CROM3TOP
a Cross-species Multimodal Modular Model of Tolerance to Pathogens

Sahil Loomba

October 14, 2016

Abstract

Given that most bacteria are becoming resistant to antibiotics, there is an urgent need to flip our practice of diagnosing and treating pathogen infections. Instead of exterminating the pathogen, we must start looking at ways to make a host "tolerant" to the infection, that is, stay asymptomatic despite infection. For that, we look at the problem of predicting tolerance, and unearthing the underlying biological mechanisms of tolerance. We define a highly generalized and powerful Bayesian probabilistic framework called CROM3TOP (pronounced *'chrome-top'*), which can feed on (possibly temporal) multi-omics data across various host and pathogen species, to develop a rich ontology which not only differentiates between states of tolerance and sensitivity, but also provides a valuable interface for biologists to ask arbitrary queries of interest. This would help discover novel host-pathogen mechanisms, and hasten the design of gene therapies and other medical interventions.

Keywords: *tolerance to pathogens, multi-omics data, hybrid dynamic Bayesian networks, MCMC*

Contents

1 Introduction

As the use of antibiotics becomes increasingly prevalent, more and more strains of bacteria have started to show signs of resistance. In response to which even stronger antibiotics have been created, the evolutionary selective pressures of which have given rise to yet more resistant so-called 'superbugs'. Clearly, the path of fighting diseases no longer lies in trying to exterminate the pathogen. Rather, it resides in an alternative philosophy of modifying those host systems which the pathogen attacks. It is in this spirit that we aim to discover models of tolerance in hosts, especially in *Homo sapien*, to pathogens.

We define **tolerance** as the ability of a host infected by a pathogen to not develop symptoms of the disease caused by the pathogen (at possibly different pathogenic loads, or time instances from the point of infection). Discovering and incorporating markers of tolerance allows us to stay asymptomatic, despite being exposed to the disease-causing pathogen. An **ideal model** of tolerance must have the following desirable characteristics. It must be:

Predictive Given sufficient data about the host and pathogen, it should be able to predict whether the host will be tolerant or sensitive to the pathogen.

Informative The model must be rich enough to provide insightful information on critical pathways, which can help design better drugs and gene therapies.

Elegant It should be easy to make sense of the model in the biological domain.

Complete Any arbitrary queries on the host-pathogen interaction must be answerable, even in the face of incomplete data.

Efficient Making inferences from the model must be time (and space) efficient.

Up until now, clinical or physiological data and indirect biomarkers extracted from the blood, have been the 'source data' of choice for medical practitioners to diagnose and treat a disease. Although such data might suffice for a superficial detection of (a family of) diseases, when we wish to inquire about more specific information, such as that related to tolerance, high-level data fall short of providing meaningful answers. Ideally, the kind of data which would best predict tolerance is that on specific biomolecular flows and reactions in the host-response pathway. However, there are many different ways (spanning a range of genes, proteins and metabolites) in which a host might react to an infection, which are yet to be entirely deciphered, yet alone understood. We must, therefore, rely on proxy sources such as **omics data** (transcriptomics, proteomics, metabolomics), data on the **microbiome** of the organism, and some clinical data, to approximate the true mechanism of tolerance. Since *Homo sapien* eventually is the most important host of our study, it would be a good idea to collect our data from humans. However, it is naturally easier and convenient to perform infection studies on lower organisms. Additionally, human biological systems tend to be more complicated and lesser understood than those of other species. Therefore, collecting such data across multiple host species, and tying them together through known homologies, should paint a more comprehensive and complete picture of tolerance.

Furthermore, we inform our models not just by the usual training samples of hosts with known tolerance/sensitivity, but also by a rich hierarchy of relationships between biomolecules, such as gene regulatory interactions from GRNs, protein interactions from PPIs, metabolic and signaling pathways, among others. Most of these publicly available interactions have been curated and validated from experiments and other computational studies, and each of them can serve as distinct but related modalities to improve the model fit.

In the following sections, we describe a probabilistic graphical model which fits the template of tolerance. We define the model structure, methods of learning and making inferences from it, concluding with possible challenges and limitations.

2 Overview of Probabilistic Modeling of Tolerance

More often than not, it is natural to see a variable x of a non-deterministic system as a random variable, that follows some probability distribution $P(x)$. If we know $P(x)$, we can answer a lot questions about the behaviour of x, like its mean (or 'most expected value'). Most systems have more than one variable though, let us call them $\{x_1, x_2, \cdots, x_n\}$, and we can now define a joint distribution over these variables as $P(x_1, x_2, \cdot, sx_n)$. Once we know P, we can ask almost any arbitrary query on the variables by finding the appropriate marginal probability distribution. Say the variables are discrete, then if we are interested in the joint distribution of $\hat{X} = \{x_2, x_5, x_9\}$ we find it by summing over all other variables

$$P(x_2, x_5, x_9) = \sum_{x_i \notin \hat{X}} P(x_1, x_2, \cdots, x_n) \tag{1}$$

Similarly for continuous variables, the sum simply gets replaced by an integral, and the probability distribution P by a probability density function p. Let us assume for the sake of discussion that these variables are discrete, and can take m possible values. Therefore, the size of our probability distribution P becomes $m^n - 1$. Now clearly, the larger is the size of P, the more will be the number of summations and thus the longer is the time taken for inference. Not only that, but learning larger distributions naturally requires more data for a good fit. Can we then somehow reduce the size of the distribution? This is where the notion of a probabilistic graphical model comes in. We consider the following. Given two random variables a and b, we can write their joint distribution as

$$P(a, b) = P(a)P(b|a) = P(b)P(a|b)$$

Now, a and b are said to be independent random variables iff

$$P(a|b) = P(a)$$

This allows us to write a simpler formula for the joint distribution, simply as the product of the marginal probabilities, as

$$P(a, b) = P(a)P(b)$$

The right hand side of above equation is referred to as the factored form of joint distribution P. Notice that although the joint probability has a size of $m^n - 1$, the factored form is a product of n marginal distributions of size $m-1$ each. Therefore, for n variables that can take m possible values each, if all are independent of each other, then we get a best-case reduction in the size of distribution (or 'number of parameters' of the distribution) from exponential m^n to linear mn. This allows the probabilistic model to be computationally tractable.

However, it is very rare to find that all variables are independent of one another. But, with the help of domain knowledge, once can adjudge many *conditional independencies* between variables. A random variable a may not be independent from a random variable b, but they could be independent of each other given a third random variable c. This can be mathematically written as

$$a \perp b|c$$

and the distribution can be written as

$$P(a, b|c) = P(a|c)P(b|c) \tag{2}$$

For example, *sunny* and *humid* may not be independent of each other, but given that it is *rainy*, they are independent of each other. This could be equivalently interpreted as *rainy* influencing *sunny*, and *rainy* influencing *humid*, but no relationship between *sunny* and *humid*. Such relationships between random variables are best encoded in a graphical representation, such as that in Figure 1. This directed (acyclic) graph is called a **Bayesian Network**, where the nodes of the graph are random variables, and an edge $a \to b$ can be interpreted as an edge of directed influence or causality. If $Pa(x_i)$ denotes parents and $NonDe(X_i)$ denote the non-descendants of node x_i, then the conditional independencies represented by the graph can be mathematically written as

$$x_i \perp NonDe(x_i)|Pa(x_i)$$

That is given its parents, a node is independent of its non-descendants. Now, we can write the factored form of joint distribution as

$$P(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i)) \tag{3}$$

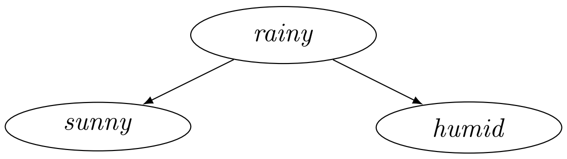

Figure 1: A simple Bayesian network encoding conditional independencies

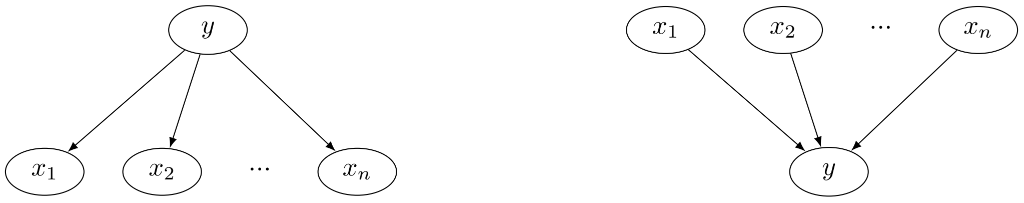

Figure 2: Examples of 'good' and 'bad' network dependencies

If a node has maximum of q parents, then the number of parameters get reduced to no more than $n(m^{q+1} - 1)$. The smaller is q, the more can the dependency structure be exploited to learn from data. For instance, in Figure 2, the first structure is better than the second. **This must be taken care of while designing the network structure.**

2.1 Multimodality: towards Bayesian Networks

We've seen how a Bayesian network can be a very compact representation of our feature space. Let us now formally describe how we can define the structure of this representation for the problem at hand, which is to find a model of tolerance. We define tolerance as a random variable $\lambda \in \{0, 1\}$, and the manifestation of it in the form of various biological modalities $A = \{a_1, a_2, \cdots, a_n\} \in \mathbb{R}^n$. For instance, an infection can cause over/underexpression of some proteins like signalling molecules in the cell, which can induce regulation of certain genes that in turn release proteins essential to an immunity response. Moreover, the microbiome of the host organism can be directly affected by the pathogen infection, and also by immune proteins. Together, the two can affect some key biochemical reactions which occur in the body, therefore changing metabolite concentrations. Another important modality, often used as a proxy for medical diagnosis of tolerance, are the vitals of the organism like heartbeat and oxygen in the blood. (See Figure 3 for a succinct picture.) In terms of data modeling, we call these modalities the *features* of our model. Thus in all, we have a rich multimodal dataset $\mathcal{D} = \{\lambda_i, A_i\}_{i=1}^m$ collected from m infected hosts. This appears exactly in the template of a typical classification problem, where we are trying to find a map $f : \mathcal{A} \to \lambda$. In a probabilistic paradigm, one popular technique for classification is the naive Bayes model ($\mathcal{B}_0$), whose structure is depicted in Figure 4. Clearly, tolerance influences the 'levels' of all biological modalities, which makes intuitive sense. Using Equation 3, the joint distribution can be written as

$$P(\lambda, a_1, a_2, \cdots, a_n) = P(\lambda) \prod_{i=1}^{n} P(a_i|\lambda) \tag{4}$$

However, what we wish to find host tolerance given modality data, or $P(\lambda|\mathcal{D})$. We can do that using **Bayes' Rule**

$$\begin{aligned} P(\lambda, \mathcal{D}) &= P(\lambda)P(\mathcal{D}|\lambda) = P(\mathcal{D})P(\lambda|\mathcal{D}) \\ \Rightarrow P(\lambda|\mathcal{D}) &= \frac{P(\lambda)P(\mathcal{D}|\lambda)}{P(\mathcal{D})} \\ \Rightarrow P(\lambda|\mathcal{D}) &= \frac{P(\lambda)P(\mathcal{D}|\lambda)}{\sum_\lambda P(\lambda, \mathcal{D})} \\ \Rightarrow P(\lambda|\mathcal{D}) &= \frac{P(\lambda)P(\mathcal{D}|\lambda)}{\sum_\lambda P(\lambda)P(\mathcal{D}|\lambda)} \end{aligned} \tag{5}$$

Wherein $P(\lambda)$ is called the **prior** probability (of tolerance), $P(\mathcal{D}|\lambda)$ is called the **likelihood** and $P(\lambda|\mathcal{D})$ is called the **posterior** probability[1]. Therefore given the right-hand-side of Equation 5, we can easily infer the tolerance of a new host. However, note the conditional independencies encoded by this

$$\begin{tikzcd}
 & \boxed{infection} \arrow[dl] \arrow[d] \arrow[dr, dashed] & \\
proteins \arrow[r] \arrow[d, leftrightarrow] \arrow[dr] & microbiome \arrow[d] & vitals \\
genes & metabolites &
\end{tikzcd}$$

Figure 3: A high-level cartoon of relationships between multiple biological modalities, which can be affected by a pathogenic infection

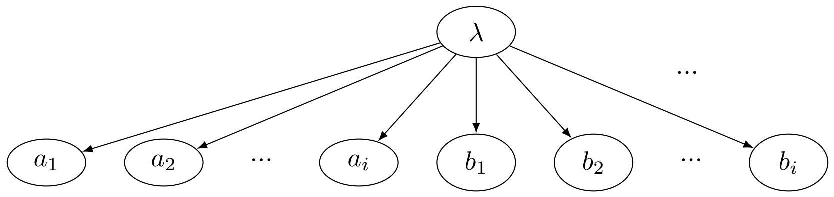

Figure 4: A naive Bayesian representation $\mathcal{B}_0$ for the tolerance model

structure. Given the state of tolerance λ, every biological modality is independent of each other, which is in gross violation of all our biological understanding.

Let us use some domain knowledge to produce a biological interaction network, which better encodes the senses of influence and independence, like in Figure 5. Note that one key requirement of this network is that it should be a directed acyclic graph, or DAG. Let us now append other key variables of our model: λ, and we introduce variables for the pathogen, namely the pathogen type η which belongs to the discrete space of pathogens, and the amount of viral load $\xi \in \mathbb{R}$. One could assume that each of these would directly affect every biological modality. We know end up with a Bayesian network graph $\mathcal{B}_1$ which looks like that in Figure 6.

2.2 Crossmodality: towards Transfer Learning through Hierarchical Bayes

We have already considered multiple species of pathogen which could infect the host, against whom the model of tolerance is being built. However, it would also be interesting to model multiple host organisms together, so that their individual models could 'learn more' from across each other through shared representations. This subfield of learning across domains is called **transfer learning** in machine learning literature[3]. Say we have K host types, each with their own set of biological modalities A_k. Now some of these modalities are (functionally, if not physically) common across hosts, particularly all metabolites, some (homologous) proteins, and few (homologous) genes. Let $A_{shared} = \bigcap_k A_k$ be the set of common modalities, and $A_{all} = \bigcup_k A_k$ be the set of all modalities. For every host, we define an extended feature set such that the feature space for all hosts spans A_{all}. Say we have m samples, then for every sample corresponding to the k^{th} host, there are in some sense $|A_k - A_{shared}|$ number of incomplete data values in this space. On the other hand, every sample of every host has complete values for $|A_{shared}|$ number of features.

For this space, we can define a Bayesian network representation as shown in Figure 7. The parameter θ describes the host species, and directly influences all the biological modalities. Notice that we have actually defined here a hierarchical Bayesian model, with a parameter $\boldsymbol{\rho}$ that defines the probability distribution of θ, and hyperparameters $\boldsymbol{\alpha}, k$ that define the distribution of itself. This permits a deeper sense of parameter tying, and induces more knowledge transfer across domains.

2.3 Temporality: towards Dynamic Bayesian Networks

More often than not, tolerance is not an absolute state of the organism, and evolves with time. Say when hosts get infected by a pathogen, they all may show signs of tolerance for a few hours. But then, things begin to diverge when some hosts suddenly become sensitive to the infection and develop symptoms of the disease, while others don't. Therefore to create a full model of tolerance, it is crucial to make the model temporal. Fortunately, it is straightforward to extend Bayesian networks to this setting, using

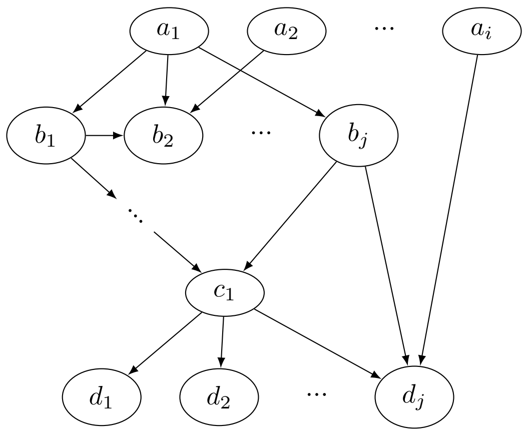

Figure 5: A biological interaction network, across various modalities, represented as a directed acyclic graph (DAG) where a node could represent any of the biological entities mentioned in Figure 3, as long as the graph has no self-loops and cycles

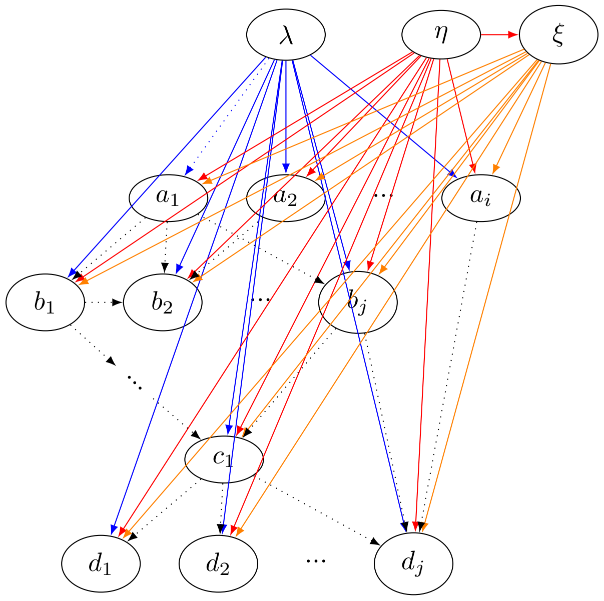

Figure 6: **Incorporating multimodality** The biological interaction network is represented as a Bayesian network $\mathcal{B}_1$, with λ being a random variable for tolerance, η that of pathogen type, and ξ that of pathogen load, with the assumption that they directly 'influence' every node in the directed interaction network

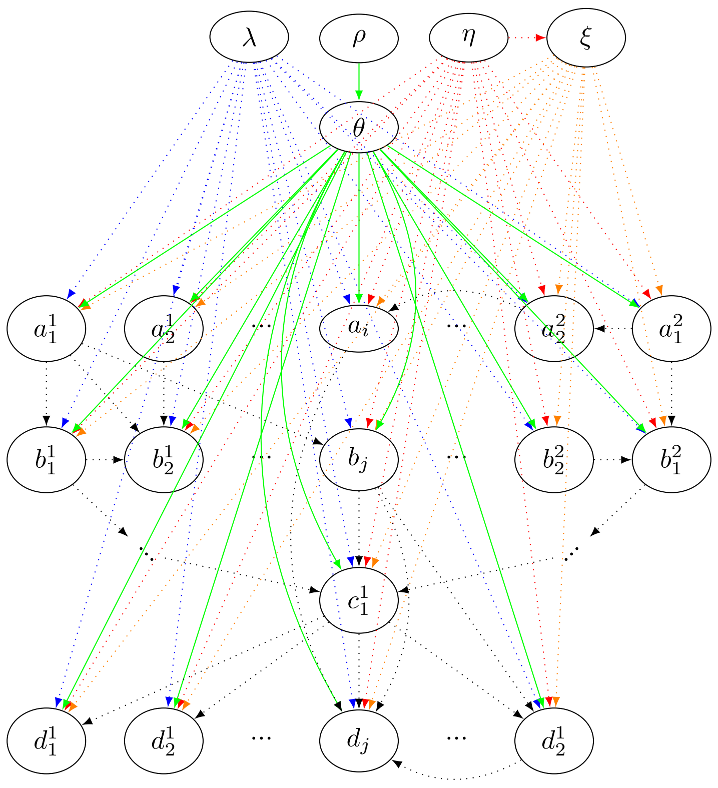

Figure 7: **Incorporating crossmodality** A Bayesian network representation $\mathcal{B}_2$ spanning multiple host species, with the random variable θ indicating host species, and a hyperparameter ρ over θ. Note that some biological nodes (a_i, b_j, c_1^1, d_j) are shared by the hosts, which can be used to induce a transfer of learning across multiple species.

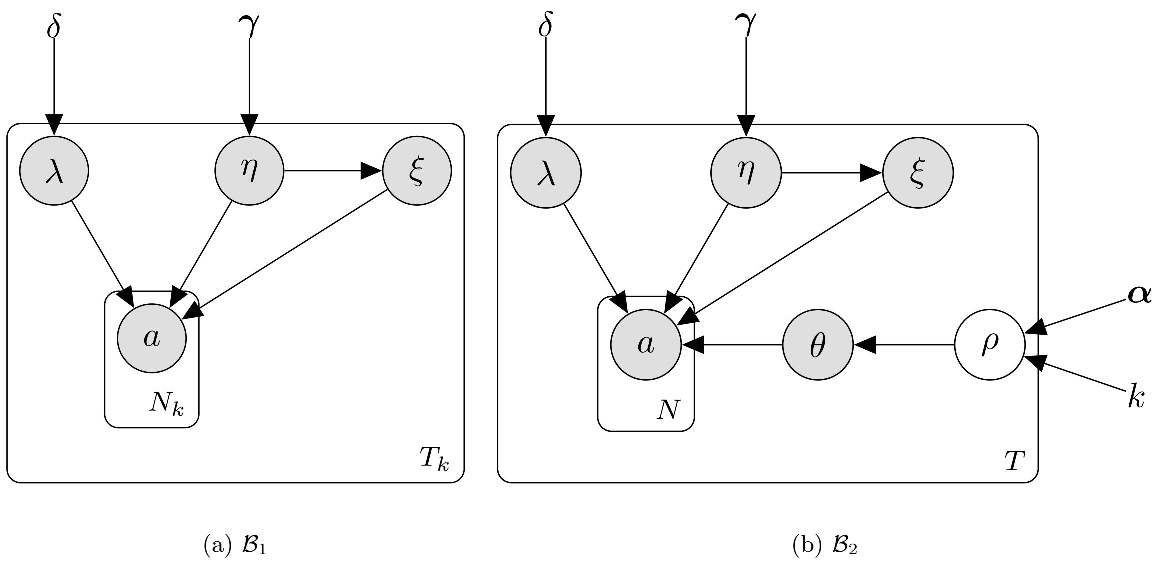

Figure 8: Plate notation to concisely describe the dynamic Bayesian models, with occluded biological interactions amongst a, and persistence variables: λ, a

dynamic Bayesian networks or DBNs. Essentially, a DBN encodes not only the standard equitemporal conditional independences, but also intertemporal ones.

Now, every variable X is actually a template variable X^t, whose value is instantiated at some timestamp. Consider a distribution over a 'trajectory' of the template random variables in feature space X.

$$P(X^{0:T}) = P(X^0) \prod_{t=0}^{T-1} P(X^{t+1}|X^{0:t})$$

This distribution can get very complex for a large T. We can compact this representation by making a reasonable assumption, that given the present state, the future is independent of the past. This is called the **Markovian assumption**. Given it holds, we can write the above equation as

$$P(X^{0:T}) = P(X^0) \prod_{t=0}^{T-1} P(X^{t+1}|X^t)$$

But what if T is undefined, possibly even infinite? We can still come up with a compact representation if another assumption of the 'Markov chain' holds true, that it is **stationary or time-invariant**. This assumption says that $\forall t, P(X^{t+1}|X^t)$ is the same. We can thus define a transition model $P(X'|X)$, which along with the initial probability distribution $P(X^0)$, completely describe the entire model. Interface variables X_I are those whose value at time t have a direct effect on variables at time $t{+}1$. For our tolerance model, it makes sense to include the tolerance state as an interface variable. We could also include every biological modality in this set, although that would induce many dense connections, which looses out on the compactness of our representation. Let us further (safely) assume that every biological modality, and tolerance variable, influences only its own future state directly. Such variables are called persistent variables. Therefore, we define all our interface variables to be persistent, to capture trajectories very succinctly. The final Bayesian representations have been drawn in Figure 8.

2.4 Modularity: towards Structure Learning in Bayesian Networks

Up until now, we have assumed that the state of tolerance and pathogen, which we can phenomenally term as the "infection", affects every biological modality directly. However we know that this is not entirely true. A pathogen infection is responded to often in stages, by different sets or **modules** of biological entities (modalities here). Most curated biological databases give us a an idea of how the modalities are interrelated, but not of how they group together during host response. Therefore, we also incorporate a computational approach to automatically learn proxy-modules which, by definition, improve the predictive power of our model. These may not have a one-to-one correspondence to real biological subsystems, but this can only be (un)confirmed in hindsight of employing the model.

We hypothesize a layer of hidden variables $H = \{h_1, h_2, \cdots, h_{|H|}\}$ sandwiched by the "pathogen infection" variables (λ, η, ξ) and "host response" variables $(\theta, A = \{a_1, a_2, \cdots, a_{|A|}\})$. See Figure 9 for the Bayeisan network structure. The introduction of hidden variables can greatly simplify the structure, reducing complexity of the network which needs to be learnt[2]. For example, if we quantify the number of modality parameters involved in the non-modular $\mathcal{B}_2$ versus modular $\mathcal{B}_3$ Bayesian representations (see Table 1), they are as follows:

$$\begin{aligned} |\phi_{\mathcal{B}_2}| &= \sum_{a \in A} \big((Pa(a) + 2).(|\theta||\eta||\lambda| - 1) + 1 \big) \\ |\phi_{\mathcal{B}_3}| &= \sum_{a \in A} \big((Pa(a) + 2).(|\theta| - 1) + 1 \big) + \sum_{h \in H} \big(2.(|\eta||\lambda| - 1) + 1 \big) \end{aligned} \tag{6}$$

By a rough approximation, especially if $|H| \ll |A|$ we find that:

$$|\phi_{\mathcal{B}_2}| \approx \frac{|\eta||\lambda|}{1 + \dfrac{2}{2 + \overline{Pa(a)}} \dfrac{|\eta||\lambda|}{|\theta|} \dfrac{|H|}{|A|}} |\phi_{\mathcal{B}_3}| \approx |\eta||\lambda||\phi_{\mathcal{B}_3}| \tag{7}$$

Therefore it might be a good idea to go ahead with this modular design. However, how do we define the number of hidden variables, and to which biological modalities they are connected? This can be achieved through another step of learning, wherein the graph structure $\mathcal{G}$ of the Bayesian network itself is variable. If we can define a space of structures, then one could write something similar to Equation 5 and find

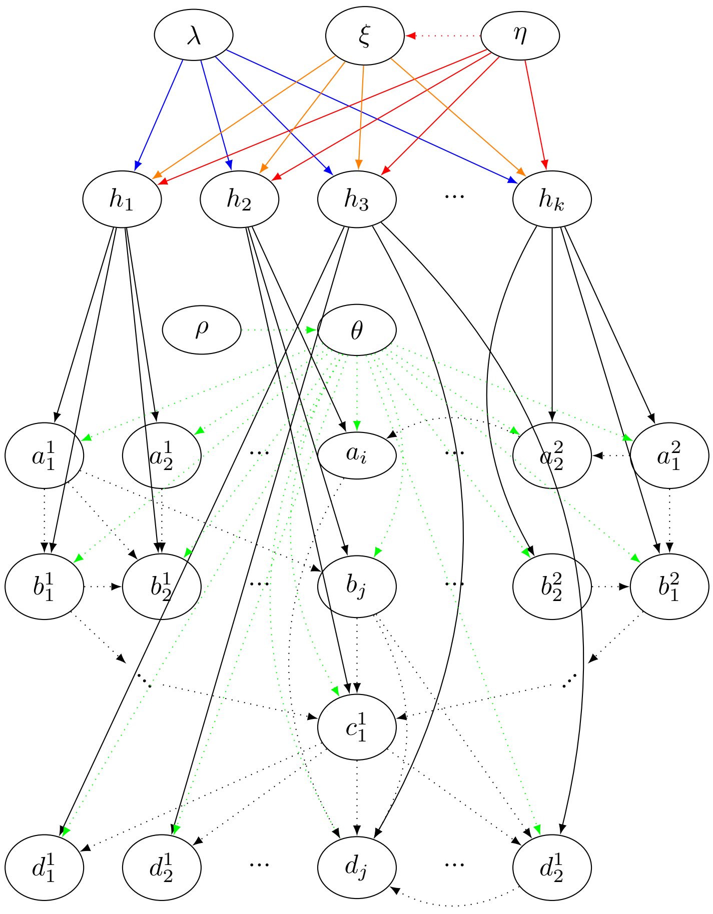

Figure 9: **Incorporating modularity** This Bayesian network representation $\mathcal{B}_3$ introduces a hidden variable layer between the pathogen and host variables. Note that every biological modality has a single hidden modality parent.

the "best" graph structure $\hat{\mathcal{G}}$ by taking a MAP (Maximum a Posteriori) estimate in this meta-Bayesian problem:

$$\begin{aligned} P(\mathcal{G}|\mathcal{D}) &= \frac{P(\mathcal{G})P(\mathcal{D}|\mathcal{G})}{P(\mathcal{D})} \\ \hat{\mathcal{G}} &= \arg\max_{\mathcal{G}} P(\mathcal{G}|\mathcal{D}) \end{aligned} \tag{8}$$

3 Pragmatics of the Bayesian Network Model

We now formally describe some key ideas and algorithms for pragmatically employing the model described above, spanning representation and parametrization, parameter learning, and (approximate) inference.

3.1 Representation and Parametrization

3.1.1 Independencies encoded by a Bayesian Network

As described above, Bayesian networks are a compact representation of the joint probability distribution P by directly encoding its conditional independencies which allows us to decompose the distribution locally at every node, conditioned on its parents. Let us denote the BN graph structure by $\mathcal{G}$. We define $\mathcal{I}(P)$ to be the set of all independence assertions of the form $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})$ that hold in P. A graph object $\mathcal{K}$ with a set of independencies $\mathcal{I}(\mathcal{K})$ is said to be an **I-map** for a set of independencies $\mathcal{I}$ if $\mathcal{I}(\mathcal{K}) \subseteq \mathcal{I}$. Thus, we say that $\mathcal{G}$ is an I-map for P if it is an I-map for $\mathcal{I}(P)$. If $\mathcal{G}$ is over variable space $\mathcal{X} = \{X_1, \cdots, X_n\}$ then the distribution P over $\mathcal{X}$ is said to **factorize** over $\mathcal{G}$ if we can express it as a product of individual factors called conditional probability distributions:

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i|Pa^{\mathcal{G}}(X_i)) \tag{9}$$

Thus, a Bayesian network is a pair $\mathcal{B} = (\mathcal{G}, P)$ where P factorizes over $\mathcal{G}$. Also, if $\mathcal{G}$ is an I-map for P, then P factorizes over $\mathcal{G}$. The converse holds true as well[2]. The resulting factored representation can be substantially more compact, especially for sparse structures with small indegrees. Now clearly, there are many possible structures that are consistent with the same set of independencies (since we can have various $\mathcal{G}'$ such that $\mathcal{I}(\mathcal{G}') \subseteq \mathcal{I}(P)$, particularly a fully connected graph being an I-map for every P). It is practically a good idea to choose structures that reflect some causal order, where causes are parents of effects, since there is a local influence in the real world and because causal graphs tend to be sparser.

Note that the set of "local" independencies $\mathcal{I}_l(\mathcal{G})$ described above are not the only ones which can be read off of $\mathcal{G}$, and there are some "global" independencies as well. For that, it is useful to view probabilistic influence as a flow in the graph, and we try to analyze when influence can flow from X through Z to affect our beliefs about Y. We define a trail to be a walk from a node to another, irrespective of the direction of edges, but without any loops. Say $X_1 \rightleftharpoons \cdots \rightleftharpoons X_n$ be a trail in $\mathcal{G}$, and $\mathbf{Z}$ a subset of observed variables. The trail $X_1 \rightleftharpoons \cdots \rightleftharpoons X_n$ is active given $\mathbf{Z}$ if whenever there is a "v-structure" $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ then X_i or one of its descendants are in $\mathbf{Z}$ and no other node along the trail is in $\mathbf{Z}$. Now, let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three sets of nodes in $\mathcal{G}$. We say that $\mathbf{X}$ and $\mathbf{Y}$ are **d-separated** given $\mathbf{Z}$ if there is no active trail between any node $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given $\mathbf{Z}$. This set of independencies can be written as:

$$\mathcal{I}(\mathcal{G}) = \{(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) : d-sep_{\mathcal{G}}(\mathbf{X};\mathbf{Y}|\mathbf{Z})\} \tag{10}$$

We also have that if P factorizes over $\mathcal{G}$, then $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$. That is, if X and Y are *not d-separated* given $\mathbf{Z}$ in $\mathcal{G}$, then X and Y are *dependent* in all distributions P that factorize over $\mathcal{G}$. This notion of d-separation is key to infer independence properties of a distribution P that factorizes over $\mathcal{G}$ by simply examining the connectivities in the latter. Alternately, if we know these independencies in P, then it can help us pick a better structure $\mathcal{G}$. Now, many structures can encode the same set of independencies $\mathcal{I}(\mathcal{G})$, which makes them **I-equivalent**, like in Figure 10. A theorem which captures this concisely says that for two structures $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathcal{X}$, if they have the same skeleton (the underlying structure with dropped directionality of edges) and the same set of v-structures, then they are I-equivalent. Thus, we could choose any of these I-maps as possible structures for P. However, going ahead with the sparsity intuition, we define the notion of a **minimal I-map**. A graph $\mathcal{K}$ is a minimal I-map for P if it is an I-map for P and if the removal of even a single edge from $\mathcal{K}$ renders it not an I-map. We can now devise a procedure for creating an I-map. Given a determined (topological) variable ordering $\{X_1, \ldots, X_n\}$, for each X_i select a minimal subset $\mathbf{U}$ of $\{X_1, \ldots, x_{i-1}\}$ to be parents of X_i in $\mathcal{G}$, such that

$$X_i \perp \{X_1, \ldots, X_{i-1}\} - \mathbf{U}|\mathbf{U} \tag{11}$$

Although in our model of tolerance, we already pick up the graph structure from curated biological interaction networks, this notion of minimal I-maps can be used to validate and prune those structures further. However, one challenge which remains is that of extracting dependencies of the kind in Expression 11 from P. We come back to this after describing our parametrization of the joint distribution.

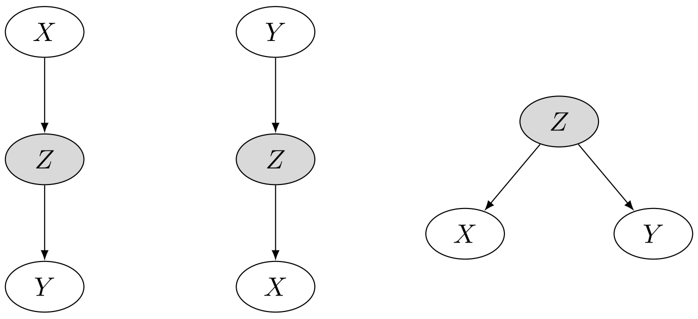

Figure 10: Three I-equivalent graph structures for the conditional independency $X \perp Y|Z$

Variable	Description	Discrete	Distribution
λ	Tolerance	Yes	$\lambda\|\delta \sim Bernoulli(1,\delta)$
η	Pathogen type	Yes	$\eta\|\boldsymbol{\gamma} \sim Categorical(\boldsymbol{\gamma})$
ξ	Pathogen load	No	$\xi\|\eta \sim \mathcal{N}(\beta_\eta,\sigma^2)$
θ	Host	Yes	$\theta\|\boldsymbol{\rho} \sim Categorical(\boldsymbol{\rho})$
$\boldsymbol{\rho}$	Parameter for θ	No	$\boldsymbol{\rho}\|\boldsymbol{\alpha},k \sim Dirichlet(\boldsymbol{\alpha},k)$
For Non-modular Bayesian Model $\mathcal{B}_2$			
a_i	Biological modality	No	$a_i\|\lambda,\eta,\theta,\xi,Pa(a_i) \sim \mathcal{N}(\beta^0_{\lambda\eta\theta}+\beta^{\xi}_{\lambda\eta\theta}\xi+\beta^1_{\lambda\eta\theta}b_1+\dots+\beta^j_{\lambda\eta\theta}b_j,\sigma^2)$
For Modular Bayesian Model $\mathcal{B}_3$			
h_i	Latent modality	No	$h_i\|\lambda,\eta,\xi \sim \mathcal{N}(\beta^0_{\lambda\eta}+\beta^{\xi}_{\lambda\eta}\xi,\sigma^2)$
a_i	Biological modality	No	$a_i\|\theta,h_{a_i},Pa(a_i) \sim \mathcal{N}(\beta^0_{\theta}+\beta^{h_{a_i}}_{\theta}h_{a_i}+\beta^1_{\theta}b_1+\dots+\beta^j_{\theta}b_j,\sigma^2)$

Table 1: Probability distributions for variables of the (atemporal) Bayesian model

3.1.2 Parametrizing the Probabilistic Model

Table 1 provides a succinct overview of the variables and parameters of the model, and the hypothesized distributions they follow. Note that all continuous data is normalized to zero mean and unit variance. We elaborate on each of them below:

- **Tolerance**, which is the primary variable of interest, is a Boolean variable $\lambda \in \{0,1\}$ following a Bernoulli distribution parametrized by $\delta \in [0,1]$

$$P(\lambda = 1) = \delta$$

- **Pathogen type** is a discrete variable $\eta \in \{p_1,\dots,p_{|\eta|}\}$ where p_k is some pathogen species like *E. Coli*, following a Categorical distribution parametrized by $\boldsymbol{\gamma} = \{\gamma_1,\dots,\gamma_{|\eta|}\}$ where $\forall k : \gamma_k \in [0,1]$ and $\sum_{k=1}^{|\eta|}\gamma_k = 1$

$$P(\eta = p_k) = \gamma_k$$

- **Pathogen load** is a continuous variable $\xi \in \mathbb{R}$ following a univariate Normal distribution parametrized by $\beta_\eta \in \mathbb{R}$ (given η) and $\sigma \in \mathbb{R}$

$$p(\xi = x) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\beta_\eta)^2}{2\sigma^2}}$$

- **Host** is a discrete variable $\theta \in \{s_1,\dots,s_{|\theta|}\}$ where s_k is some host species like *Homo sapiens*, following a Categorical distribution parametrized by $\boldsymbol{\rho} = \{\rho_1,\dots,\rho_{|\theta|}\}$ where $\forall k : \rho_k \in [0,1]$ and $\sum_{k=1}^{|\eta|}\rho_k = 1$

$$P(\theta = s_k) = \rho_k$$

- $\boldsymbol{\rho}$, the parameter for θ, is a multivariate continuous variable following the Dirichlet distribution parametrized by $\boldsymbol{\alpha} \in \mathbb{R}^{+k}$ and $k = |\theta|$ where $k \geq 2$

$$p(\boldsymbol{\rho} = \boldsymbol{x}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{i=1}^{k}x_i^{\alpha_i - 1}$$

Note that choosing different αs for every host species parameter can be done with another hyperparametrization to, say, a uniform distribution which might induce further parameter tying. However, we restrict to the simpler symmetric Dirichlet distribution, where the same α is selected and used.

- **Modalities**, latent or biological, are continuous variables $a_i \in \mathbb{R}$ which are a linear Gaussian of their continuous parent variables $Pa^C(a_i)$ conditioned on their discrete parent variables $Pa^D(a_i)$, parametrized by $\beta^0 \in \mathbb{R}, \{\beta^j \in \mathbb{R}\}_{j=1}^{|Pa^C(a_i)|}$ and $\sigma \in \mathbb{R}$

$$p(a_i = x|Pa^C(a_i), Pa^D(a_i)) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{\left\{ x - \left(\beta^0 + \sum_{j=1}^{|Pa^C(a_i)|} \beta^j Pa_j^C(a_i) \right) \right\}^2}{2\sigma^2} \right]$$

3.1.3 Gaussian Bayesian Network

Although Gaussian is a strict assumption on the continuous variables (pathogen load, latent modalities & biological modalities) and we can choose a more generic distribution from the family of exponentials like the Gamma distribution, we employ it primarily due to two reasons. First, it often approximates real datasets sufficiently well, especially considering that we are most interested in "mean shifts" of biological modalities during host-pathogen interactions. And second, because the mathematics for a Gaussian BN works out elegantly, with the number of parameters being as few as quadratic in the number of variables, making learning and inference simpler and more efficient in a complex setting such as ours.

In the hybrid Bayesian setting, wherein we have both discrete as well as continuous variables, one can imagine an assignment to the discrete variables as a "switching" of the BN to a particular state, within which we now have the continuous variables that vary. Consider one such assigned state of the BN, wherein we can ignore the discrete variables for now and focus on the network of continuous variables. Since all of them follow the linear Gaussian distribution, we call this a **Gaussian Bayesian Network** $\mathcal{B}_{\mathcal{N}}$ with structure $\mathcal{G}_{\mathcal{N}}$.

For Y which is a linear Gaussian of its parents $X_1, \dots, X_k$, we have $p(Y|\mathbf{X}) = \mathcal{N}(\beta_0 + \boldsymbol{\beta}^T\mathbf{X}; \sigma^2)$. Assuming that $\mathbf{X}$ is a joint Gaussian, that is, $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then the distribution $p(Y)$ is normal where $Y \sim \mathcal{N}(\beta_0 + \boldsymbol{\beta}^T\boldsymbol{\mu}, \sigma^2 + \boldsymbol{\beta}^T\Sigma\boldsymbol{\beta})$, and the joint distribution $p(\mathbf{X}, Y)$ is normal with $Cov(X_i, Y) = \sum_{j=1}^k \beta_j \Sigma_{ij}$. From here, it's straightforward to induce that a linear Gaussian BN defines a joint distribution over the entire space that is a multivariate Gaussian. (Note that while the distribution over continuous variables *given* an assignment to discrete ones is a multivariate Gaussian, in general it is simply a mixture of multivariate Gaussians, with the number of mixture components being exponential in number of discrete variables weighted by their respective prior probabilities.)

It makes sense to look more carefully at a multivariate Normal distribution over $X_1, \dots, X_n$ as $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector and $\Sigma \in \mathbb{R}^{n \times n}$ is the symmetric covariance matrix that is positive definite ($\forall \boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x} \neq \mathbf{0}$, we have that $\boldsymbol{x}^T\Sigma\boldsymbol{x} > 0$ for a well-defined density that integrates to 1)

$$p(\mathbf{X} = \mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right]$$

There as an alternate representation for the multivariate Gaussian, which expresses inverse of the covariance matrix as the "information matrix" J. The exponent term now becomes

$$\begin{aligned} -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) &= -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T J (\boldsymbol{x} - \boldsymbol{\mu}) \\ &= -\frac{1}{2}(\boldsymbol{x}^T J \boldsymbol{x} - 2\boldsymbol{x}^T J \boldsymbol{\mu} + \boldsymbol{\mu}^T J \boldsymbol{\mu}) \\ &\propto -\frac{1}{2}\boldsymbol{x}^T J \boldsymbol{x} + (J\boldsymbol{\mu})^T \boldsymbol{x} \end{aligned} \tag{12}$$

Now we are interested in primarily two kinds of operations for the purpose of learning and inference: marginalizing out a distribution and conditioning a distribution. Say we have a joint distribution over $\{\boldsymbol{X}, \boldsymbol{Y}\}$ Then we can rearrange the mean vector and covariance matrix of the joint such that

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu_X} \\ \boldsymbol{\mu_Y} \end{pmatrix}, \begin{bmatrix} \Sigma_{\boldsymbol{XX}} & \Sigma_{\boldsymbol{XY}} \\ \Sigma_{\boldsymbol{YX}} & \Sigma_{\boldsymbol{YY}} \end{bmatrix} \right)$$

Then the marginal distribution over $\boldsymbol{Y}$ is simply $\mathcal{N}(\boldsymbol{\mu_Y}, \Sigma_{\boldsymbol{YY}})$. While for the distribution conditioned on a set of variables $\boldsymbol{Z}$, we can simply use the information form in Equation 12, where the values of variables in $\boldsymbol{Z}$ are set appropriately.

Another key idea which we can read off of the Gaussian distribution is the independencies it encodes. For any two $X_i, X_j \in \boldsymbol{X}$ where $\boldsymbol{X} \sim \mathcal{N}(\mu, \Sigma)$, we have $X_i \perp X_j$ iff $\Sigma_{ij} = 0$. Additionally and more interestingly, the information matrix provides us insight into conditional independencies. The following holds: $X_i \perp X_j | \boldsymbol{X} - \{X_i, X_j\}$ iff $J_{ij} = 0$. That is, given *everything else*, we know which two nodes of the network are independent of each other. Here, we come back to the notion of minimal I-maps, since we can read off independencies of the kind in Expression 11 in Section 3.1.1, from J. We can define the **Markov blanket** of a variable (node) X in BN graph structure $\mathcal{G}$, $MB_{\mathcal{G}}(X)$, as X's parents, X's children, and the parents of X's children. The Markov blanket d-separates X from *everything else* in the network, in some sense "shielding" it from external probabilistic influence. Thus, for all nodes X_i, X_j with entries $J_{ij} = 0$, they are definitely not in the Markov blankets of one another.

3.1.4 State-observation Model for Dynamic Bayesian Networks

All the discussion above has been on atemporal Bayesian networks. While these ideas remain the same for DBNs as far as describing the initial distribution $P(X^0)$ is concerned. To define the transition model $P(X'|X)$, we must first interpret the DBN as a **state-transition model**. For every observed template variable X_i, we assume it to be a hidden variable (or state) and generate its corresponding observed template variable (or observation) O_i, joining them by a directed edge $X_i \rightarrow O_i$. Now our model has two components, the usual transition model $P(X'|X)$ and the observation model $P(O|X)$. We can parametrize them for the continuous interface variables (modalities) by:

$$\begin{aligned} P(X^t | X^{t-1}) &= \mathcal{N}(AX^{t-1}, Q) \\ P(O^t | X^t) &= \mathcal{N}(HX^t, R) \end{aligned} \tag{13}$$

where A and H define linear transition and observation models respectively, and Q and R are corresponding Gaussian noises. This is referred to as a **Kalman filter**[13]. For the only discrete interface variable, tolerance, we can define a simple (valid) probability transition function $f : \mathbb{B} \times \mathbb{B} \rightarrow [0, 1]$. This combination of discrete and continuous variables in a temporal setting is often referred to as a **switching linear dynamical system**.

3.2 Learning the Bayesian Model

3.2.1 Parameter Estimation with Complete Training Data

The problem of parameter estimation is essentially that of "learning" the conditional probability distributions (CPDs), such as those listed in Table 1. In practice, the amount of data we have, especially with respect to a high-dimensional variable space, is rarely sufficient to learn an accurate representation of the entire space. Thus instead, we try to estimate a "best approximation" $\tilde{P}$ to the actual P, the notion of "best" depending on our learning and inference goals. This **bias-variance trade-off** underlies many of our design choices in learning. When selecting a hypothesis space of different models, we must take care not to allow too rich a class of possible models. Indeed, with limited data, the error introduced by variance may be larger than the potential error introduced by bias, and we may choose to restrict our learning to models that are too simple to correctly encode P[2]. Although the learned model is guaranteed to be incorrect, our ability to estimate its parameters more reliably may well compensate for the error arising from incorrect structural assumptions.

This trade-off is often seen when comparing **discriminative** to **generative** modeling. While the former tries to approximate to the conditional distribution $P(Y|X)$, the latter approximates to the joint $P(X, Y)$. Now clearly, the discriminative model makes assumptions of independence only on Y, while generative on both X and Y. Since the latter defines $\tilde{P}(X, Y)$, it induces both $\tilde{P}(Y|X)$ and $\tilde{P}(X)$ simultaneously, while the former must only fit $\tilde{P}(Y|X)$ well. Thus although generative models have a higher bias, they help regularize the model, thereby reducing its ability to overfit to data, making it more useful when limited data is available. On the other hand, as the amount of data grows, the bias imposed can dominate the error of learning, which is when discriminative models become more useful.

Say we are given a complete (no missing values) training dataset $\mathcal{D} = \{\mathcal{X}^i\}_{i=1}^M$ on the variable space, which is what we use to learn the model parameters ϕ. We define the likelihood function as the probability which the model assigns to our data, assuming every data point is an independent identically distributed (IID) variable:

$$L(\phi : \mathcal{D}) = \prod_m P(\mathcal{X}^m|\phi)$$

We can now define the problem of parameter estimation as finding ϕ which maximizes this likelihood, referred to as **maximum likelihood estimation** or MLE:

$$L(\hat{\phi} : \mathcal{D}) = \max_{\phi \in \Phi} L(\phi : \mathcal{D}) \tag{14}$$

In the context of Bayesian networks, because the distribution decomposes into local distributions at every node, we can easily estimate parameters for every child node, given its parents. This property is called the global decomposition of likelihood function, and implies that we can maximize each likelihood function independent of rest of the network, and combining them for the MLE solution.

$$\begin{aligned} L(\hat{\phi} : \mathcal{D}) &= \max_{\phi \in \Phi} \prod_i L_i(\phi_{X_i|Pa(X_i)} : \mathcal{D}) \\ &= \prod_i \max_{\phi \in \Phi} L_i(\phi_{X_i|Pa(X_i)} : \mathcal{D}) \end{aligned} \tag{15}$$

Intuitively enough, for discrete variables of our model, MLE is nothing but the usual probabilistic adage of "number of favorable outcomes upon the total number of outcomes" Say we have $Pa^{\mathcal{G}}(X_i) = \{U_j\}_{j=1}^k$ then:

$$\phi_{X_i=x|U_1=u_1,\cdots,U_k=u_k} = \frac{\sum_m \mathbf{1}\{\mathcal{X}_i^m == x, \mathcal{U}_1^m == u_1, \cdots, \mathcal{U}_k^m == u_k\}}{\sum_m \mathbf{1}\{\mathcal{U}_1^m == u_1, \cdots, \mathcal{U}_k^m == u_k\}} \tag{16}$$

where $\mathbf{1}\{\mathcal{Y} == y\}$ is the indicator function that outputs 1 if assignment of $\mathcal{Y}$ is y, else 0. For the conditional linear Gaussian distribution, this amounts to solving a system of linear equations. These are referred to as **sufficient statistics**. Now, say we have $P(X|U) = \mathcal{N}(\beta_0 + \beta_1 U_1 + \cdots + \beta_k U_k, \sigma^2)$ then we can solve for MLE parameters $\phi_{X|U} = \langle \beta_0, \beta_1, \cdots, \beta_k, \sigma \rangle$ as:

$$\begin{aligned} E_{\mathcal{D}}[X] &= \beta_0 + \beta_1 E_{\mathcal{D}}[U_1] + \cdots + \beta_k E_{\mathcal{D}}[U_k] \\ Cov_{\mathcal{D}}[X, U_i] &= \beta_1 Cov_{\mathcal{D}}[U_1, U_i] + \cdots + \beta_k Cov_{\mathcal{D}}[U_k, U_i] \\ \sigma^2 &= Cov_{\mathcal{D}}[X, X] - \sum_{i,j} \beta_i \beta_j Cov_{\mathcal{D}}[U_i, U_j] \end{aligned} \tag{17}$$

Note that some of the parameters in the model have hyperparameters (we can introduce as many as we like), which take parameters to be as random variables with a prior distribution $P(\phi)$. This allows us to not take a mere point-estimate of the "best parameter", rather maintain a *belief* about ϕ's values, and use these beliefs to reach conclusions. For this, we can write the probability of parameters akin to Equations 5 and 8

$$P(\phi|\mathcal{D}) = \frac{P(\phi)P(\mathcal{D}|\phi)}{P(\mathcal{D})} \tag{18}$$

Now, maximizing this quantity gives us the **maximum a posteriori** or MAP estimate (assuming prior probabilities are locally decomposable as well, and noting that $P(\mathcal{D})$ is independent of ϕ):

$$L(\mathcal{D} : \hat{\phi}) = \prod_i \max_{\phi \in \Phi} L_i(\phi_{X_i|Pa(X_i)} : \mathcal{D}) P(\theta_{X_i|Pa(X_i)}) \tag{19}$$

If we convert this into log-likelihood, then we can see how the prior acts almost as a regularizer on the likelihood, which constrains the model and makes it more useful in the limit of sufficiently available data

$$log(L(\mathcal{D} : \hat{\phi})) = \sum_i \max_{\phi \in \Phi} log(L_i(\phi_{X_i|Pa(X_i)} : \mathcal{D})) + \sum_i \max_{\phi \in \Phi} log(P(\theta_{X_i|Pa(X_i)}))$$

However, we still had to reduce our parameters to some new, although more appropriate, point-estimate. We can skip that entirely, by doing a **full-Bayesian** analysis over the parameter space, by averaging over it. That is, say to infer about an entirely new test data point $\mathcal{X}'$, and using that in the meta-Bayesian network $\phi \to \{\mathcal{X}\}_{i=1}^m$ all data points are d-separated given the parameters ϕ, we can do the following:

$$\begin{aligned} P(\mathcal{X}'|\mathcal{D}) &= \int P(\mathcal{X}'|\mathcal{D}, \phi) P(\phi|\mathcal{D}) d\phi \\ &= \int P(\mathcal{X}'|\phi) P(\phi|\mathcal{D}) d\phi \\ &= E_{P(\phi|\mathcal{D})} \left(P(\mathcal{X}'|\phi) \right) \end{aligned} \tag{20}$$

The downside of full-Bayesian learning is that it is often difficult to compute such integrals in closed-form, and in a computationally efficient manner. If the posterior distributions $P(\phi|\mathcal{D})$ are in the same family as the prior probability distribution $P(\phi)$, then the prior is called a **conjugate prior** for the likelihood function $P(\mathcal{D}|\phi)$. For example, the Gaussian family is conjugate to itself, with respect to a Gaussian likelihood function: if the likelihood function is Gaussian, choosing a Gaussian prior over the mean will ensure that the posterior distribution is also Gaussian. Similarly for discrete distributions, a Beta distribution is the conjugate prior for Bernoulli, and the Dirichlet distribution for Categorical (hence used in Table 1). A conjugate prior is an algebraic convenience, giving a closed-form expression for the posterior. Furthermore, conjugate priors give intuition by transparently showing how observing $\mathcal{D}$ "updates" the distribution on ϕ.

Note that for Dynamic Bayesian networks, the parameters for discrete variables (namely tolerance here) can be straightforwardly estimated using aggregate sufficient statistics, that is by pooling the sufficient statistics across the unrolled (grounded) Bayesian network. However for the continuous variables, when we are in the realm of Kalman Filters, although estimation can be done in a closed form it is too complicated to be discussed here. The reader is referred to [14] for more.

3.2.2 Parameter Estimation with Incomplete Data for Crossmodality

Recall the extended feature space (A_{all}) model described in Section 2.2. Since every training/testing data point would be derived from only one species, we have "missing" data features for each of them. In particular, $\epsilon_k = |A_k - A_{shared}|$ number of incomplete features for the k^{th} species. Now this situation can be handled straightforwardly at the time of inference, by considering the query over a probability distribution which is marginalized over missing features.

However, we run into a problem during parameter estimation. Since the likelihood function would now be a sum over all possible joint assignment to the missing features, it loses its critical property of decomposability. Which implies that we also cannot expect to maximize the overall likelihood by maximizing them independently at every node. Additionally, the number of joint assignments is exponential in the number of missing values ϵ_k, which would likely be a very large number indeed. Thus eventually, it becomes necessary to come up with some estimate of these missing values, to retain the nice properties of decomposable CPDs which we earlier enjoyed. One popular algorithm to do this is called the **expectation-maximization** or EM algorithm. The intuition behind EM is to start with a "rough" estimate of the parameters of the model, and then through an iterative procedure, keep improving the actual likelihood estimates by trying to optimize over a simpler "expected" likelihood function. This is done by iteratively interleaving between the E-step of inferring missing values assuming correct parameters, and M-step of parameter estimation by assuming fully observed feature values.

Unfortunately, there are some problems with EM. Due to the convexity of the likelihood function, it generally converges to a local maxima, than a global one. Even if one tries to resolve this using simulated annealing or other stochastic strategies, it present issues in a high-dimensional setting such as ours. When number of dimensions D is much larger than sample size N ($N \ll D$) then EM offers no theoretical guarantees of convergence, unless there are careful sparsity constraints on the model[15]. Therefore, we depart from an EM setting, towards that of transfer learning.

There are multiple ways to transfer knowledge across domains, namely[3]:

1. **Instance-transfer**, wherein we re-weight data in the source domain (other species) for using it in the target domain (the species of interest).

2. **Feature-representation transfer**, in which we find a "good" shared representation that reduces the difference between source and target domains.

3. **Parameter-transfer**, wherein we discover shared parameters or priors between the two domains.

4. **Relational-knowledge-transfer**, in which maps of relational knowledge are built.

We employ a composition of the first three flavors of transfer learning in our model. Let us describe these ideas starting with the third one, up to the first:

Priors on Host Species As seen in Figure 8 where we have the variable θ that represents the species category, the probability distribution on it, $P(\theta)$, is itself parametrized not by fixed values, but by another random variable ρ. That is, instead of evaluating point estimates of the species to which the data belongs, we maintain a *belief* over the space of species. This deeper sense of parameter tying aids in parameter-transfer flavor of transfer learning.

Finding a shared low-D space for Data Recovery using mDRUR Most natural high-dimensional datasets, with or without missing values, tend to lie on a low-dimensional manifold with very few degrees of freedom (see Figure 11). Thus, if we are able to find the underlying manifold of our data, we can map from this low-D space back to the original high-D space, hence "filling in" for missing values in the data. In the transfer learning setting, we make the assumption that this low-D space is a shared representation, from which the data of all species arises.

This problem setting combines two popular problems in data analysis and linear algebra: nonlinear dimensionality reduction (or manifold learning) and the matrix completion problem. In [10], the authors suggest a method for missing data recovery through dimensionality reduction with unsupervised regression, or mDRUR. Say N data points in high-D space are $Y \in \mathbb{R}^{D\times N}$, to be mapped to a low-D manifold $X \in \mathbb{R}^{L\times N}$. The maps are given by $f : X \to Y$ and $F : Y \to X$. Referring to missing data features as Y_0, we must return the values of X, Y_0, and the maps f, F. We do this in an EM-style algorithm, where assuming some maps, we first minimize the "reconstruction error"

$$E(X, Y_0) = ||Y - f(X)||^2 + ||X - F(Y)||^2 + \lambda_f ||f|| + \lambda_F ||F||$$

(which can be seen as the unfolding of an autoencoder) with respect to unknowns X and Y_0. Then, assuming correct X and Y_0, we find the maps by assuming they have some parametric forms, say linear maps like $Y = f(X) = AX + a$ and $X = F(Y) = BY + b$, and thus solving these two linear regressions. We then iteratively refine these estimates to convergence. There are certain practical considerations of the algorithm, like a good initial estimate of X, which could be found by some simple missing value imputation on Y and then applying a spectral method like PCA, or a simpler dimensionality reduction method like random projections. Also, we can choose more complicated kernels rather than simple linear mappings. But in all, mDRUR has been shown to be successful in recovering missing data with very high number of missing values, even as much as 50% values missing in a high-D setting.

Therefore, appending a step of mDRUR on the modalities will allow us to perform regular parameter estimation and inference in the fully-observable data setting, where CPDs are decomposable. A point of consideration here, which needs empirical testing, is **whether the states of tolerance and intolerance are identified by two different manifolds, or the same manifold**. Algorithmically, this would mean that we either divide the dataset by tolerance labels (and perhaps other labels like the pathogen type) and run separate mDRURs, or apply it on the entire aggregate dataset by disregarding the labels. Additionally, a parameter which would require tuning based on the log-likelihood score, is the manifold dimension L.

Boosting for Instance Weighting Although mDRUR is likely to improve the initial estimates of Y_0 and escape local minimas, it may not be a good idea to completely trust the training instances obtained from another species, for the species of interest, with very high confidence. In other words, we can weight cross-species training instances appropriately, for more accurate transfer of learning. One such technique, in the supervised learning setting, is called TrAdaBoost[8]. Based on the principle of boosting, developed as AdaBoost as explained in [9], TrAdaBoost trains a classifier on an aggregate weighted cross-domain dataset, and iteratively updates weights of training instances depending on how they minimize the training error on the target domain dataset. In this sense, it (down)upweights those cross-domain instances which are very (dis)similar to the target domain.

Let the training instances be $\mathcal{D}_t = \{(X_i, Y_i)\}_{i=1}^{|\mathcal{D}_t|}$ from the target domain and $\mathcal{D}_o = \{(X_i, Y_i)\}_{i=1}^{|\mathcal{D}_o|}$ from other domains. Note that in TrAdaBoost, we are in the supervised setting, so we must manipulate our Bayesian Model as following. Since we want to optimize our training procedure for predicting tolerance, we consider it to be the output label $Y \in \{0, 1\}$ of this classification problem, X to be all other variables of the model, where $f : X \to Y$, and $P(Y|X)$ as the confidence of classification and hence our predicted hypothesis $\mathcal{H} : X \to [0, 1]$. Let us define the aggregate dataset, with target instances indexed first, as $\mathcal{D} = \mathcal{D}_t \cup \mathcal{D}_o$. Given an estimate of the weight vector at iteration t as $w^t \in \mathbb{R}^{|\mathcal{D}|}$, maximum number of iterations N, we can:

1. Train the Bayesian model by using weighted sufficient statistics, a modified version of Equation 16 given by:

$$\phi_{X_i=x|U_1=u_1,\cdots,U_k=u_k} = \frac{\sum_m w_m^t \mathbf{1}\{\mathcal{X}_i^m == x, \mathcal{U}_1^m == u_1, \cdots, \mathcal{U}_k^m == u_k\}}{\sum_m w_m^t \mathbf{1}\{\mathcal{U}_1^m == u_1, \cdots, \mathcal{U}_k^m == u_k\}}$$

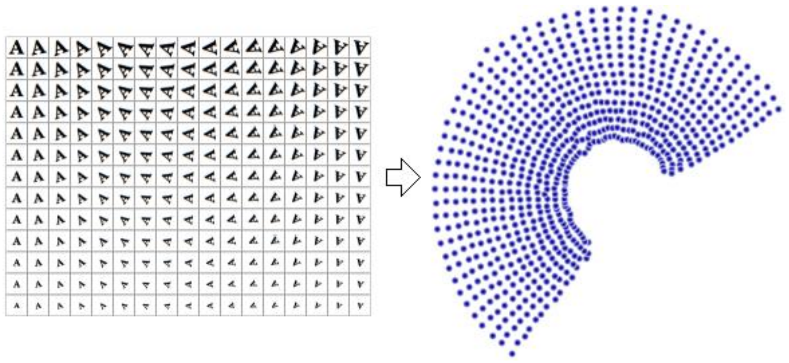
Figure 11: A dataset where the letter A has just been scaled and rotated across different samples, and thus has only 2 degrees of freedom, which the low-dimensional manifold on the right captures well. Source: Wikipedia entry on Nonlinear Dimensionality Reduction

2. Infer $\mathcal{H}_t(X) = P(Y|X)$
3. Define error on the target domain as

$$\epsilon_t = \sum_{\{X_i, Y_i\} \in \mathcal{D}_t} \frac{w_i^t |\mathcal{H}_t(X_i) - Y_i|}{\sum_{i=1}^{|\mathcal{D}|} w_i^t}$$

using which we define the parameters

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}, \beta = \frac{1}{1 + \sqrt{2ln(|\mathcal{D}_t|)/N}}$$

and update the weight vector for next iteration as

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|\mathcal{H}_t(X_i) - Y_i|}, & 1 \le i \le |\mathcal{D}_t| \\ w_i^t \beta_t^{-|\mathcal{H}_t(X_i) - Y_i|}, & |\mathcal{D}_t| + 1 \le i \le |\mathcal{D}| \end{cases}$$

After applying TrAdaBoost, the crossmodal training dataset $\mathcal{D}$ is now tailored to be used with the weight vector w_N, for learning the Bayesian Model of a particular target species t. If we have multiple species of interest, we can repeat this algorithm for that particular species.

The intuition behind TrAdaBoost makes sense, and further mathematical analysis on AdaBoost reveals how boosting works. Given that the classifier (here, Bayesian Model) performs (just) better than random (called the **weak learner condition**), it is mathematically provable using the theory of VC Dimensions that the training error of AdaBoost decreases to zero very rapidly, eventually perfectly fitting the training set[9]. Thus, it can seem to tend to overfit the data. However, given enough training data, and if the classifier is much better than random, then the "margins" or confidence with which it fits data greatly improves, thus avoiding overfitting. Also, there is a sense of l_1-regularization implicit to AdaBoost, whose effect weakens with more number of iterations. Thus, an early stopping could further keep overfitting in check. Lastly, although TrAdaBoost doesn't *always* improve on AdaBoost, if the quality of distributions in the target and cross-domains is not poor, it can greatly improve accuracy on final predictions[8].

3.2.3 Learning Hidden Layer Structure for Modularity

As discussed in Section 3.1.1, many BN structures can satisfy the set of independencies of the joint distribution they are I-maps to. Deeper thought leads us to realise that a fully-connected Bayesian Network will encode every possible P, and we might just learn that as the "best" structure. Indeed, this is the result of doing an MLE on $P(\mathcal{D}|\mathcal{G})$.

Let us define the score of a graph structure G as the log-likelihood of data when we use the MLE parameters $\hat{\phi}_{\mathcal{G}}$ given $\mathcal{G}$. That is,

$$score_{MLE}(\mathcal{G} : \mathcal{D}) = log(P(\mathcal{D}|\mathcal{G})) = l(\hat{\phi}_{\mathcal{G}} : \mathcal{D})$$

Now using the definitions of likelihood and some algebraic manipulations, we can write this score as

$$score_{MLE}(\mathcal{G}:\mathcal{D}) = M\sum_{i=1}^{n} I_P(X_i; Pa^{\mathcal{G}}(X_i)) - M\sum_{i=1}^{n} H_P(X_i)$$

where $I_P(A;B)$ is the mutual information between A and B under P, and $H_P(A)$ is the entropy of A under P

$$H_P(A) = -\sum_{a\in A} P(a)log(P(a))$$

$$I_P(A;B) = \sum_{a\in A}\sum_{b\in B} P(a,b)log\left(\frac{P(a,b)}{P(a)P(b)}\right)$$

Thus, the likelihood of the network measures the strength of dependencies betweens variables and their parents. Notice that the second term of the score is same for all structures, while the first term would naturally be larger for more highly-connected networks, reaching its maxima at the fully connected network. This would obviously make training and inference intractable, and we would like to ensure some sort of sparsity in $\mathcal{G}$. This is exactly what can be done by using a prior distribution over structure space, and taking a MAP estimate instead of MLE, as expressed in Equation 8. The score can be written as

$$score_{MAP}(\mathcal{G}:\mathcal{D}) = log(P(\mathcal{D}|\mathcal{G})) + log(P(\mathcal{G}))$$

Furthermore, if we choose Dirichlet priors over all parameters, then we can write the score as

$$score_{MAP}(\mathcal{G}:\mathcal{D}) = M\sum_{i=1}^{n} I_P(X_i; Pa^{\mathcal{G}}(X_i)) - M\sum_{i=1}^{n} H_P(X_i) - \frac{log(M)}{2}Dim[\mathcal{G}] \tag{21}$$

where $Dim[\mathcal{G}]$ refers to the number of parameters in $\mathcal{G}$. Thus maximizing this score, also known as the **Bayesian Information Criterion** score, gives us a nice trade-off between capturing dependencies as well as maintaining a sparse structure.

Now, one can do structure learning in the same way as one does parameter estimation. However, since the space of structures is superexponential in the number of nodes, and especially because we are in the missing data regime, this will be an intractable challenge. Therefore, we need to use some heuristics and additional constraints, to realize our modular Bayesian model. As described in Section 2.4, we need only consider a single layer of hidden variables that is sandwiched between host and pathogen variables, while ensuring that every biological modality has a single latent modality for its parent. This hugely constrains the space of our structures, which can essentially be parametrized by a single variable $\kappa = |H|$, the number of hidden nodes. Now clearly, $1 \leq \kappa \leq |A|$, and for a given κ we will have exactly $S(|A|,\kappa)$ number of assignments of hidden parents to biological modalities, where $S(n,k)$ refers to Stirling number of the second kind (the number of ways to partition a set of n objects into k non-empty subsets). This gives us a total of $B_{|A|} = \sum_{\kappa=0}^{|A|} S(|A|,\kappa)$ number of structures, where B_n is the n^{th} Bell number. Using very rough upper limit for B_n, we can say

$$B_{|A|} \leq |A|^{|A|}$$

which gives us an exponentially large number of possible structures in the unconstrained hidden layer. However, using some domain knowledge about the "number of modules" possible, say $\hat{\kappa}$, in the biological system response, we can constrain it to just $\hat{\kappa}$ number of hidden variables, and thus $S(|A|,\hat{\kappa})$ number of structures. Using another rough upper limit for $S(n,k)$, we can say that

$$S(|A|,\hat{\kappa}) \leq |A|^{\hat{\kappa}}$$

This is still potentially a very large number, because it is exponential in the number of biological modalities, although raised to a seemingly smaller power.

Let us now use a heuristic inspired by Equation 21. Clearly, a good structure is the one where mutual information between a variable and its parents is maximized. Given that the rest of the network structure is fixed, we look particularly at the way the hidden layer is a parent to biological modalities. Although we don't know the distribution of these hidden labels, we can "induce" them to be more mutually informative about their children by making them parents of mutually informative children. We can therefore reduce the problem of structure learning here to that of finding subsets of biological modalities, such that the variables in a set are highly mutually informative of one another, and then assign every subset to the same parent hidden variable h.

A straightforward way to do this is through **hierarchical agglomerative clustering** (HAC), wherein N data points (here, the biological modalities) are hierarchically clustered from bottom up such that at any level in the hierarchy, intra-cluster points are closer and inter-cluster points are farther. Therefore, HAC requires as its input (a) a (symmetric) pairwise distance matrix $D \in \mathbb{R}^{N \times N}$ where D_{ij} is some distance metric between points i and j, and (b) a linkage criterion to decide the clusters as a function of pairwise distances between cluster members. Because we want to subset modalities according to mutual information, a good distance metric to use here is the **variation of information** given by

$$\begin{aligned} VI(X;Y) =& H(X,Y) - I(X;Y) \\ =& H(X|Y) + H(Y|X) \end{aligned} \tag{22}$$

where $H(X,Y)$ refers to joint entropy over X and Y while $H(X|Y), H(Y|X)$ are the corresponding conditional entropies. Unlike mutual information, variation of information is a true metric in that it follows the triangle inequality, amongst other key metric properties. For the linkage criterion, it is sensible to use "mean" linkage clustering, the idea being that hidden modalities would separate the cluster centers. Once the clustering is done, we can recover the clusters at any level of the hierarchy, depending on the number of clusters that we want: 1 structure for every given number of hidden variables κ, which gives us just $|A|$ number of structures, exactly linear in the number of biological modalities and a huge reduction from $|A|^{|A|}$ or $|A|^{|\hat{\kappa}|}$. Deciding the number of clusters κ is a key parameter. As discussed in Section 2.4, it must be much smaller than the number of biological modalities. Indeed, some biological knowledge can be used to restrict the range of κ, which can then be fine-tuned by using the log-likelihood score on training data. An engineering question which must be tested is whether structure learning should be done for every species separately with just the training data $\mathcal{D}_t$ for our species of interest, or on the entire crossmodal model using the aggregate dataset $\mathcal{D}$.

3.3 Making Inferences

The key advantage of a probabilistic model of tolerance is the amount of flexibility it offers, in terms of the possible queries which can be made to the model. Since a joint distribution P is learnt over the entire feature space, one can ask any arbitrary queries about a subset of features, conditioned on any subset of features. Although popular exact inference algorithms like the variable elimination algorithm, message passing algorithm, and belief propagation algorithm exist, for very large networks such as ours they are highly intractable. Therefore, we work with approximate particle-based sampling algorithms for inference, in particular, Monte Carlo Markov Chain method of Metropolis Hastings, using a collapsed particle setting for hybrid dynamic Bayesian networks (called **Rao-Blackwellised Particle Filtering for a Switching Kalman Filter**)[16]. Due to the engaging complexity of these algorithms, the reader is referred to Chapter 5 of [13] and Chapters 12, 14 and 15 of [2].

We now elucidate below some interesting queries which can be made to this model:

Predicting Tolerance This is the most important kind of prediction expected of the model to make. Given the data $\mathcal{D}$ of a new subject (of any species k), classify it as tolerant or sensitive. The probabilistic query which can be made here is $P(\lambda|\mathcal{D})$, where $\mathcal{D}$ may contain complete evidence of every other variable of the model, or only partial evidence (wherein the query would marginalize over the non-evidential variables). This allows us to provide as little, or as much, data available, say using just the metabolomics data, or only transcriptomics data, or all the biological modalities. One could enquire even without knowing the type and amount of pathogen that has infected the host, and still come up with a reasonable estimate of tolerance. Additionally, querying without providing the host species k in evidence would induce a marginalization over host species, permitting more domain transfer. Moreover, since this is a temporal model, we can predict future states of the host based on present state data (referred to as "particle tracking" in Kalman Filter literature), permitting an "early prediction" of tolerance, aiding in early medical interventions. Furthermore, one can always extend the model to differentiate not just *tolerant* states to those that are *sensitive*, but also to those that are entirely *resistant* to infection.

Cross-Species Analyses Using data $\mathcal{D}$ for a different species k (such as a pig, frog or mouse), which could include its state of tolerance, pathogen information, etc. we can make interesting queries on our species of interest (say *Homo sapiens*). We can fill in for the biological modalities of humans, a_h, using MAP queries like $\hat{a}_h = \arg\max_{a_h} P(a_h|\mathcal{D}_k)$. These could help us simulate biological conditions of and do in-silico experiments on a species of interest, by conducting biological in-vivo experiments on another.

Determining Key Biomolecules and Pathways Finding differentially expressed (DE) biological modalities becomes a very straightforward task in this setting. For a modality of interest a, one can find out differential expression by estimating the difference in the conditional distributions $P(a|\lambda = 0)$ and $P(a|\lambda = 1)$ using Kullback-Liebler Divergence, which for discrete distributions P_1 and P_2 is given by

$$D_{KL}(P_1||P_2) = \sum_i P_1(i) log \left(\frac{P_1(i)}{P_2(i)} \right)$$

The higher is D_{KL}, the more differentially expressed the modality is. Up or down regulation can be figured by comparing means of $P(a|\lambda = 0)$ and $P(a|\lambda = 1)$. Usually, these DE modalities are used as a proxy for "key biomolecules" which could be driving the tolerance response. However, a good analysis would be to see *their effect* on tolerance itself, a more direct measure of keyness. That is, defining importance of a modality in driving tolerance as D_{KL} between $P(\lambda|a_h = min(a_h))$ and $P(\lambda|a_h = max(a_h))$. Moreover, this effect on tolerance can be checked at the level of a module, by querying for tolerance given a hidden node. Identified sets of key genes, proteins and metabolites can then be mapped to important biochemical pathways.

Simulating Perturbation Studies The advantage of having a computer model which directly maps to the underlying biology is that experiments can be "simulated" on the computer model itself. A Bayesian Network model allows opportunities to do various mutation and perturbation studies, say by removing certain edges of the network structure and seeing their impact on predicting tolerance. This will direct more focused research in host-pathogen interactions at a faster pace, by exploiting the rich ontology of this model to quickly pre-discover novel observations and mechanisms, worthy of expensive and time-consuming confirmatory tests at the lab.

3.4 Challenges

The power of this model is contingent on a number of challenges which must be overcome to make confident predictions. The nature of our problem is skewed from a conventional machine learning problem setting, in that we have very high-dimensional data (say of dimension D), but we can expect small training datasets of size N such that $N \ll D$. Usually, an ML algorithm learns on a real-word training dataset well, given that N is exponential in D, and hence $N \gg D$. Therefore in our flip-case, the **curse of dimensionality** might appear to be heavy, making these two key challenges to be addressed. Fortunately, the premise of Bayesian Networks is that of a compact representation of the feature space in terms of conditional independencies. Thus, the entire dataset can be used to independently inform us about subsets of the feature space, each of whose dimensionality is much much smaller.

3.4.1 Dealing with High-dimensional Data through Random Projections

Another key idea for dealing with the curse of dimensionality is to preprocess our data through a step of dimensionality reduction (DR). The problem of DR is essentially of finding a map from the original high-D space $\mathcal{X} \in \mathbb{R}^D$ to a low-D space $\mathcal{Y} \in \mathbb{R}^L$, where $L \ll D$. Usually, the construction of this map depends on not only the data, but also on the purpose for which we want to reduce the dimensions, thus giving rise to various DR algorithms. Recall that we used one such algorithm called mDRUR in Section 3.2.2, but only temporarily, so as to impute missing values in the data. Nothing particularly stops us from reducing the entire feature space of the model. One of the fastest and simplest DR algorithms which can be used for that, is called Random Projections (RPs). An RP finds a map $f_{RP} : \mathcal{X} \to \mathcal{Y}$ which preserves the pairwise distances between all points of the dataset. That is $\forall x_1, x_2 \in \mathbb{R}^D$,

$$||x_1 - x_2|| \approx ||f_{RP}(X_1) - f_{RP}(X_2)||$$

Although various maps satisfy this property, a famous map is the Normal RP, wherein $f_{RP}(x) = Rx$ where $R \in \mathbb{R}^{L \times D}$ such that $R_{ij} \sim \mathcal{N}(0, 1/D)$. Interestingly, this map preserves the entire subspace, and also maintains cluster separations, while making the lower dimensional Gaussians more spherical, thus making it easier to model the distribution as a Conditional Linear Gaussian (or a mixture of multivariate Gaussians)[11]. Which is why this is a suitable DR technique for our Bayesian model. We can apply RP on the entire huge space of biological modalities for our training and testing datasets with the same function f_{RP}.

One problem which we immediately run into, however, is that of defining the underlying biological network for this random projected feature space. Although we'll have a much smaller and thus more tractable Bayesian Network to work with, its connectivities would be quite unclear. This issue can be redressed by using structure learning, similar to that described in Section 2.4. However, another issue which we face is that we lose a direct correspondence between the biological system and the computational model, making it less interpretable. This is because we cannot project back to the original space $\mathcal{X}$ from $\mathcal{Y}$ using RP. A workaround for this is to use a different DR method, which is so-to-speak "reversible", like Principle Component Analysis, or PCA. However, applying PCA will certainly not preserve cluster separations, giving us a poor, albeit interpretable, model. In conclusion, we must empirically test out the advantages of applying dimensionality reduction to the entire feature space, given the costs and benefits of doing so, to the overall model quality and interpretability.

3.4.2 Dealing with Small Training Datasets by being a Bayesianist

As discussed in Section 3.2.1, the notion of conjugate priors gives us a nice interpretation of the Bayesian approach to probabilistic modeling. That the probability distribution on our feature space $\mathcal{X}$ is a "prior belief of our world" $P(\mathcal{X})$, which gets updated as we acquire more and more data samples $\mathcal{D}$ from the world, (through multiplication with the likelihood of data given current belief: $P(\mathcal{D}|\mathcal{X})$,) to what is called the posterior belief $P(\mathcal{X}|\mathcal{D})$. The process of acquiring data is merely an incremental step in our belief of the world. Thus, in the limit of small training data, if one assumes some appropriate "pseudo data points" to begin with, we can be more confident in our initial beliefs. This, in essence, is exactly what maintaining a prior does.

For example, in a Bernoulli trial experiment of sequence of coin tosses, with probability of heads for the coin being θ, the MLE comes out to be $\theta = \frac{|H|}{|H|+|T|}$, where $|H|$ is number of heads and $|T|$ is number tails in the sequence. Having a Beta prior on it, that is assuming $\theta \sim Beta(\alpha, \beta)$, gives a MAP estimate of $\theta = \frac{|H|+\alpha}{|H|+|T|+\alpha+\beta}$. Intuitively, α works as a pseudo-count of number of heads and β as a pseudo-count of number of tails. So if we have a prior belief for the coin to be fair, we can bias the value of θ to remain close to 0.5 by selecting large equal values for α and β.

This strategy is often referred to as a Bayesianist approach to probability, contrary to a frequentist approach. In the current model, we maintain a prior only on the host species variable. However, we could add a prior on any other variable, which we strongly believe to center around some prior value. This would relax a little pressure from the training data to inform our models. But defining and designing priors for this problem, is going to be a scientific challenge in itself, which could require further tuning. (A mild extension of help here is that of a hierarchical Bayes model, but it will still not circumvent this massive challenge.) And if grossly wrong, priors could bias the entire model to fit to noise. Therefore, it remains to be empirically tested if a Bayesianist approach will improve our model, or not.

References

[1] Thomas L. Griffiths, Charles Kemp and Joshua B. Tenenbaum, *Bayesian models of cognition*

[2] Daphne Koller and Nir Friedman, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, Cambridge, Massachusetts (2009)

[3] Sinno Jialin Pan and Qiang Yang, *A Survey on Transfer Learning*, IEEE Transactions on Knowledge and Data Engineering (2009)

[4] Stefen L. Lauritzen and Frank Jensen, *Stable local computation with conditional Gaussian distributions*, Statistics and Computing (2001) 11, 191–203

[5] Timo J. T. Koski and John M. Noble, *A Review of Bayesian Networks and Structure Learning*, Mathematica Applicanda (2012), vol. 40(1), 53–103

[6] Hal Daumé III and Daniel Marcu, *Domain Adaptation for Statistical Classifiers*, Journal of Artificial Intelligence Research 26 (2006), 101-126

[7] Jing Jiang and Cheng Xiang Zhai, *Instance Weighting for Domain Adaptation in NLP*, Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (2007), 264–271

[8] Wenyuan Dai, Qiang Yang, Gui-Rong Xue and Yong Yu, *Boosting for Transfer Learning*, Proceedings of the 24th International Conference on Machine Learning (2007)

[9] Robert E. Schapire, *Explaining AdaBoost*

[10] Miguel Á. Carreira-Perpiñán and Zhengdong Lu, *Manifold Learning and Missing Data Recovery through Unsupervised Regression*, IEEE 11th International Conference on Data Mining (2011)

[11] Sanjoy Dasgupta, *Experiments with Random Projection*, Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (2000), 143-151

[12] Eric Bauer, Daphne Koller and Yoram Singer, *Update rules for parameter estimation in Bayesian networks*, Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (1997), 3-13

[13] Kevin P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD Thesis (2002)

[14] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu and Stefano Soatto, *Dynamic Textures*, International Journal of Computer Vision (2003), 51(2), 91–109

[15] Zhaoran Wang, Quanquan Gu, Yang Ning and Han Liu, *High Dimensional Expectation-Maximization Algorithm: Statistical Optimization and Asymptotic Normality*

[16] Arnaud Doucet, Nando de Freitas, Kevin Murphy and Stuart Russell, *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*, Proceedings of the 16th conference on Uncertainty in Artificial Intelligence (2000), 176-183