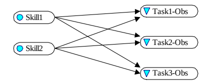
Learning CPTs

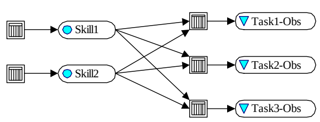
Thanks to Bob Mislevy for letting me use some of the slides from his class.

First Layer



- A simple model with two skills and 3 observables

Distributions and Variables



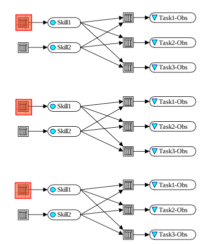
- Variables (values are person specific)
- *Distributions* provide probabilities for variables

Different People, Same Distributions

Student 1

Student 2

Student 3

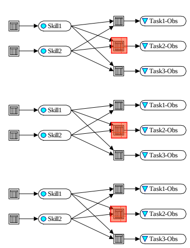


Different People, Same Distributions

Student 1

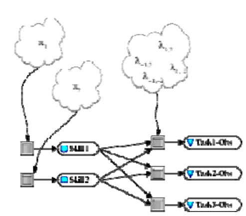
Student 2

Student 3



Second Layer

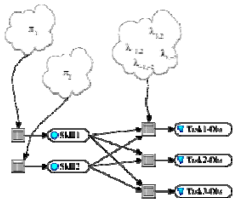
- Distributions have Parameters
- Parameters are the same across all people
- Parameters drop down into first layer to do person specific computations (e.g., scoring)



Probability distributions of parameters are called *Laws*

Second Layer

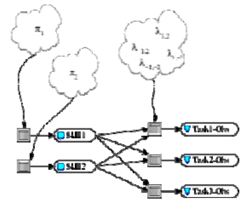
- Distributions have Parameters
- Parameters are the same across all people
- Parameters drop down into first layer to do person specific computations (e.g., scoring)



Probability distributions of parameters are called *Laws*

Second Layer

$$\pi_1 = \Pr(\bigcirc \text{Skill1})$$
$$\pi_1 = \Pr(\bigcirc \text{Skill2})$$
$$\lambda_{1,2} = \Pr(\triangledown \text{Task1} - \text{obs} | \bigcirc \text{Skill1}, \bigcirc \text{Skill2})$$
$$\lambda_{-1,2} = \Pr(\triangledown \text{Task1} - \text{obs} | \neg\bigcirc \text{Skill1}, \bigcirc \text{Skill2})$$
$$\lambda_{1,-2} = \Pr(\triangledown \text{Task1} - \text{obs} | \bigcirc \text{Skill1}, \neg\bigcirc \text{Skill2})$$
$$\lambda_{-1,-2} = \Pr(\triangledown \text{Task1} - \text{obs} | \neg\bigcirc \text{Skill1}, \neg\bigcirc \text{Skill2})$$



Speigelhalter And Lauritzen (1990)

- *Global Parameter Independence* – parameters of laws for different CPTs are independent
- *Local Parameter Independence* – parameters for laws for different rows of CPTs are independent

Under these two assumptions, the natural conjugate law of a Bayesian network is a *hyper-Dirichlet law*, a law where the probabilities on each row of each CPT follow a Dirichlet law.

Abusing the definition, we say that a CPT for which each rows is given an independent Dirichlet law follows a *hyper-Dirichlet distribution (*really should be law).

A closer look at the binomial distribution

- **Binomial.** For counts of successes in binary trials, each with probability p, in n independent trials. E.g., n coin flips, with p the common probability of heads.

The variable

Count of successes

$$p(r|\pi,n) \propto \pi^{\boxed{r}}(1-\pi)^{\boxed{n-r}}$$

Count of failures

The "success probability" parameter

The failure probability

The success probability

We will be using this as a likelihood in an example of the use of conjugate distributions.

A closer look at the Beta distribution

- **Beta.** Defined on [0,1]. Conjugate prior for the probability parameter in Bernoulli & binomial models.

```
p ~ dbeta(a,b)
```

Mean(p): $\frac{a}{a+b}$

Variance(p): $\frac{ab}{(a+b)^2(a+b+1)}$

Mode(p): $\frac{a-1}{a+b-2}$

PseudoCount of successes

PseudoCount of failures

The variable: "success probability"

$$beta(\pi|a,b) \propto \pi^{\boxed{a-1}}(1-\pi)^{\boxed{b-1}}$$

The failure probability

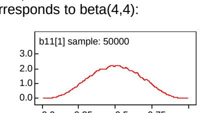
Shape, or "prior sample info"

The success probability

An example with a continuous variable: A beta-binomial example--the Prior Distribution

- The prior distribution:

Let's suppose we think it is more likely that the coin is close to fair, so π is probably nearer to .5 than it is to either 0 or 1. We don't have any reason to think it is biased toward either heads or tails, so we'll want a prior distribution that is symmetric around .5. We're not real sure about what π might be--say about as sure as only 6 observations. This corresponds to 3 pseudo-counts of H and 3 of T, which, if we want to use a beta distribution to express this belief, corresponds to beta(4,4):



An example with a continuous variable: A beta-binomial example--the Prior Distribution

- **Beta.** Defined on [0,1]. Conjugate prior for the probability parameter in Bernoulli & binomial models.

```
π ~ dbeta(4,4)
```

Mean(π): $\frac{4}{4+4} = .5$

Variance(π): $\frac{4 \cdot 4}{(4+4)^2(4+4+1)} = .028$

Mode(π): $\frac{4-1}{4+4-2} = .5$

The variable: "success probability"

PseudoCount of successes

PseudoCount of failures

$$p(\pi|\underbrace{4,4}) \propto \pi^{\boxed{4-1}} \underbrace{(1-\pi)}^{\boxed{4-1}}$$

Shape, or "prior sample info"

The success probability

The failure probability

An example with a continuous variable: A beta-binomial example--the Likelihood

- The likelihood:

  Next we will flip the coin ten times. Assuming the same true (but unknown to us) value of π is in effect for each of ten independent trials, we can use the binomial distribution to model the probability of getting any number of heads: i.e.,

The variable

Count of *observed* successes

$$p(r|\pi,10) \propto \pi^{\boxed{r}} \underbrace{(1-\pi)}^{\boxed{10-r}}$$

Count of *observed* failures

The "success probability" parameter

The success probability

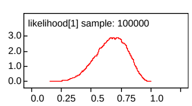
The failure probability

An example with a continuous variable: A beta-binomial example--the Likelihood

- The likelihood:

  We flip the coin ten times, and observe 7 heads; i.e., r=7. The likelihood is obtained now using the same form as in the preceding slide, except now r is fixed at 7 and we are interested in the relative value of this function at different possible values of π:

$$p(7|\pi,10) \propto \pi^7 (1-\pi)^3$$



An example with a continuous variable: Obtaining the posterior by Bayes Theorem

General form:

$$\underset{\text{posterior}}{p(y \mid x^*)} \propto \underset{\text{likelihood}}{p(x^* \mid y)} \; \underset{\text{prior}}{p(y)}$$

In our example, 7 plays the role of x*, and p plays the role of y. Before normalizing:

$$p(\pi|r=7) \propto \left[\pi^7 (1-\pi)^3\right]\left[\pi^{4-1}(1-\pi)^{4-1}\right]$$

$$= \left[\pi^{10}(1-\pi)^6\right]$$

$$= \left[\pi^{11-1}(1-\pi)^{7-1}\right]$$

This function is proportional to a beta(11,7) distribution.

An example with a continuous variable: Obtaining the posterior by Bayes Theorem

posterior

$$p(y \mid x^*) = \frac{p(x^* \mid y)\,p(y)}{\int_y p(x^* \mid y)\,p(y)\,\partial y}$$

After normalizing:

$$p(\pi|r=7) = \frac{\pi^{11-1}(1-\pi)^{7-1}}{\int_z \left[z^{11-1}(1-z)^{7-1}\right]\partial z}$$

Now, how can we get an idea of what this means we believe about π after combining our prior belief and our observations?

An example with a continuous variable: In pictures

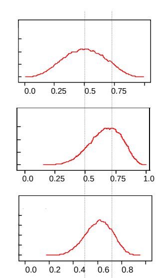
Prior

x

Likelihood

$\propto$

Posterior



Dirichlet—Categorical conjugate distribution

- Assume a variable X takes on category $1, \ldots, K$ with probabilities $\pi_1, \ldots \pi_K$
- Take N draws from this distribution and observe counts $N = X_1 + \ldots + X_K$
- Likelihood is $p(X_1, \ldots, X_K) \propto \pi_1^{X_1} \cdots \pi_K^{X_K}$
- Dirichlet Prior: $f(\pi_1, \ldots, \pi_K) \propto \pi_1^{\alpha_1 - 1} \cdots \pi_K^{\alpha_K - 1}$
- Posterior:

$$f(\pi_1, \ldots, \pi_K | X_1, \ldots, X_K) \propto \pi_1^{X_1 + \alpha_1 - 1} \cdots \pi_K^{X_K + \alpha_K - 1}$$

Updating an unconditional probability table (no parent variables)

- Prior is a table of alphas:

α_1	…	α_K

- Sum of alphas is pseudo-sample size for prior: Netica calls this Node Experience $A = \alpha_1 + \cdots + \alpha_K$
- Sufficient statistic is a table of counts in each category

X_1	…	X_K

- Posterior is an updated table

$\alpha_1 + X_1$	…	$\alpha_K + X_K$

- With updated Node Experience $A' = A + N$

Details

- Equivalent to beta-binomial when variable only takes two values
- Alphas must be positive, but don't need to be integers
- Alpha = ½ is non-informative prior
- A (sum of alphas) acts like a pseudo-sample size for the prior
- Can also write as $\alpha_k = A\pi_k^*$

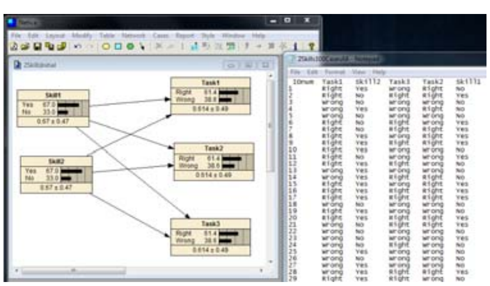
CPT updating when parents are fully observed

- Data are contingency table of child variable given parents
- Prior is a table of pseudo-counts
- Get posterior by adding them together

$$\begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1K} \\ \vdots & \ddots & \vdots \\ \alpha_{J1} & \cdots & \alpha_{JK} \end{pmatrix} + \begin{pmatrix} X_{11} & \cdots & X_{1K} \\ \vdots & \ddots & \vdots \\ X_{J1} & \cdots & X_{JK} \end{pmatrix} = \begin{pmatrix} \alpha_{11} + X_{11} & \cdots & \alpha_{1K} + X_{1K} \\ \vdots & \ddots & \vdots \\ \alpha_{J1} + X_{J1} & \cdots & \alpha_{JK} + X_{JK} \end{pmatrix}$$

Note: Both prior and posterior effective sample size (Node Experience) can be different for each row.

Netica example – fully observed



RNetica example (Ex 8.3)

- File Hyperdirchlet
- Set up network
- Two parents, one child

```
hdnet <- CreateNetwork("hyperDirchlet")
skills <-
NewDiscreteNode(hdnet,c("Skill1","Skill2"),c("High","Medium","Low"))
obs <- NewDiscreteNode(hdnet,"Observable",c("Right","Wrong"))
NodeParents(obs) <- skills
```

Set up prior for Observation

- Do this by setting CPT and NodeExperience (row pseudo-sample sizes)

```
ptab <-
data.frame(Skill1=rep(c("High","Medium","Low"),3),
Skill2=rep(c("High","Medium","Low"),each=3),
Right=c(.975,.875,.5,.875,.5,.125,.5,.125,.025),
Wrong=1-c(.975,.875,.5,.875,.5,.125,.5,.125,.025))

obs[] <- ptab
NodeExperience(obs) <- 10 #All rows equally weighted
```


Aside: Using CPTtools

- The function calcDPCframe will (among other things) calculate tables according to the DiBello—Samejima models described in the morning session.

```
## Using CPTtools
ptab1 <-
calcDPCFrame(ParentStates(obs),NodeStates(obs),
             log(c(Skill1=1.2,Skill2=.8)),0,
             rules="Compensatory")
```

- Note uses log of discrimination as parameter

Prior CPT

ptab

	Skill1	Skill2	Right	Wrong
1	High	High	0.975	0.025
2	Medium	High	0.875	0.125
3	Low	High	0.500	0.500
4	High	Medium	0.875	0.125
5	Medium	Medium	0.500	0.500
6	Low	Medium	0.125	0.875
7	High	Low	0.500	0.500
8	Medium	Low	0.125	0.875
9	Low	Low	0.025	0.975

rescaleTable(ptab,10)

	Skill1	Skill2	Right	Wrong
1	High	High	9.75	0.25
2	Medium	High	8.75	1.25
3	Low	High	5.00	5.00
4	High	Medium	8.75	1.25
5	Medium	Medium	5.00	5.00
6	Low	Medium	1.25	8.75
7	High	Low	5.00	5.00
8	Medium	Low	1.25	8.75
9	Low	Low	0.25	9.75

Netica Case files

- Text file, column separated by tabs (same as .xls files, but have .cas extension)
- One column for each observed variable (need both parents and child in this case)
- Optional IDnum column
- Optional NumCases column gives replication count
- So can either repeat out cases, or use summary counts.
- write.CaseFile() writes out a case file for use with Netica

Case Table for Ex 8.3

```
dtab <- data.frame(Skill1=rep(c("High","Medium","Low"),3,each=2),
                   Skill2=rep(c("High","Medium","Low"),each=6),
                   Observable=rep(c("Right","Wrong"),9),
                   NumCases=c(293,3,
                     112,16,
                     0,1,
                     14,1,
                     92,55,
                     4,5,
                     5,1,
                     62,156,
                     8,172))

write.CaseFile(dtab,"Ex8.3.cas")
```


Example Case File

	Skill1	Skill2	Observable	NumCases
1	High	High	Right	293
2	High	High	Wrong	3
3	Medium	High	Right	112
4	Medium	High	Wrong	16
5	Low	High	Right	0
6	Low	High	Wrong	1
7	High	Medium	Right	14
8	High	Medium	Wrong	1
9	Medium	Medium	Right	92
10	Medium	Medium	Wrong	55
11	Low	Medium	Right	4
12	Low	Medium	Wrong	5
13	High	Low	Right	5
14	High	Low	Wrong	1

Learn CPTs

- LearnCases does complete data hyper-Dirichlet updating

```
LearnCases("Ex8.3.cas",obs)
NodeExperience(obs)
               Skill2
Skill1   High Medium Low
  High    306     25  16
  Medium  138    157 228
  Low      11     19 190
```


Prior and Posterior CPTs

Prior

```
  Skill1 Skill2 Right Wrong
1   High   High 0.975 0.025
2 Medium   High 0.875 0.125
3    Low   High 0.500 0.500
4   High Medium 0.875 0.125
5 Medium Medium 0.500 0.500
6    Low Medium 0.125 0.875
7   High    Low 0.500 0.500
8 Medium    Low 0.125 0.875
9    Low    Low 0.025 0.975
```

Posterior

```
  Skill1 Skill2 Right Wrong
1   High   High 0.989 0.011
2 Medium   High 0.848 0.152
3    Low   High 0.795 0.205
4   High Medium 0.760 0.240
5 Medium Medium 0.588 0.412
6    Low Medium 0.276 0.724
7   High    Low 0.859 0.141
8 Medium    Low 0.277 0.723
9    Low    Low 0.068 0.932
```


Prior and Posterior Alphas

Prior

```
  Skill1 Skill2 Right Wrong
1   High   High  9.75  0.25
2 Medium   High  8.75  1.25
3    Low   High  5.00  5.00
4   High Medium  8.75  1.25
5 Medium Medium  5.00  5.00
6    Low Medium  1.25  8.75
7   High    Low  5.00  5.00
8 Medium    Low  1.25  8.75
9    Low    Low  0.25  9.75
```

Posterior

```
  Skill1 Skill2  Right  Wrong
1   High   High 302.75   3.25
2 Medium   High 117.00  21.00
3    Low   High   8.75   2.25
4   High Medium  19.00   6.00
5 Medium Medium  92.25  64.75
6    Low Medium   5.25  13.75
7   High    Low  13.75   2.25
8 Medium    Low  63.25 164.75
9    Low    Low  13.00 177.00
```


Problems with hyper-Dirichlet approach

- Learn more about some rows than others
- Local parameter independence assumption is unrealistic – often want CPT to be monotonic (increasing skill means increasing chance of success)
  - $\lambda_{2,2} > \lambda_{2,1} > \lambda_{1,1}$ and $\lambda_{2,2} > \lambda_{1,2} > \lambda_{1,1}$
- Solution is to use parametric models for CPT:
  - Noisy-min & Noisy-max
  - DiBello-Samejima families
  - Discrete Partial Credit families

Learning CPTs for a parametric family

- Contingency table is sufficient statistic for law for any CPT!
- Pick value of law parameters that maximize the posterior probability (or likelihood) of the observed contingency table.
- Fully Bayesian method
  - Put hyper-laws over law hyperparameters
  - Calculate observed contingency table
  - MAP estimates maximize posterior probability of contingency table
- Semi-Bayesian method
  - Use prior hyperparameters to calculate prior table.
  - Establish a pseudo-sample size for each row and calculate prior alphas
  - Do hyper-Dirichlet updating to get posterior alphas
  - MAP estimates maximize posterior probability of posterior alphas (treating them as if they were data)
  - CPTtools function `mapCPT` does this

Latent and Missing Values

- These are okay as long as they are *missing at random*
- MAR means missingness indicator is conditionally independent of the value of the missing variable given the fully observed variables
- Latent variables are always MCAR
- With other missing variables, it depends on the study design
- Can use the EM or MCMC algorithms in the presence of MAR data

EM Algorithm (Dempster, Laird & Rubin, 1977)

Key idea:

1. Pick a set of value for parameters
2. *E-step (a):* Calculate distribution for missing variables given observed variables & current parameter values.
3. *E-step (b):* Calculate expected value of sufficient statistics
4. *M-step:* Use Gradient Decent to produce MAP/MLE estimates for parameters given sufficient statistics
5. Loop 2—4 until convergence

EM algorithm details

- Only need to take a few steps of the gradient algorithm in Step 4 (Generalized EM)
- Can exploit conditional independence conditions, particularly global parameter independence (Structual EM, Meng and van Dyke)
  - Once CPT at a time
- Can be slow
  - But not as slow as MCMC
- Netica provides built-in support for special case of hyper-Dirichlet law

Expected value of missing (latent) node

- Can calculate this using ordinary Netica operations (instantiate all observed variables and read off joint beliefs)
- Instead of adding count to the table, add fractional count to the table
- Similarly use joint beliefs when more than one parent is missing

Example

- Observable X in $\{0, 1\}$; Latent θ in $\{H,M,L\}$
- Observations:
  1. $X=1;\ p(\theta) = H{:}.33,\ M{:}.33,\ L{:}.33$
  2. $X=1;\ p(\theta) = H{:}.5,\ M{:}.33,\ L{:}.2$
  3. $X=0;\ p(\theta) = H{:}.2,\ M{:}.3,\ L{:}.5$
- Expected table:

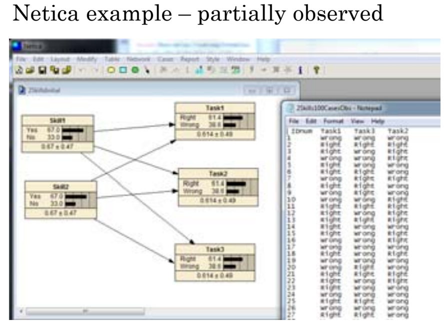
	H	M	L
1	.83	.67	.53
0	.2	.3	.5

EM for hyper-Dirichlet (RNetica LearnCPTs function)

1. Use current CPTs to calculate expected tables for all of the CPTs we are learning
2. Use the hyper-Dirichlet conjugate updating to update the CPTs
3. Loop 1 and 2 until convergence

Note: RNetica LearnCPT function currently does not reveal whether or not convergence was reached.

Netica example – partially observed



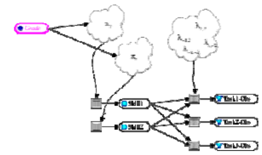
Parameterized tables

1. Use current parameters to set initial CPTs
2. Use Netica's LearnCPTs to calculate posterior tables
3. Multiple posterior tables by node experience to get pseudo-table for each CPT
4. Use gradient decent to optimize CPT parameters
5. Loop 1—4 until convergence

I'm currently working on an implementation in R (Peanut package function `GEMfit`; available from RNetica site).

Breakdown of global parameter independence

- Even if parameters are *a priori* independent, when there is missing (or latent) data then parameters are not independent *a posteriori*.
- EM algorithm only gives point estimate, does not capture this dependence
- There might also be other information which makes parameters dependent.



Markov Chain Monte Carlo (MCMC)

- In place of E-step, randomly sample values for unknown (latent & missing) variables
- In place of M-step, randomly sample values for parameters
- Takes longer than EM, but gives you an impression of the whole distribution rather than just a part.