Introduction to Bayesian Networks

Motivation
Last time we talked about the small-scale dynamics of regulatory networks, but our analysis was mostly mechanistic in nature. We were interested in analyzing the fluctuations that occur in a gene between a regulated gene being activated, and the gene reaching the steady-state response after the activation. Analyzing this region often requires parameters we don’t have, and it requires a level of detail we don’t want to get into. Specifically, the dynamic region probably only lasts for seconds or minutes, but we only generally sample on the order of hours and days.

We are interested today in the steady-state response of a gene to regulatory changes. When we make a change in Gene 1, which affects Gene 2, what change in the steady state response do we see in Gene 2? Expression data gives this steady state information. It helps to explain the relationships between expression of different genes. If the expression of two genes is not randomly distributed, there is probably a regulatory link between them. That is, expression data gives insight to the joint probabilistic distribution of the states of all of the genes we are analyzing. Our goal in this lecture is to model this joint distribution. This can be done using Bayesian networks.

Bayesian Network Preliminaries
Before we discuss the details of Bayesian Networks (BNs), it is important to discuss the structure and features that these networks use to capture useful relationships between genes. Here we abstract away from gene annotation, and treat genes as nodes in a graph.

Specifically, a BN uses a directed, acyclic graph (DAG) to capture information about dependencies between nodes. In this graph, an edge from node A to node B represents a causal and probabilistic dependence of B on A. Each node has associated with it a probability distribution conditioned on all its parents (if it has any), and it is these conditional distributions that allow the model to be adjusted. We can then use BNs to reason probabilistically about effects of perturbations to a system.

Understanding Probability Relationships in Bayesian Networks
Example 1: Icy Roads

![Graph example](Icy_Roads.png)
There is some casual impact associated with our knowledge of whether the roads are icy. This increases the chances of a crash.

Implications?

Say we learn that Watson has crashed. We can now say that Holmes is more likely to have crashed. This is because we now assign a higher probability to the roads being icy. If we already knew, however, that the roads are not icy, the information flow from Watson to Holmes is “blocked”, and the probability of Holmes crashing is unaffected by the evidence of Watson crashing.

This yields the following property of BNs:

*Conditional Independence: If we know nothing about ice, W and H are dependent. However, if we know about ice, W and H are conditionally independent.*

**Example 2: Rain and Sprinklers**

![Diagram of Rain and Sprinklers]

If Holmes discovers his lawn is wet, the chance of rain and sprinkler go up, because either could have caused the wetness. Because the chance of rain goes up, so too does the chance that Watson’s lawn is wet.

If we then find out that Watson’s lawn is indeed wet, the probability of rain goes up even more. In addition, the probability of the sprinkler being on falls from its elevated state back down towards its a priori, as Holmes’s lawn being wet is now explained away by rain.

This yields two more key properties of BNs:

1. *Explaining Away: Once we have more evidence for Rain, we have explained away the change in Holmes, and have less reason to increase the probability of Sprinkler.*
2. *Conditional Dependence: If we don’t know whether Watson’s lawn is wet, Rain and Sprinkler are independent. But if we do know that his lawn is wet, the explaining away principle makes these three nodes conditionally dependent.*

Being more concrete:

Independence: $P(X|Y) = P(X)$ or equivalently, $P(X,Y) = P(X)P(Y)$.

Conditional Independence: $P(X|Y,Z) = P(X|Z)$ or $P(X,Y|Z) = P(X|Z)P(Y|Z)$.

Independence does not imply conditional independence, nor does conditional independence imply independence!
**Graph Semantics**

In a DAG, there are three basic building blocks that we encounter in the graph structure. There are chains, in which Z depends on Y, which in turn depends on X. There are diverging relationships, where both X and Z depend on Y. Finally there are converging relationships, where Y depends on both X and Z. Each of these yields a different governing probability rule.

**Chain/Linear**

Relationship: X and Z are dependent if Y is unknown, but independent if Y is known (i.e. X and Z conditionally independent on Y).

**Diverging**

Relationship: Same as Chain/Linear

**Converging**

Relationship: X and Z are independent if Y is unknown, but dependent if Y is known (i.e. X and Z conditionally dependent on Y).

Generalizing these relationships, we come to the concept of d-separation:

**D-Separation:** Two nodes A and B are d-separated (or blocked) if for every path between A and B there is a node V such that:

1. The connection from V to A and B is diverging and V is known
2. The connection from V to A and B is converging and V (and all of its descendants) are unknown. If we know something about the descendants, this can translate up to provide some knowledge about V.

If A and B are not d-separated, they are d-connected.

**Network Equivalence**

It is interesting to note that two of the three patterns – chain and diverging – give rise to the conditional independence rule. We call these patterns equivalent in the probabilistic sense. Further, if we reverse the direction of a chain building block, we still preserve the probabilistic network relationship of the chain (and the chains are also equivalent), although causally they are opposites. It is important to remember the difference between probabilistic and causal equivalence moving forward.
**Bayesian Networks**

Finally we are ready to formally define a Bayesian Network. A BN consists of a network structure $S$ over a set of nodes $X$, and a set of conditional probability distributions (CPD) defined locally over every $X$. Theoretically we could use the conditional probability chain rule

$$P(A, B, C, D, \ldots) = P(A)P(B|A)P(C|A, B)P(D|A, B, C) \ldots$$

to define the probabilities over each node. However, this would be overkill, because for a given node $A$, it is likely that many of the other nodes in $X$ are independent of $A$. In fact, applying the probability rules we defined above, it becomes clear that the only nodes that $A$ depends on are its parents. Thus, the probability distribution over $A$ conditioned on every other node in $X$ is the same as the distribution conditioned only on the parents of $A$.

This greatly cuts down on the description length of the joint distribution over all nodes. In this sense, BNs are a *compact* way to describe a factorization of the distribution over the entire graph. On a side note, HMMs are an example of a Bayesian Network.

**Observational Inference and Junction Trees**

Now we are given a Bayesian Network, and want to use it. Specifically, by observing values of certain nodes in a set, we want to predict the state of other nodes.

Methods for observational inference:
- Exact Inference - Junction Tree (discussed below)
- Approximate Inference – Variational Approach, Monte Carlo, etc. (not discussed here)

Junction Tree Example:

We define clusters of nodes in our BN and link them by their common elements. Then, when we get a piece of information about a node:
1. Update the probability information in the cluster(s) that the node is in.
2. Calculate probabilities for the linking nodes from those cluster(s) to other clusters.
3. Transmit this probability to the other clusters.
In general this is the concept of *message passing* in Junction Trees. You have a set of nodes X and set of nodes Y and a separator S that stores the common nodes between the two. When Y is updated, it sends a message to X through the changes in S, so that X agrees with Y on the new value of S.

If we have a more complicated graph, we can follow some basic principles to ensure that all evidence is correctly propagated:
1. A node can send a message to a neighbor only after receiving all messages from all other neighbors.
2. When messages have passed both ways along a link, it is considered *consistent*.
3. Passing continues until all links are consistent.

**HUGIN Algorithm**
The HUGIN algorithm formalizes coordinated message passing. It is very simple, and runs as follows:
Select a node, V, as the root
1. Call `collectEvidence(V)`, which asks all neighbors to send message to V, or recursively pass message on to all neighbors but V
2. Call `distributeEvidence(V)`, which sends a message to all neighbors of V, and has them recursively send message to all neighbors but V

This will pull messages from all nodes, and then redistribute messages to all nodes. It should be noted that HUGIN may take more operations that necessary to make a BN correct, but it is guaranteed to make the BN correct in finite time.

**Bayesian Network/Junction Tree Conversion**
Converting BNs to Junction Trees is a complicated topic. The algorithm is roughly as follows:
1. Build a Moral Graph – Link all parents of nodes
2. Triangulate – add links so all cycles >3 have a cord
3. Cliques become nodes of the Junction Tree.
This topic is too complicated to be fully discussed here.
As a side note, a CRF is a Junction Tree.

**Applications**
1. We can use BNs to measure some gene expression and then predict the rest. Why would we not measure all gene expression? In some types of organisms, doing expression arrays is difficult because of noise, etc. Thus, you can do more expensive and accurate gene expression testing (using PCR) for a subset of the genes that is easier to measure, and use a BN to infer the effect on the rest of the genes in the regulatory network. You could also use regression to do this inference, but the key of BNs is that you can use other information to help structure the causal relationships between genes, where regression purely looks at the empirical data.

2. BNs can help to infer the effects of latent variables (variables that can be inferred but not observed directly). It is often the case that we know an immediate (parent) cause of a certain effect, but that this cause is itself the result of some other cause, etc. For this reason, it can be difficult (without using BNs) to predict the ultimate effects of an input such as a drug or environmental factor. With BNs however, as long as we can estimate a reasonable CPD for each variable in the system – latent and expressed – we can probabilistically infer the effects.
**Observation vs. Intervention**

It should be noted that we have talked completely about observation to this point. That is, we have not influenced the data in any way, only observed states and reported their effects to the network. However, if we intervene and force an action, we would have to change the network by cutting the edges from the influenced node to its parents, because the parents no longer control the node. For this to work, we need to make sure all parent-child edges are pointed in the right causal direction, even though reversing edges leaves the graph equivalent in the probabilistic sense by our earlier definition.

**Learning with Bayesian Networks**

Given a set of observations, can we build a BN including both topology $S$ and parameters $\Theta$ for probability distributions over $S$?

Given $S$, learning the probability distributions is easy.

$$\hat{\Theta} = \arg \max_{\theta} P(D|\Theta,S) = \prod_{i=1}^{n} P(X_i|parents(X_i), \Theta)$$

Or to include prior information $P(\Theta)$ in a Bayesian approach

$$P(\Theta|S,D) = P(S,D|\Theta)P(\Theta)$$

$$\theta_{\text{Bayes}} = \int \theta P(\Theta|S,D)d\Theta$$

Learning $S$ is more difficult (it is an NP-hard problem). We want to find an optimal structure $S$ given $D$. In some cases we can use Heuristic Search to maximize conditional probability of each $S$ given $D$. However, this has two problems:

1. The number of possible values for $S$ grows super-exponentially with number of nodes.
2. If we have sparse data over the nodes, it is easy to overfit to the data because there are only slight differences between the top results to the heuristic search.

To resolve the second issue, we could try to sample from an ensemble of models instead of just one, and then average the result over all of these samples. How to sample models is a topic for another lecture.

Finally, there are a number of inference and learning approaches to try to build BNs beyond Juncture Trees. For more information on these methods, talk to David Sontag, who is doing research in this area.