CSCI E-82a
Probabilistic Programming and AI
Lecture 4
Learning for Graphical Models

Steve Elston

HARVARD
Extension School

Outline

- Learning for Graphical Models
- Introduction to Learning Model Parameters
    - The Binomial and Beta Distributions
    - The Kullback-Leibler Divergence
    - Maximum Likelihood for the Binomial Distribution
    - Bayesian Estimation for Binomial Distribution
    - Categorical and Dirichlet Distributions
    - Working with Multivariate Distributions
- Learning Model Structure
    - Learning structure of DAGs
    - Mutual Information for K2 Scoring
    - Chow-Liu Tree Algorithm

Reminders

- Homework 2 is due today!
- Homework 3 is due next Wednesday, October 2
- Homework 4, TBD – depending on how far we get in lecture

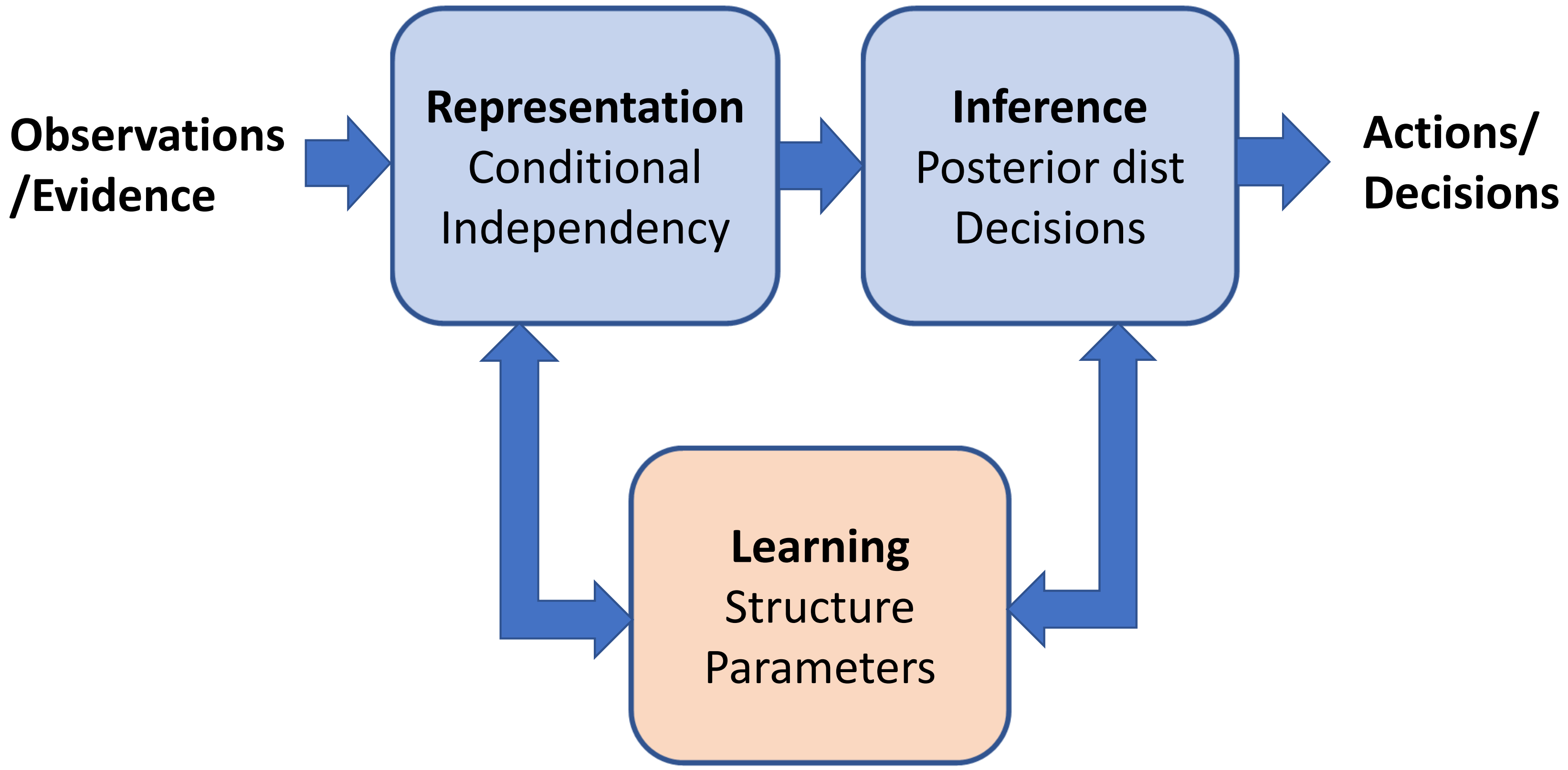
- Next week; making decisions under uncertainty
- Following week; approximate inference and time dependent models
- Then, on to reinforcement learning

Learning for Graphical Models

Learning can involve estimating model parameters or finding structure

- Estimating model parameters is relatively easy
  - Requires **minimal data**
  - **Learning can be incremental**
- Learning structure is very difficult
  - Is **unsupervised learning**
  - Potentially requires **massive amounts of data**
  - Exhaustive methods are **NP complete!**

Learning for Graphical Models



Learning for Graphical Models

In the first 2 lessons we focused on **representation**

- Representation allows us to define or represent a model
- We have investigated two representations:
    - **Bayes networks** or directed acyclic graphs (DAGs)
    - **Undirected graphical networks**, Markov random fields (MRF)
- **Efficient representation**
    - Factor distributions
    - Represent independences
- A good representation is required to perform inference

Learning for Graphical Models

In the last lesson our focus was on **inference**

- Inference is the process of getting useful results from graphical models
  - Posterior distributions
  - MAP
- Exact inference algorithms include:
  - Variable elimination
  - Belief propagation
  - Junction-tree algorithm

Part 1
Learning Model Parameters

HARVARD
Extension School

Introduction to Learning Model Parameters

Two classes of learning methods

- Frequentist method
  - Need only **likelihood function**
  - Find parameters for **maximum likelihood (ML)**
- Bayesian learning
  - Requires **prior distribution** and likelihood function
  - Can compute **posterior distribution** or **maximum a postiori (MAP)**
  - **Learn incrementally** when new observations are available

The Binomial and Beta Distributions

Models for binary variables in {0,1}

- **Single realization**, {0,1} == {failure, success}
- **Bernoulli distribution** models **probability of success**, Θ, for single trial:

$$p(v = 1) = \Theta$$

where:

$$v = an\ observation$$

$$\Theta = probability\ parameter$$

- Bernoulli distribution is a **parametric distribution** with a single parameter

The Binomial and Beta Distributions

Models for binary variables in {0,1}

- How do we model distribution of multiple trials
- **Binomial distribution** determines the probability of *k* successes in *n* trials, given a parameter Θ

$$p(v = k \mid \Theta) = \binom{n}{k} \Theta^k (1 - \Theta)^{n-k}$$

Where, $\binom{n}{k}$ is the **binomial coefficient**, spoken as *n* choose *k*

The Binomial and Beta Distributions

How can we express the **likelihood** for the trials with binary outcomes?

- Use the **Beta distribution**:

$$p(x \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1},\ 0 \leq x \leq 1$$

Where the **Beta function** is,

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha, \beta)}$$

and, $\Gamma(x)$ is the Gamma function

The Binomial and Beta Distributions

The Beta distribution looks complicated!

- How can we interpret the Beta distribution?

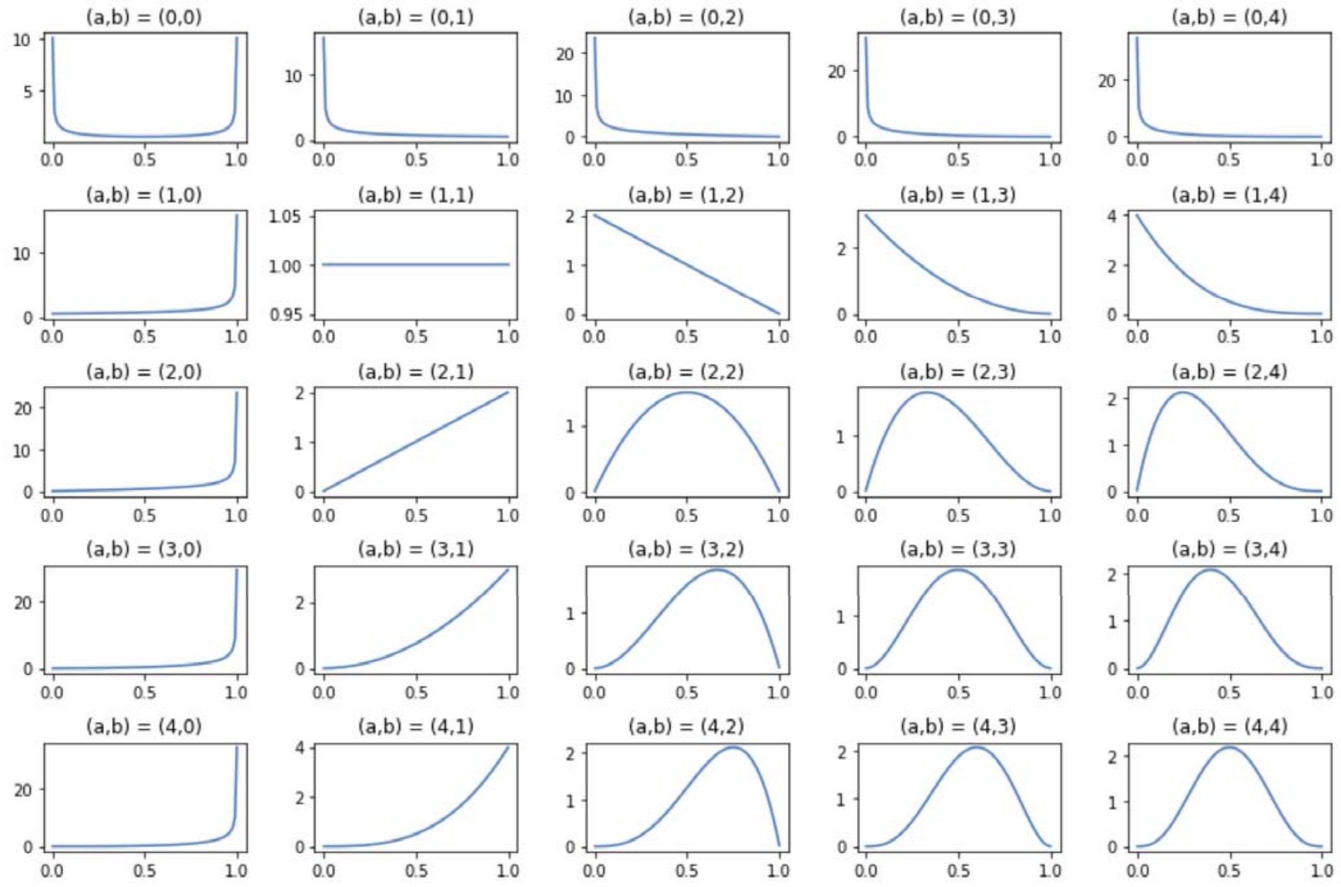
$$p(x \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1},\ 0 \leq x \leq 1$$

- Simple interpretation, α and β are counts:
  - α are counts of successes
  - β are counts of failures
  - Higher counts given narrower distribution
  - If $\alpha = \beta$, is symmetric
  - Beta distribution defined on the interval (0,1)

The Binomial and Beta Distributions

How to interpret the Beta distribution

- Distribution changes with counts



Estimation of Distribution Parameters

How can best find the parameters of a distribution?

- Two possible approaches
    - Give same results in limit of large dataset
- **Maximum likelihood** finds model parameter that maximizes likelihood of distribution
    - Can use the **Kullback-Leibler (KL) divergence** between the estimated distribution and the data – information theoretic approach
    - Can find root of likelihood function by taking derivatives
- Use Bayesian estimation to find MAP or posterior distribution of parameter
    - Must specify a prior distribution

The Kullback-Leibler Divergence

How can we measure the fit of a likelihood function?

- One possibility is the Kullback-Leibler (KL) divergence between an estimated distribution and the actual distribution
    - **Information theoretic** approach
- Start with **Shannon entropy**:

$$\mathbb{H}(I) = E[I(X)] = E[-ln_b(P(X))] = -\sum_{i=1}^{n} P(x_i)ln_b(P(x_i))$$

- Shannon entropy measures the **information content** of a probability distribution
- But we need to compare two distributions

The Kullback-Leibler Divergence

How can we measure the fit of a likelihood function?

- The K-L divergence is between two distributions *p(x)* and *q(x)* is defined:

$$\mathbb{D}_{KL}(P \parallel Q) = -\sum_{i=1}^{n} p(x_i)\, ln_b \frac{p(x_i)}{q(x_i)}$$

- The K-L divergence is 0 if *p(x) = q(x)*
- K-L divergence increases the more *p(x)* differs from *q(x)*
- *But, K-L divergence is **not symmetric***

The Kullback-Leibler Divergence

How can we measure the fit of a likelihood function?

- The **cross entropy** is defined:

$$\mathbb{H}(P,Q) = -\sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

- We can then rewrite the K-L divergence as:

$$\mathbb{D}_{KL}(P \parallel Q) = \sum_{i=1}^{n} p(x_i)\ ln_b p(x_i) - \sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

$$\mathbb{D}_{KL}(P \parallel Q) = \mathbb{H}(P) + \mathbb{H}(P,Q)$$

$$\mathbb{D}_{KL}(P \parallel Q) = Entropy(P) + Cross\ Entropy(P,Q)$$

The Kullback-Leibler Divergence

How can we measure the fit of a likelihood function?

- Start with the expansion of the K-L divergence:

$$\mathbb{D}_{KL}(P \parallel Q) = \sum_{i=1}^{n} p(x_i)\ ln_b p(x_i) - \sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

$$\mathbb{D}_{KL}(P \parallel Q) = \mathbb{H}(P) + \mathbb{H}(P, Q)$$

- When comparing distributions, $\mathbb{H}(P)$ is constant for any reference distribution *p(x)*
- We only need to deal with $\mathbb{H}(P, Q)$ to compare the two distributions

The Kullback-Leibler Divergence

How can we measure the fit of a likelihood function?

- We only need to deal with $\mathbb{H}(P, Q)$ to compare the two distributions

$$\mathbb{H}(P, Q) = -\sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

- The problem is, we never will know *p(x)*
- So how can we compute *p(x)*?
- We can use an approximation:

$$\mathbb{H}(P, Q) = -\frac{1}{N}\sum_{i=1}^{n} ln_b q(x_i)$$

The Kullback-Leibler Divergence

Maximum likelihood with K-L divergence

- For a generating distribution p*(x) we can write the K-L divergence with the approximation q(x):

$$\mathbb{D}_{KL}(p^* \parallel q) = \sum_{i=1}^{n} p^*(x_i)\, ln_b \frac{p^*(x_i)}{q(x_i)} = -\mathbb{H}(p^*) - \mathbb{E}_{x \sim p^*}[log\ q(x)]$$

- The first term only depends on p* and does not depend on q(x)
- We want to minimize K-L divergence, so maximize $\mathbb{E}_{x \sim p^*}[log\ q(x)]$
- But, the expectation requires generating data from p*
- Complication: we will never know p*!

The Kullback-Leibler Divergence

Maximum likelihood with K-L divergence

- Complication: we will never know p*!
- We can use a Monte Carlo like approximation:

$$\mathbb{E}_{x\sim p^*}[log\ q(x)] = \frac{1}{|D|}\sum_{x\in D} log\ q(x)$$

Where D is the dataset drawn from the distribution p*

- The maximum likelihood can be expressed:

$$\max_{p\in D} \frac{1}{|D|}\sum_{x\in D} log\ q(x)$$

- This approximation minimizes K-L divergence!

Maximum Likelihood for the Binomial Distribution

Maximum likelihood for model parameter Θ

- How can we apply the maximum likelihood method to find the parameter of the Binomial distribution?
- Start with the likelihood function $\mathcal{L}(\Theta, \mathcal{V}) = p(\mathcal{V} \mid \Theta)$
- Using the log likelihood we wish to find Θ such that:

$$\hat{\Theta} \doteq argmax_{\Theta}\{log(p(\mathcal{V} \mid \Theta))\}$$

Maximum Likelihood for the Binomial Distribution

Maximum likelihood for model parameter Θ

- For the Binomial distribution the likelihood is:

$$L(\Theta) = \prod_{i=1}^{n} \Theta^{\nu_i} \, (1 - \Theta)^{(1-\nu_i)})$$

Where ν is the data vector

- The log-likelihood is then:

$$\log(L(\Theta)) = \log(\Theta) \sum_{i=1}^{n} \nu_i + \log(1 - \Theta) \sum_{i=1}^{n} (1 - \nu_i)$$

Maximum Likelihood for the Binomial Distribution

Maximum likelihood for model parameter Θ

- Setting the derivative of the log-likelihood to 0:

$$\frac{\partial l(\Theta)}{\partial \Theta} = \frac{\sum_{i=1}^{n} v_i}{\Theta} - \frac{\sum_{i=1}^{n}(1 - v_i)}{1 - \Theta} = 0$$

- With solution:

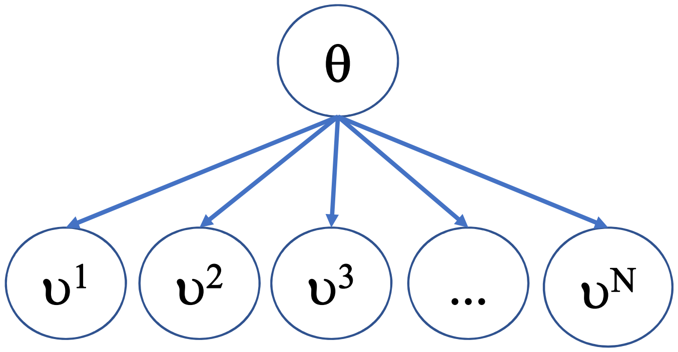
$$\Theta = \frac{1}{n}\sum_{i=1}^{n} v_i$$

- Or in simple terms:

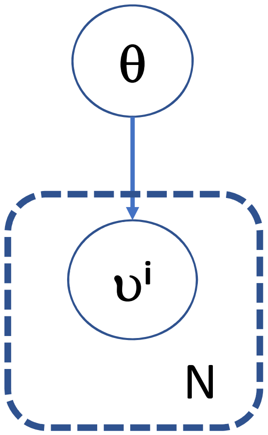
$$\Theta = \frac{count\ of\ successes}{total\ count}$$

Bayesian Estimation for Binomial Distribution

How to perform Bayesian estimation on the model parameter Θ?



- Start with a vector of observations
  $\vec{\nu} = \{\nu^1, \nu^2, \nu^3, \ldots \nu^N\}$
- We can represent the generating distribution as a directed graphical model with parameter θ

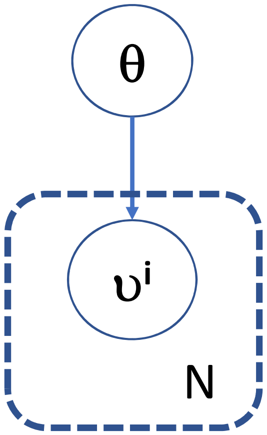


- Or, summarize with **plate notation**

Bayesian Estimation for Binomial Distribution

How to perform Bayesian estimation on the model parameter Θ?

- The posterior distribution of Θ given the data vector $\vec{\nu}$ can be written:



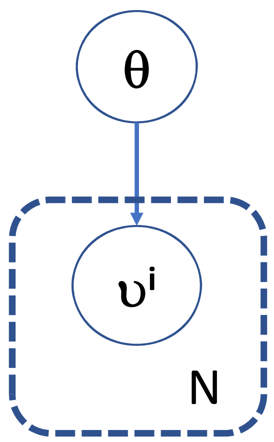
$$p(\Theta \mid \nu^1, \nu^2, \nu^3, \dots \nu^N) \propto p(\Theta) \prod_{n=1}^{N} p(\nu^n \mid \Theta)$$

$$\propto p(\Theta) \prod_{n=1}^{N} \Theta^{I(\nu^n=1)} (1-\Theta)^{I(\nu^n=0)}$$

$$\propto p(\Theta)\ \Theta^{\sum_{n=1}^{N} I(\nu^n=1)} (1-\Theta)^{\sum_{n=1}^{N} I(\nu^n=0)}$$

Bayesian Estimation for Binomial Distribution

How to perform Bayesian estimation on the model parameter Θ?



- Now, define the binary result {True, False}, we can define the posterior distribution of Θ:

$$p(\Theta \mid v^1, v^2, v^3, \ldots v^N) \propto p(\Theta)\, \Theta^{N_T} (1-\Theta)^{N_F}$$

- Or, given a count operator *C* we can write the likelihood as:

$$\Theta^{C(v_i=1)}(1-\Theta)^{C(v_i=0)}$$

- But how do we express the prior distribution $P(\Theta)$?

Bayesian Estimation for Binomial Distribution

How to perform Bayesian estimation on the model parameter Θ?

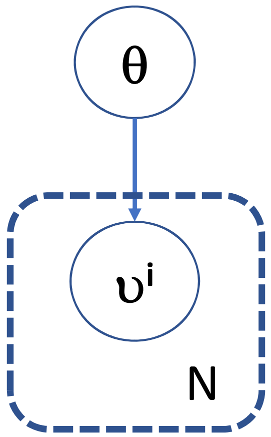
- How do we express the prior distribution $P(\Theta)$?
- Recall, the Beta distribution is the **conjugate** of the Binomial distribution and has the form:

$$p(x \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

- The product of a distribution and its conjugate has the same family as the distribution
- Define a prior using **pseudo counts**
- Set the parameter α to the number of ‘prior successes’
- Set the parameter β to the number of ‘prior failures’

Bayesian Estimation for Binomial Distribution

How to perform Bayesian estimation on the model parameter Θ?



- How do we express the prior distribution $P(\Theta)$?
- Using the pseudo counts for the prior, the posterior of q becomes:

$$p(\Theta \mid v^1, v^2, v^3, \ldots v^N) \propto \Theta^{\left(\sum_{n=1}^{N} I(v^n=1)\right)+\alpha}(1-\Theta)^{\left(\sum_{n=1}^{N} I(v^n=0)\right)+\beta}$$

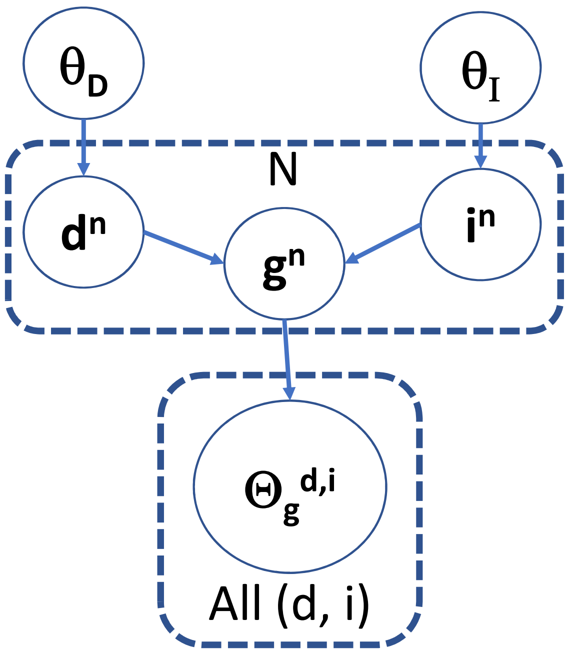
- And, the **Maximum A Posteriori (MAP)** value of θ is:

$$\Theta = \frac{count\ of\ successes + \alpha}{total\ count + \alpha + \beta}$$

Bayesian Estimation for Binomial Distribution

Properties of the prior distribution of Θ given pseudo counts

- The higher the ratio α/β the larger the prior value of Θ
- The larger the sum of α and β the stronger the prior. Small values of α and β specify a vague prior
- The larger the actual counts, the less important the prior in determining Θ
- If $\alpha = \beta = 0$ the Bayesian estimate of Θ is identical to the maximum likelihood estimate
- If there are no actual counts, the MAP estimate of Θ is the prior
- The prior regularizes the estimation of Θ, even for variables with no observations


θ_D
θ_I
N
d^n
g^n
i^n
$\Theta_g^{d,i}$
All (d, i)

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- Use the categorical distribution for multi-category variables
    - Sometimes called (incorrectly) the multinomial distribution
    - k category distribution parameterized by k probabilities
    - Is the multi-class generalization fo the Binomial distribution
- The Dirichlet distribution is the conjugate of the categorical distribution
    - Is the multi-class extension of the Beta distribution

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- The categorical distribution, where *x* is in the *i*th category, is expressed:

$$f(x = i \mid p) = p_i$$

Where for k categories have probabilities:

$$p = (p_1, \ldots, p_k)$$

and with the constraint,

$$\sum_{i=1}^{k} p_i = 1.0$$

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- Use **Iverson bracket notation**:

  $[x = i] = 1$ or 0 otherwise

- This notation gives a compact way to write the categorical distribution:

$$f(x = i \mid p) = \prod_{i=1}^{k} p_i^{[x=i]}$$

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- What is the maximum likelihood estimate (MLE) of the probability of each of the k categories, $p = (p_1, \ldots, p_k)$?
- Given the counts for each of the categories, $X = (c_1, c_2, \ldots, c_n)$, the MLE is:

$$E(p_i \mid X) = \frac{c_i}{N}$$

- The above is a generalization of the MLE for the Binomial distribution

Categorical and Dirichlet Distributions

The Dirichlet distribution is the conjugate of the categorical

- Dirichlet distribution is the generalization of the Beta distribution for multiple categories
- For n categories $x_1, x_2, \ldots, x_n$, and **concentration parameters** $\alpha_1, \alpha_2, \ldots, \alpha_n$
- Concentration parameters specify how **concentrated the distribution is** for each category
- Express the Dirichlet distribution:

$$f(x_1, x_2, \ldots, x_n; \alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1}$$

Categorical and Dirichlet Distributions

The Dirichlet distribution is the conjugate of the categorical

- Express the Dirichlet distribution:

$$f(x_1, x_2, \ldots, x_n; \alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1}$$

Where the Beta function in terms of gamma (Γ) functions is;

$$B(\alpha) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\sum_{i=1}^{n} \Gamma(\alpha_i)}$$

And with constraints,

$$\sum_{i=1}^{n} x_i = 1$$

$$x_i \geq 0$$

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- The probability parameters of the categorical distribution for each category, $p = (p_1, \ldots, p_k)$
- Bayesian MAP estimate of the parameters of the categorical distribution given the counts for each category $\mathcal{X} = (c_1, c_2, \ldots, c_n)$

$$E(p_i \mid \mathcal{X}, \boldsymbol{\alpha}) = \frac{c_i + \alpha_i}{N + \sum_k \alpha_k}$$

- Is a generalization of the expression for Binomially distributed results with Beta prior

Categorical and Dirichlet Distributions

How can we work with problems with multiple categories?

- Concentration parameters specify how **concentrated the distribution is** for each category
- The concentration pseudocounts parameters specify the prior Dirichlet distribution:
    - Categories with larger pseudocounts have higher prior probabilities
    - The ratio of the total sum of pseudocounts to the total number of actual counts expresses the strength of the prior. A larger ratio is a stronger prior
    - When all α_i are equal the prior is uniform
    - For a relatively large number or actual counts the prior has less influence and the estimates approach the MLE.

Working with Multivariate Distributions

How do we estimate the parameter vector, Θ?

- Up to now, we have only considered single parameter models
- How can we perform parameter estimation for multivariable distributions?
- Consider the joint distribution of the grade, G, the difficulty of the course, D, and the intelligence of the student, I, from the student job example:

$$p(\Theta_D, \Theta_I, \Theta_G)$$

- Three parameters, $\Theta_D, \Theta_I, \Theta_G$, must be estimated simultaneously
- Simplify this process by the **global independence assumption**
- The global independence assumption **decouples the estimation of the parameter**

Working with Multivariate Distributions

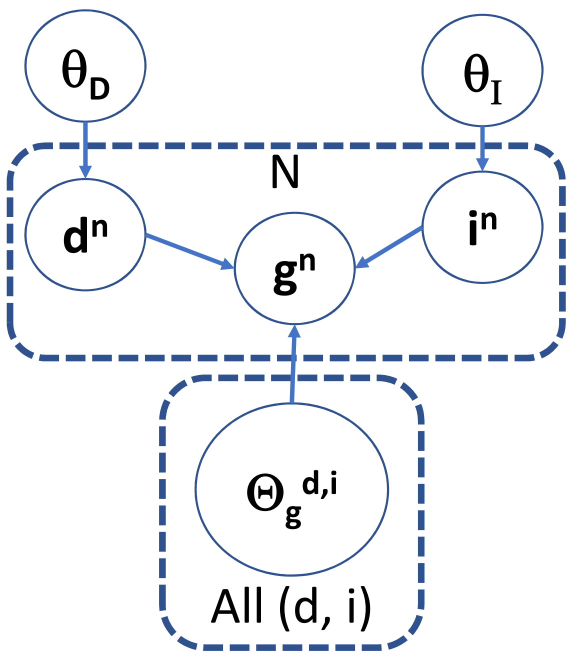
How do we estimate the parameter vector, $\boldsymbol{\Theta}$?

- With the global independence assumption, fit the parameters for of the factorized distribution:

$$p(\Theta_D, \Theta_I, \Theta_G) = p(\Theta_D)p(\Theta_I)p(\Theta_G)$$

- Use the N data samples, $X = (c_1, c_2, \ldots, c_n)$

Working with Multivariate Distributions



How do we estimate the parameter vector, Θ?

- For factorized distribution:

$$p(\Theta_D, \Theta_I, \Theta_G) = p(\Theta_D)p(\Theta_I)p(\Theta_G)$$

- Use plate notation to show model for estimation from N samples
- If all samples are iid the factorization represented by the Dag becomes:

$$p(\Theta_D, \Theta_I, \Theta_G) = p(\Theta_D)p(\Theta_I)p(\Theta_G)\prod_{n=1}^{N} p(d^n \mid \Theta_D)p(i^n \mid \Theta_I)p(g^n \mid d^n, i^n, \Theta_G)$$

- In summary, we can fit the model by **independently maximizing the likelihood**

Part 2
Learning Model Structure

HARVARD
Extension School

Learning structure of DAGs

Methods to find the structure of a DAG?

- Focus on learning structure of DAGs from data
- Learn structure from data?
- Use expert opinion?
- Combination of both?

Learning structure of DAGs

How do we find the structure of a DAG?

- Expert opinion is both useful and problematic
  - Involves subjective judgement
  - May not have agreement between experts
  - Hard for complex problems
- Expert option may provide useful constraint or prior

Learning structure of DAGs

How do we find the structure of a DAG?

- Is an unsupervised learning problem
  - Used as a data mining method
  - No marked cases
  - Results **hard to evaluate**
- Typically, requires massive amounts of data
- Significant bias-variance trade-off
  - Model with high complexity or low bias is overfit with high variance
  - Model with low complexity or high bias is underfit with low variance

Learning structure of DAGs

How do we find the structure of a DAG?

- Algorithms have two components
- Define the score metric
  - Score metric must reflect trade-off between fit (low variance) and complexity
  - **Likelihood** based score
  - **Bayesian** score
  - **Information theortic**
- A search problem looking for the optimal score
  - Search for a good solution is a classic AI problem

Learning structure of DAGs

How can we score a model?

- Score metric must reflect trade-off between fit (low variance) and complexity
- Given a graph, $\mathcal{G}$, and data D a general likelihood formulation is:

$$Score(\mathcal{G} : D) = log\Big(\mathcal{L}(\mathcal{G} : D)\Big) - \phi(n) \parallel \mathcal{G} \parallel$$

- Where,

$$\mathcal{L}(\mathcal{G} : D) = likelihood\ given\ the\ fitted\ graph\ parameters$$

$\|\mathcal{G}\| = number\ of\ parameters\ in\ \mathcal{G}$, a measure of complexity

$$n = number\ of\ samples$$

$\phi(n)$ = a function to adjust complexity for the amount of data

Learning structure of DAGs

How can we score a model?

- Score metric must reflect trade-off between fit (low variance) and complexity

$$Score(\mathcal{G} : D) = log\Big(\mathcal{L}(\mathcal{G} : D)\Big) - \phi(n) \parallel \mathcal{G}\parallel$$

- Maximizing the first term, $log\Big(\mathcal{L}(\mathcal{G} : D)\Big)$, minimizes bias, but increases variance
- The second term, $-\ \phi(n) \parallel \mathcal{G}\parallel$, penalizes variance at the expense of bias

Learning structure of DAGs

How can we score a model?

- The Bayesian Information Criteria, BIC, is one such metric
- Also know as Schwartz Information Criteria
- BIC is related to the Akaki Information Criteria, AIC
- Given a graph, $\mathcal{G}$, and data D the BIC is:

$$BIC = ln(n) \parallel G \parallel -2\, ln\Big(\mathcal{L}(\mathcal{G} : D)\Big)$$

- Where,

$$\mathcal{L}(\mathcal{G} : D) = likelihood\ given\ the\ fitted\ graph\ parameters$$

$\|\mathcal{G}\| = number\ of\ parameters\ in\ \mathcal{G}$, a measure of complexity

$\phi(n)$= *ln(ϕ)* = adjust complexity for the amount of data

Learning structure of DAGs

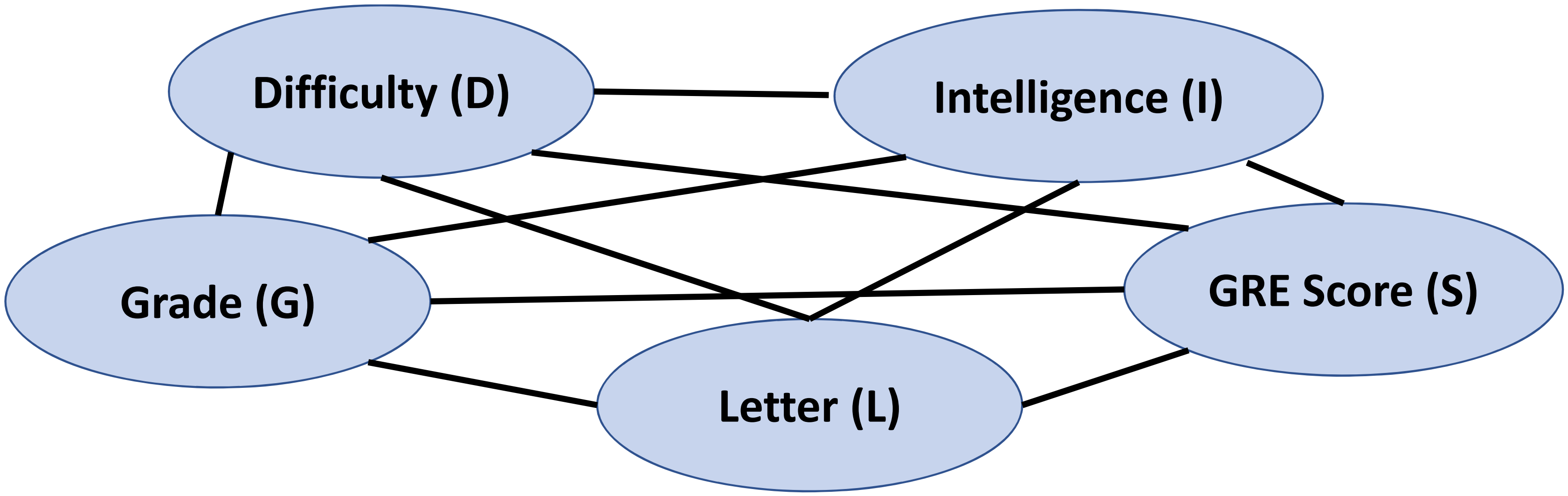
How can we score a model?

- Properties of the Bayesian Information Criteria, BIC:
- The **higher the likelihood, the lower the BIC**
- Models with **lower BIC** are considered **better**
- Models with **more parameters** have **higher BIC**
- **BIC penalizes complex models** and prefers simple models with few edges
- But, name is misleading; Despite the name, this is **not a Bayesian method!**
  - No prior
  - Computes a score, not a probability

Learning structure of DAGs

How to perform the search to optimize the model score

- Consider a simple example of possible edges in a graph



- **Exhaustive search** across all possible n edges is computationally prohibitive:

$$C(n) = n^{O(2^{O(n)})}$$

Learning structure of DAGs

How to perform the search to optimize the model score

- Exhaustive search is not an option, so what can we do?
- Apply **search heuristics**!
- AI researchers have worked on search heuristics for decades
    - For a comprehensive discussion of search heuristics for AI see chapters 3 and 4 of Russell and Norvig, third edition
- Examples of search heuristics used for finding DAG structure
    - **Hill climbing search**
    - **Taboo search**
- Search can work in **two possible starting points:**
    - A fully connected network, wherein edges are removed
    - Competly unconnected network, wherein edges are added

Learning structure of DAGs

How to perform the search to optimize the model score

- Hill climb search is a **greedy search algorithm** for **discrete valued problems**
  - Not to be confused with gradient ascent/descent for continuous valued problems
  - Both are **locally greedy** search methods
- Proceeds through a series of discrete states to find the optimal or terminal state
- Next state found by a single addition or removal of edge – **local search**
- Search continues until the improvement in the objective function is less than some threshold

Learning structure of DAGs

How to perform the search to optimize the model score

- Hill climb search is a **greedy search algorithm** and can become **stuck at a local optimum**
  - **Multiple random starts** help better explore the state (solution) space
  - Using a **taboo list** of last N steps limits backtracking and encourages exploration of space
- Commonly used improvements to the basic algorithm include:
  - Closest state which improves the objective function is taken as the next step
  - Successor state with the largest possible improvement is taken at each step

Learning structure of DAGs

How to perform the search to optimize the model score

- **Taboo search** was developed in the 1980s
- Keeps a global **taboo list** of states already visited to prevent wasted search cycles
- Can take **random exploration steps** if no local improvement found
- Often faster than hill climbing
- But taboo list can use too much memory for larger problems

Other Approaches to Learning structure of DAGs

The K2 score and K2 algorithm are a Bayesian approach

- The **K2 algorithm** uses locally greedy search; e.g. hill climbing
- Steps of the K2 algorithm:
    1. Node order that search follows is determined - NP hard problem!
    2. Begin search with first node in the order; initially has no parents
    3. Add parents incrementally in order that maximizes score increase
    4. The search terminates when the score no longer increases
- K2 algorithm requires no constraint on the number of parents

Other Approaches to Learning structure of DAGs

The K2 score and K2 algorithm are a Bayesian approach

- Success of K2 algorithm is dependent on node ordering
- Poor ordering leads to poor results
- Random starts on determining the ordering can improve results
- See paper by Learner and Malka for complete details: http://www.ee.bgu.ac.il/~boaz/LernerMalkaAAI2011.pdf

Other Approaches to Learning structure of DAGs

The K2 score and K2 algorithm are a Bayesian approach

- Need a score method for the K2 algorithm
- For a graph, $\mathcal{G}$, with n nodes and data, D, sampled from the joint distribution, the Bayesian score is:

$$p(D \mid \mathcal{G}) = \frac{p(\mathcal{G} \mid D)p(\mathcal{G})}{p(D)} = \frac{p(\mathcal{G}, D)}{p(D)}$$

- The prior distribution, $p(\mathcal{G})$, is typically uniform Dirichlet

Other Approaches to Learning structure of DAGs

The K2 score and K2 algorithm are a Bayesian approach

- Since the data is the same for every model, $p(D)$, is the same
- Assuming the parameters associated with each variable are independent gives the decomposition:

$$p(\mathcal{G}, D) = p(\mathcal{G}) \prod_{i=1}^{n} g(d_i, \mathbf{P}_{\mathbf{a_i}})$$

Where, $g(d_i, \mathbf{P}_{\mathbf{a_i}})$ is the subscore for the *i*th dimension

- Allows the algorithm work locally, variable by variable

A Bit of Information Theory

Start by defining **Shannon Entropy**?

$$\mathbb{H}(I) = E[I(X)]$$

Where: $E[X] =$ the expectation of X.

$I(X) =$ the information content of X.

But, we work with probability distributions, so:

$$\mathbb{H}(I) = E[-ln_b(P(X))] = -\sum_{i=1}^{n} P(x_i)ln_b(P(x_i)$$

Where: $P(X) =$ probability of X.

$b =$ base of the logarithm.

A Bit of Information Theory

- We need to measure the difference between the distribution of our approximation and the distribution of the data
- The **Kullback-Leibler divergence** between **two distributions P(X) and Q(X)** is such a measure**:**

$$\mathbb{D}_{KL}(P \parallel Q) = -\sum_{i=1}^{n} p(x_i)\; ln_b \frac{p(x_i)}{q(x_i)}$$

A Bit of Information Theory

- How do we compute KL divergence?
- If we knew P(X) we would not need to compute KL divergence
- We can expand KL divergence as:

$$\mathbb{D}_{KL}(P \parallel Q) = \sum_{i=1}^{n} p(x_i)\ ln_b p(x_i) - \sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

$$\mathbb{D}_{KL}(P \parallel Q) = \mathbb{H}(P) + \mathbb{H}(P, Q)$$

$$\mathbb{D}_{KL}(P \parallel Q) = Entropy(P) + Cross\ Entropy(P, Q)$$

A Bit of Information Theory

Given: $\mathbb{D}_{KL}(P \parallel Q) = \mathbb{H}(P) + \mathbb{H}(P, Q)$

The term $\mathbb{H}(P)$ is constant

So, we only need the **cross entropy** term:

$$\mathbb{H}(P, Q) = -\sum_{i=1}^{n} p(x_i)\; ln_b q(x_i)$$

A Bit of Information Theory

How can we compute cross entropy when we don’t know P(X):

$$\mathbb{H}(P, Q) = -\sum_{i=1}^{n} p(x_i)\ ln_b q(x_i)$$

Since we don’t know P(X), use the approximation:

$$\mathbb{H}(P, Q) = -\frac{1}{N}\sum_{i=1}^{n} ln_b q(x_i)$$

Mutual Information for K2 Scoring

The K2 score is based on mutual information

- Mutual information is defined:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)} \right)$$

- If the two distributions are identical then:

$$log\left(\frac{p(x,y)}{p(x)p(y)} \right) = log(1) = 0$$

- In words, there is no mutual information between identical distributions
- Mutual information is **always greater than 0**; $I(X;Y) \geq 0$
- Mutual information is **symmetric**; $I(X;Y) = I(Y;X$

Mutual Information for K2 Scoring

The K2 score is based on mutual information

- Relate mutual information to **entropy** and **conditional entropy**:

$$\begin{aligned} I(X;Y) &\equiv H(X) - H(X \mid Y) \equiv H(Y) - H(Y \mid X) \\ &\equiv H(X) + H(Y) - H(X,Y) \end{aligned}$$

- Can also relate mutual information to the **KL divergence between the joint distribution and the product of the distributions of the variables**

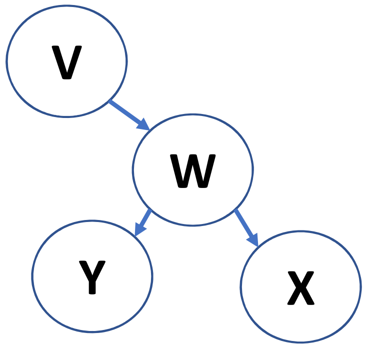
$$I(X;Y) = D_{KL}\left(p(x,y) \parallel p(x)\,p(y)\right)$$

- In words, the mutual information between two variables is **0** if the variables are **independent**

Chow-Liu Tree Algorithm

Use information theory to find node order

- **Chow-Li tree algorithm** uses information theory to find a **tree structure graph with a factorized distribution** that best fits the **data distribution**
- But, this problem is **highly under-constrained!**
- In first order tree structure graph a node can only have **one parent:**



First Order Tree

Chow-Liu Tree Algorithm

Use information theory to find node order

- Express K-L divergence and mutual information using using expectation over a distribution, $\mathbb{E}$
- K-L divergence, given data *D*:

$$KL(p \parallel q) = \mathbb{E}_{p(x)}\Big(log(p(x))\Big) - \sum_{i=1}^{D} \mathbb{E}_{p(x_i, x_{pa(i)})}\Big(log(q(x_i \mid x_{pa(i)}))\Big)$$

- And, mutual information (MI) for two variables, xi, xj:

$$MI(x_i, x_j) = \mathbb{E}_{p(x)}\left(log\Big(\frac{p(x_i, p_j)}{p(x_i)\ p(x_j)}\Big)\right)$$

- Thus, K-L divergence can be formulated in terms of MI:

$$KL(p \parallel q) = -\sum_{i=1}^{D} MI(x_i; x_{pa(i)}) - \mathbb{E}_{p(x_i)}\Big(log(p(x_i))\Big) + Constant$$

Chow-Liu Tree Algorithm

Use information theory to find node order

- K-L divergence in terms of MI:

$$KL(p \parallel q) = -\sum_{i=1}^{D} MI(x_i; x_{pa(i)}) - \mathbb{E}_{p(x_i)}\Big(log(p(x_i))\Big) + Constant$$

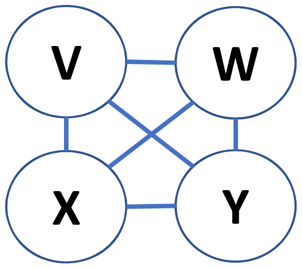
- The second term depends only on $p(x_i)$, and third term is constant
- So, minimizing the K-L divergence is the same as **maximizing the sum of MI with respect to the parents**

$$\sum_{i=1}^{D} MI(x_i; x_{pa(i)})$$

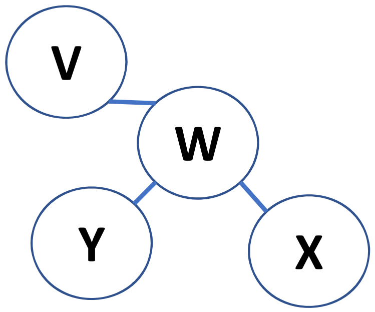
- But for any finite data set the MI is unlikely to be exactly 0, so must **constrain problem to first order tree**

Chow-Liu Tree Algorithm

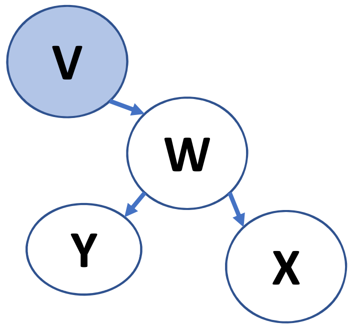
Steps of the Chow-Lui algorithm



1. Compute the MI for all edges in fully connected network



2. Compute the **maximal weight spanning tree** that connects are vertices; this is a hard problem by itself!



3. Pick an arbitrary **root** and assign directions to the arrows radiating out from the root

Monte Carlo for Structure Learning

State of the art structure learning

- Resampling methods test a great many models
- Bayesian methods use Markov Chain Monte Carlo (MCMC) sampling
  - Find models with high scores
  - Prior imposes known order
  - Compare posterior distributions of models
  - But, computationally intensive
- See Friedman and Koller, 2003 for more details
  http://people.ee.duke.edu/~lcarin/BayesianNetworkStructure.pdf

Vocabulary

- The K-L divergence is between two distributions *p(x)* and *q(x)* is defined:

$$\mathbb{D}_{KL}(P \parallel Q) = -\sum_{i=1}^{n} p(x_i)\, ln_b \frac{p(x_i)}{q(x_i)}$$

- The **cross entropy** is defined:

$$\mathbb{H}(P, Q) = -\sum_{i=1}^{n} p(x_i)\, ln_b q(x_i)$$

- **Shannon Entropy** $\mathbb{H}(I) = E[I(X)]$

Vocabulary

- Mutual information is defined: $$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)} \right)$$
- **Binomial distribution** determines the probability of *k* successes in *n* trials, given a parameter Θ
$$p(v = k \mid \Theta) = \binom{n}{k} \Theta^k (1 - \Theta)^{n-k}$$
- Concentration parameters specify how **concentrated the distribution is** for each category
- Mutual information is defined:
$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Key Points

- Beta distribution to express the likelihood for the trials with binary outcomes
- Binomial distribution to model distribution of multiple trials
- Kullback-Leibler Divergence to measure the fit of a likelihood function
- Maximize model fit with K-L divergence:
- Can do Bayesian estimation of model param $\max_{p \in D} \frac{1}{|D|} \sum_{x \in D} log\ q(x)$ imum a posteriori value
- Categorical and Dirichlet Distributions used for problems with multiple categories
- For Multivariate Distributions with some vector of unknown parameters, we can fit the model by independently maximizing the likelihood

Key Points

- Learning DAG structure is an unsupervised learning problem typically requiring massive amounts of data and with no marked cases
  - Algorithm: score metric (ex. Bayesian Information Criteria) + search method (ex. hill climbing) + starting point (fully connected network OR completely unconnected)
    - Ex. K2 score and K2 algorithm
    - Ex. Chow-Liu Tree Algorithm
  - Multiple random starts help