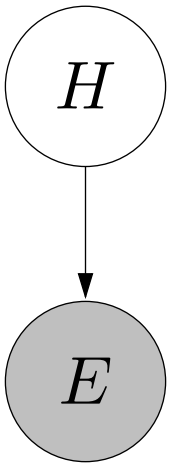
For each pair of people $i \neq j$:

Generate connectedness $E_{ij} \sim p(E_{ij} \mid H_i, H_j)$

Inference: Given a social network (graph over $n$ people), what types of people are there?

- One can also model graphs such as social networks. A very naive-Bayes-like model is that each node (person) has a "type". Whether two people interact with each other is determined solely by their types and random chance.
- Note: there are extensions called mixed membership models which, like LDA, allow each person to have multiple types.
- In summary, it is quite easy to come up with probabilistic programs that tell a story of how the world works for the domain of interest. These probabilistic programs define joint distributions over assignments to a collection of variables. Usually, these programs describe how some collection of hidden variables $H$ that you're interested in behave, and then describe the generation of the evidence $E$ that you see conditioned on $H$. After defining the model, one can do probabilistic inference to compute $\mathbb{P}(H \mid E = e)$.

# Summary so far




- Many many different types of models
- Mindset: come up with stories of how the data (input) was generated through quantities of interest (output)
- Opposite of how we normally do classification!

# Roadmap

Basics

Probabilistic programs

**Inference**

# Review: probabilistic inference

**Input**

Bayesian network: $\mathbb{P}(X_1, \ldots, X_n)$

Evidence: $E = e$ where $E \subseteq X$ is subset of variables

Query: $Q \subseteq X$ is subset of variables

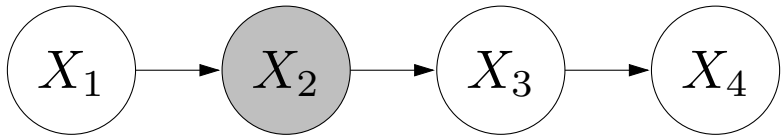
**Output**

$\mathbb{P}(Q = q \mid E = e)$ for all values $q$

Example: if coughing and itchy eyes, have a cold?

$$\mathbb{P}(C \mid H = 1, I = 1)$$

# Example: Markov model




Query: $\mathbb{P}(X_3 = x_3 \mid X_2 = 5)$ for all $x_3$

Tedious way:

$$\propto \sum_{x_1, x_4} p(x_1)p(x_2 = 5 \mid x_1)p(x_3 \mid x_2 = 5)p(x_4 \mid x_3)$$

$$\propto \left( \sum_{x_1} p(x_1)p(x_2 = 5 \mid x_1) \right) p(x_3 \mid x_2 = 5)$$

$$\propto p(x_3 \mid x_2 = 5)$$

Fast way:

[whiteboard]

- Let's first compute the query the old-fashioned way by grinding through the algebra. Then we'll see a faster, more graphical way, of doing this.
- We start by transforming the query into an expression that references the joint distribution, which allows us to rewrite as the product of the local conditional probabilities. To do this, we invoke the definition of marginal and conditional probability.
- One convenient shortcut we will take is to make use of the proportional-to ($\propto$) relation. Note that in the end, we need to construct a distribution over $X_3$. This means that any quantity (such as $\mathbb{P}(X_2 = 5)$) which doesn't depend on $X_3$ can be folded into the proportionality constant. If you don't believe this, keep it around to convince yourself that it doesn't matter. Using $\propto$ can save you a lot of work.
- Next, we do some algebra to push the summations inside. We notice that $\sum_{x_4} p(x_4 \mid x_3) = 1$ because it's a local conditional distribution. The factor $\sum_{x_1} p(x_1)p(x_2 = 5 \mid x_1)$ can also be folded into the proportionality constant.
- The final result is $p(x_3 \mid x_2 = 5)$, which matches the query as we expected by the consistency of local conditional distributions.

# General strategy

Query:

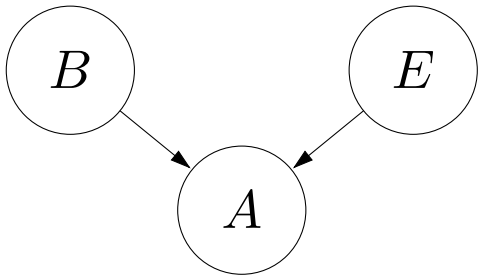
$$\mathbb{P}(Q \mid E = e)$$

**Algorithm: general probabilistic inference strategy**

- Remove (marginalize) variables that are not ancestors of $Q$ or $E$.
- Convert Bayesian network to factor graph.
- Condition on $E = e$ (shade nodes + disconnect).
- Remove (marginalize) nodes disconnected from $Q$.
- Run probabilistic inference algorithm (manual, variable elimination, Gibbs sampling, particle filtering).

- Our goal is to compute the conditional distribution over the query variables $Q \subseteq H$ given evidence $E = e$. We can do this with our bare hands by chugging through all the algebra starting with the definition of marginal and conditional probability, but there is an easier way to do this that exploits the structure of the Bayesian network.
- Step 1: remove variables which are not ancestors of $Q$ or $E$. Intuitively, these don't have an influence on $Q$ and $E$, so they can be removed. Mathematically, this is due to the consistency of sub-Bayesian networks.
- Step 2: turn this Bayesian network into a factor graph by simply introducing one factor per node which is connected to that node and its parents. It's important to include all the parents and the child into one factor, not separate factors. From here out, all we need to think about is factor graphs.
- Step 3: condition on the evidence variables. Recall that conditioning on nodes in a factor graph shades them in, and as a graph operation, rips out those variables from the graph.
- Step 4: remove nodes which are not connected to $Q$. These are independent of $Q$, so they have no impact on the results.
- Step 5: Finally, run a standard probabilistic inference algorithm on the reduced factor graph. We'll do this manually for now using variable elimination. Later we'll see automatic methods for doing this.

# Example: alarm




| $b$ | $p(b)$ |
|---|---|
| 1 | $\epsilon$ |
| 0 | $1-\epsilon$ |

| $e$ | $p(e)$ |
|---|---|
| 1 | $\epsilon$ |
| 0 | $1-\epsilon$ |

| $b$ | $e$ | $a$ | $p(a \mid b, e)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

[whiteboard]

Query: $\mathbb{P}(B)$

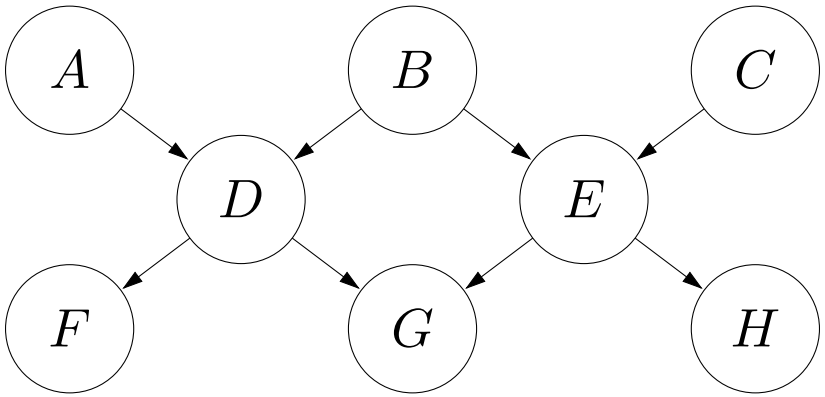
- Marginalize out $A, E$

Query: $\mathbb{P}(B \mid A = 1)$

- Condition on $A = 1$

- Here is another example: the simple v-structured alarm network from last time.
- $\mathbb{P}(B) = p(b)$ trivially after marginalizing out $A$ and $E$ (step 1).
- For $\mathbb{P}(B \mid A = 1)$, step 1 doesn't do anything. Conditioning (step 3) creates a factor graph with factors $p(b)$, $p(e)$, and $p(a = 1 \mid b, e)$. In step 5, we eliminate $E$ by replacing it and its incident factors with a new factor $f(b) = \sum_e p(e)p(a = 1 \mid b, e)$. Then, we multiply all the factors (which should only be unary factors on the query variable $B$) and normalize: $\mathbb{P}(B = b \mid A = 1) \propto p(b)f(b)$.
- To flesh this out, for $b = 1$, we have $\epsilon(\epsilon + (1 - \epsilon)) = \epsilon$. For $b = 0$, we have $(1 - \epsilon)(\epsilon + 0) = \epsilon(1 - \epsilon)$. The normalized result is thus $\mathbb{P}(B = 1 \mid A = 1) = \frac{\epsilon}{\epsilon + \epsilon(1-\epsilon)} = \frac{1}{2-\epsilon}$.
- For a probabilistic interpretation, note that all we've done is calculate $\mathbb{P}(B = b \mid A = 1) = \frac{\mathbb{P}(B=b)\mathbb{P}(A=1|B=b)}{\mathbb{P}(A=1)} = \frac{p(b)f(b)}{\sum_{b_i \in \text{Domain}(B)} p(b_i)f(b_i)}$, where the first equality follows from Bayes' rule and the second follows from the fact that the local conditional distributions are the true conditional distributions. The Bayesian network has simply given us a methodical, algorithmic way to calculate this probability.

# Example: A-H (section)




[whiteboard]

Query: $\mathbb{P}(C \mid B = b)$

- Marginalize out everything else, note $C \perp\!\!\!\perp B$

Query: $\mathbb{P}(C, H \mid E = e)$

- Marginalize out $A, D, F, G$, note $C \perp\!\!\!\perp H \mid E$

- In the first example, once we marginalize out all variables we can, we are left with $C$ and $B$, which are disconnected. We condition on $B$, which just removes that node, and so we're just left with $\mathbb{P}(C) = p(c)$, as expected.
- In the second example, note that the two query variables are independent, so we can compute them separately. The result is $\mathbb{P}(C = c, H = h \mid E = e) \propto p(c)p(h \mid e) \sum_b p(b)p(e \mid b, c)$.
- If we had the actual values of these probabilities, we can compute these quantities.

# Summary

Bayesian networks: modular definition of large joint distribution over variables

Probabilistic inference: condition on evidence, query variables of interest

Next time: algorithms for probabilistic **inference**