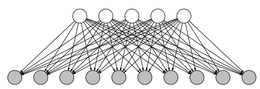
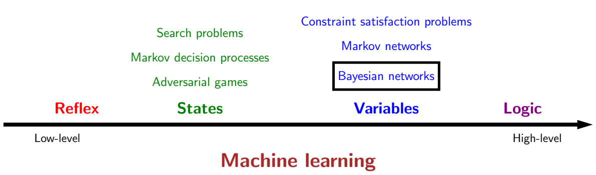
Bayesian networks: overview



- In this module, I'll introduce Bayesian networks, a new framework for modeling.

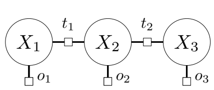
Course plan

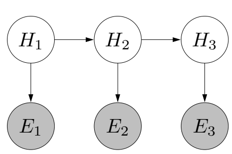


- We have talked about two types of variable-based models.
- In constraint satisfaction problems, the objective is to find the maximum weight assignment given a factor graph.
- In Markov networks, we use the factor graph to define a joint probability distribution over assignments and compute marginal probabilities.
- Now we will present Bayesian networks, where we still define a probability distribution using a factor graph, but the factors have special meaning.
- Bayesian networks were developed by Judea Pearl in the 1980s, and have evolved into the more general notion of generative modeling that we see today.

Markov networks versus Bayesian networks

Both define a joint probability distribution over assignments





Markov networks	Bayesian networks
arbitrary factors	local conditional probabilities
set of preferences	generative process

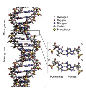
- Before defining Bayesian networks, it is helpful to compare and contrast Markov networks and Bayesian networks at a high-level.
- Both define a joint probability distribution over assignments, and in the end, both are backed by factor graphs.
- But the way each approaches modeling is different. In Markov networks, the factors can be arbitrary, so you should think about being able to write down an arbitrary set of preferences and constraints and just throw them in. In the object tracking example, we slap on observation and transition factors.
- Bayesian networks require the factors to be a bit more coordinated with each other. In particular, they should be local conditional probabilities, which we'll define in the next module.
- We should think about a Bayesian network as defining a generative process represented by a directed graph. In the object tracking example, we think of an object as moving from position H_{i-1} to position H_i and then yielding a noisy sensor reading E_i.

Applications

Topic modeling: unsupervised discovery of topics in text

Vision as inverse graphics: recover semantic description given image

Error correcting codes: recover data over a noisy channel



DNA matching: identify people based on relatives

- There are a huge number of applications of Bayesian networks, or more generally, generative models. One application is topic modeling, where the goal is to discover the hidden structure in a large collection of documents. For example, Latent Dirichlet Allocation (LDA) posits that each document can be described by a mixture of topics.
- Another application is a very different take on computer vision. Rather than modeling the bottom-up recognition using neural networks, which is the dominant paradigm today, we can encode the laws of physics into a graphics engine which can generate an image given a semantic description of an object. Computer vision is "just" the inverse problem: given an image, recover the hidden semantic information (e.g., objects, poses, etc.). While the "vision as inverse graphics" perspective hasn't been scaled up beyond restricted environemnts, the idea seems tantalizing.
- Switching gears, in a wireless or Ethernet network, nodes must send messages (a sequence of bits) to each other, but these bits can get corrupted along the way. The idea behind error correcting codes (Low-Density Parity Codes in particular) is that the sender also sends a set of random parity checks on the data bits. The receiver obtains a noisy version of the data and parity bits. A Bayesian network can then be defined to relate the original bits to the noisy bits, and the receiver can use inference (usually loopy belief propagation) to recover the original bits.
- The final application that we'll discuss is DNA matching. For example, Bonaparte is a software tool developed in the Netherlands that uses Bayesian networks to match DNA based on a candidate's family members. There are two use cases, the first one is controversial and the second one is grim. The first use case is in forensics: given DNA found at a crime site, even if the suspect's DNA is not in the database, one can match it against the family members of a suspect, where the Bayesian network is structured according to the family tree of the suspect and models the relationship between the family members's DNA using Mendelian inheritance. While this technology has been used to solve crime cases, there are some tricky ethical concerns about this expanded DNA matching, especially since an individual's decision to release their own DNA can impact the privacy of family members. The second use case is in disaster victim identification. After a big airplane crash (e.g., Malaysia Airlines flight MH17 in the Ukraine in 2014), a victim's DNA found at the crash site can be matched against their family members using the same mechanism above to identify the victim.

Why Bayesian networks?

- Handle **heterogenously** missing information, both at training and test time
- Incorporate **prior** knowledge (e.g., Mendelian inheritance, laws of physics)
- Can **interpret** all the intermediate variables
- Precursor to **causal** models (can do interventions and counterfactuals)

- These days, it's hard not to think about problems exclusively through the lens of standard supervised learning such as training a deep neural network on a pile of data.. Bayesian networks operate in a different paradigm which offers several advantages that are important to understand so that you can pick the right tool for the task.
- First, in traditional machine learning (e.g., linear models or neural networks), the input is usually of a fixed size (homogenous). With Bayesian networks, the types of inputs one can handle can be **hetereogenous** (e.g., missing features), both during training and test times.
- Second, Bayesian networks offer most leverage when you have rich **prior knowledge** (e.g., Mendelian inheritance, laws of physics). This allows one to often learn from very few samples and extrapolate beyond distribution of the training data. In contrast, deep neural networks generallly requires much more data to be effective.
- Third, because Bayesian networks are often carefully constructed based on prior knowledge, the variables in the Bayesian network are **interpretable** (more so that hidden units in a neural network), and you can ask questions about any of them via the laws of probability.
- Finally, Bayesian networks are an important precursor to developing **causal** models, which allow us to answer questions about interventions ("what would happen if we gave this drug to this patient?") and counterfactuals ("what would have happened if we had given this drug?"). These are extremely tricky and deep questions that standard machine learning or any methods that only view the world through prediction are unable to answer. For an easy introduction to some of these ideas, check out Judea Pearl's *The Book of Why*.
- Finally, Bayesian networks aren't suitable in every situation. In many vision, speech, and language problems, we have large datasets, mostly care about prediction, and it is extremely hard to incorporate prior knowledge about these very complex domains. In such cases, Bayesian networks have largely been supplanted with deep learning.

Roadmap

Modeling

Definitions

Probabilistic programming

Inference	**Learning**
Probabilistic inference	Supervised learning
Forward-backward	Smoothing
Particle filtering	EM algorithm

- In the remaining modules on Bayesian networks, I will first introduce a formal definition of Bayesian networks and explore some of its formal properties. Then I'll talk about **probabilistic programming**, a way to define Bayesian networks as (probabilistic) programs, which will provide a new perspective that allows to develop more powerful models.
- Then we turn to inference, which is what we do once we have a Bayesian network. We first define **probabilistic inference**, the problem of computing conditional and marginal probabilities and reduce this to the problem of inference in Markov networks. We then specialize to Hidden Markov Models (HMMs), an important special case of Bayesian networks, and show that the **forward-backward** algorithm can leverage the graph structure and do exact inference efficiently. Then we introduce particle filtering, which allows us to do approximate inference but scale up to HMMs where variables have larger domains.
- Finally, we talk about learning Bayesian networks from data. First we show how to do **supervised learning**, where all the variables are observed, which turns out to be very easy (just count and normalize). Then we show how to guard against overfitting in Bayesian networks by **smoothing**. Finally, we show how to do learning where some of the variables are unobserved using the **EM algorithm**.