Lecture 14

The EM algorithm to Hierarchical Models

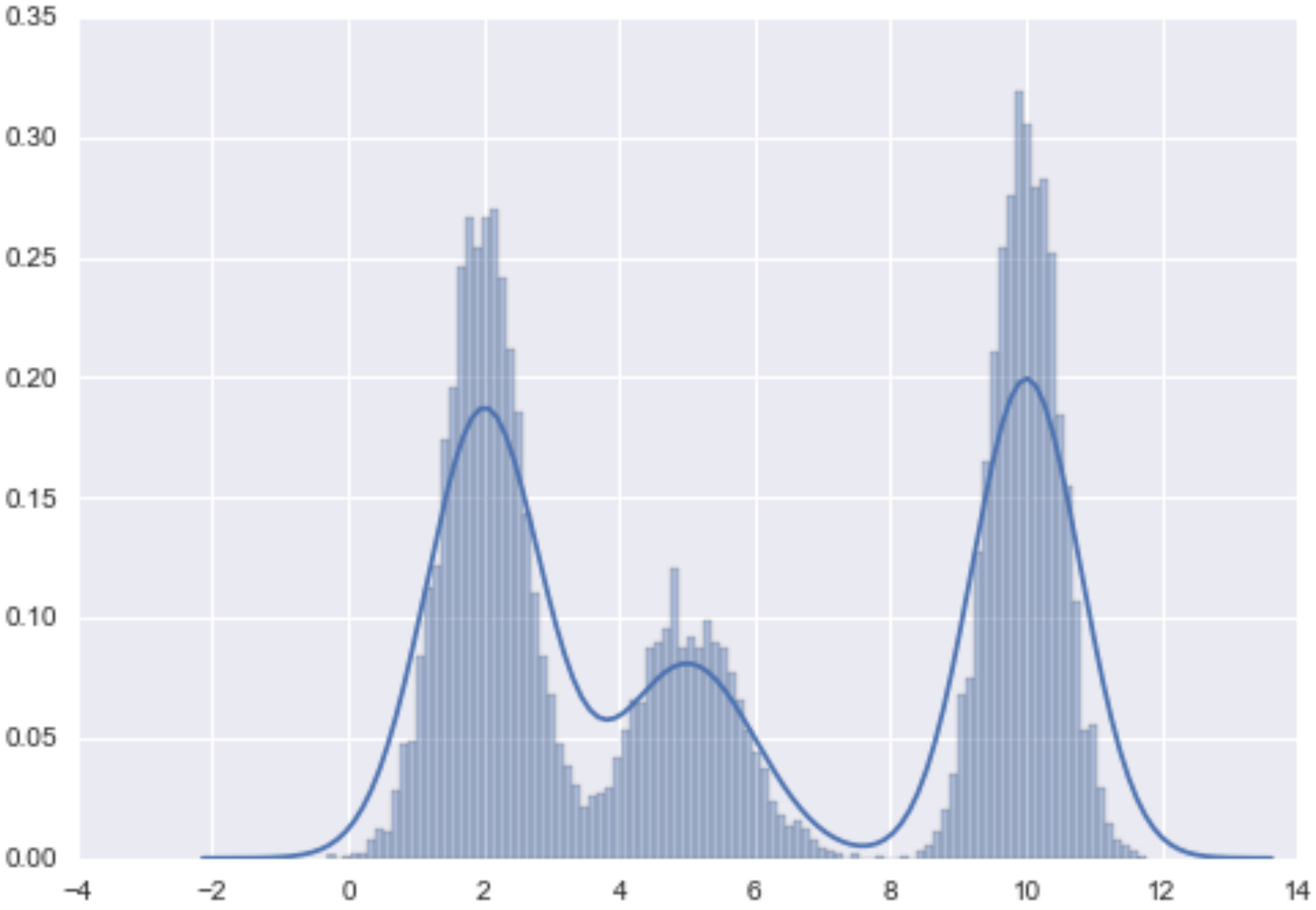
Last Time

- Latent Variables
- Mixture Models
- Supervised vs Unsupervised vs Semi-Supervised Learning
- Missing Data and the EM algorithm

Today

- EM algorithm and the mixture model
- De-Finetti's theorem
- Hierarchical models
- Empirical Bayes

Gaussian Mixture Model



$$p(x|\{\theta_k\}) = \sum_k \lambda_k N(x|\mu_k, \Sigma_k)$$

Generative:

```
mu_true = np.array([2, 5, 10])
sigma_true = np.array([0.6, 0.8, 0.5])
lambda_true = np.array([.4, .2, .4])
n = 10000

# Simulate from each distribution according to mixing proportion psi
z = multinomial.rvs(1, lambda_true, size=n) #categorical
x=np.array([np.random.normal(mu_true[i.astype('bool')][0],\
    sigma_true[i.astype('bool')][0]) for i in z])
```

```
multinomial.rvs(1,[0.6,0.1, 0.3], size=10)
array([[1, 0, 0],[0, 0, 1],...[1, 0, 0],[1, 0, 0]])
```

The two meanings of generative

Thus we **abuse** the world **generative** in two senses:

1. A way to generate data drom a data story. Here think of $\mathbf{z} = \theta$
2. A Model in which we try to figure $p(\mathbf{x}, \mathbf{z})$ or $p(\mathbf{x}|\mathbf{z})$. Here think of $\mathbf{z} = c$ or a class label.

Now lets focus on the latter. Suppose we believe their exists a "class" or representation $\mathbf{z}$. Then a dichotomy arises depending on whether $\mathbf{z}$ is observed or not.

Supervised vs Unsupervised Learning

In **Supervised Learning**, Latent Variables $\mathbf{z}$ are observed.

In other words, we can write the full-data likelihood $p(\mathbf{x}, \mathbf{z})$

In **Unsupervised Learning**, Latent Variables $\mathbf{z}$ are hidden.

We can only write the observed data likelihood:

$$p(\mathbf{x}) = \sum_{z} p(\mathbf{x}, \mathbf{z}) = \sum_{z} p(\mathbf{z}) p(\mathbf{x}|\mathbf{z})$$

GMM supervised formulation

$$Z \sim \text{Bernoulli}(\lambda)$$

$$X|z=0 \sim \mathcal{N}(\mu_0, \Sigma_0),\ X|z=1 \sim \mathcal{N}(\mu_1, \Sigma_1)$$

Full-data loglike: $l(x, z|\lambda, \mu_0, \mu_1, \Sigma) = -\sum_{i=1}^{m} \log((2\pi)^{n/2} |\Sigma|^{1/2})$

$$-\frac{1}{2} \sum_{i=1}^{m} (x - \mu_{z_i})^T \Sigma^{-1} (x - \mu_{z_i}) + \sum_{i=1}^{m} [z_i \ \log\lambda + (1 - z_i) \log(1 - \lambda)]$$

$$\mathcal{L} = \prod_i G_{1(z_i)}(x_i)^{1(z_i)}$$

$$\ell = \sum_i 1(z_i) log G_{1(z_i)}$$

Solution to MLE

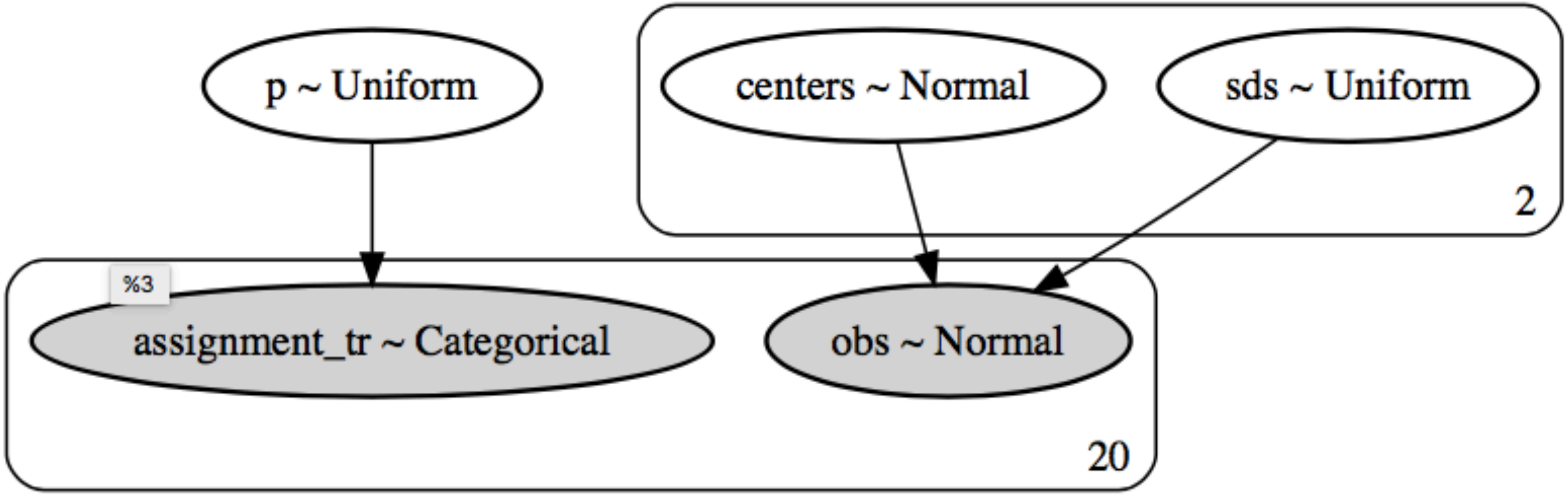
$$\lambda = \frac{1}{m}\sum_{i=1}^{m}\delta_{z_i,1}$$

$$\mu_0 = \frac{\sum_{i=1}^{m}\delta_{z_i,0}\, x_i}{\sum_{i=1}^{m}\delta_{z_i,0}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m}\delta_{z_i,1}\, x_i}{\sum_{i=1}^{m}\delta_{z_i,1}}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{z_i})(x_i - \mu_{z_i})^T$$

Supervised graph



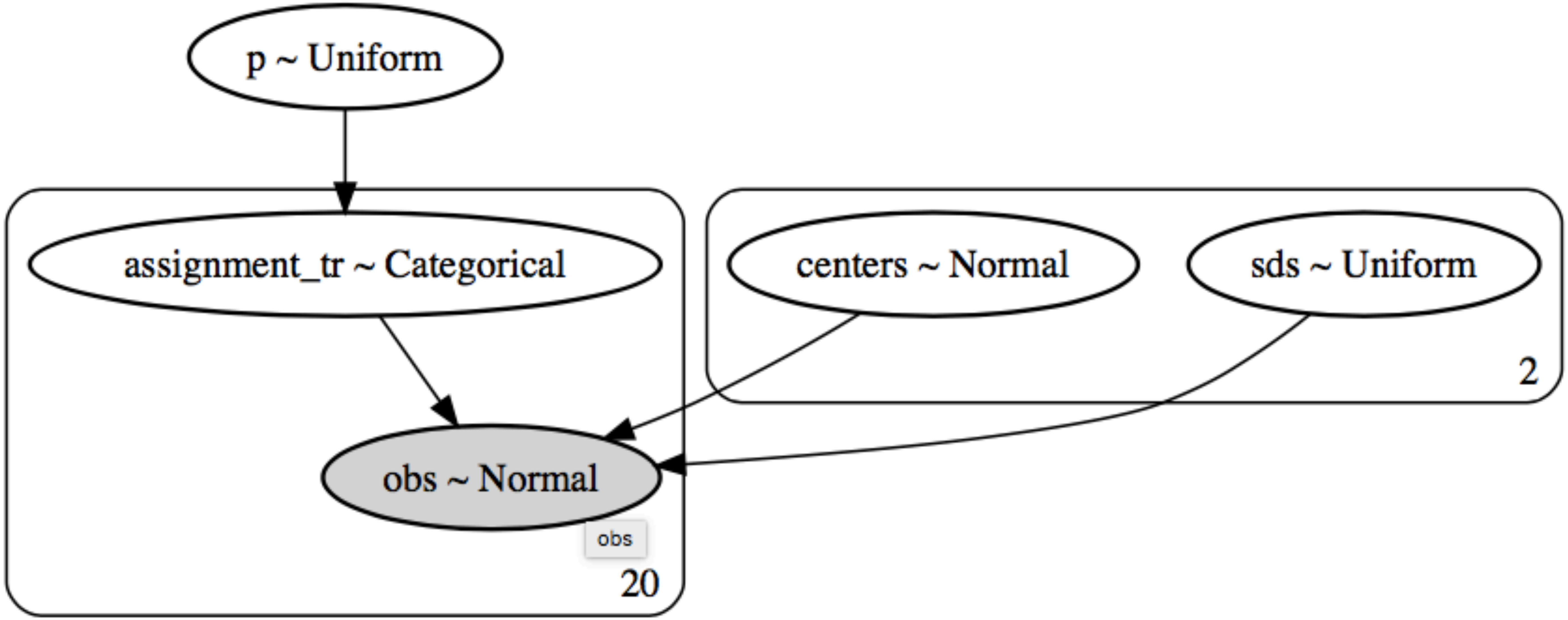
Concrete Formulation of unsupervised learning

Estimate Parameters by $\mathbf{x}$-MLE:

$$\begin{aligned} l(x \vert \lambda, \mu, \Sigma) &= \sum_{i=1}^{m} \log p(x_i \vert \lambda, \mu, \Sigma) \\ &= \sum_{i=1}^{m} \log \sum_{z} p(x_i \vert z_i, \mu, \Sigma)\, p(z_i \vert \lambda) \end{aligned}$$

Not Solvable analytically! EM and Variational. Or do MCMC.

Unsupervised graph



EXPECTATION MAXIMIZATION

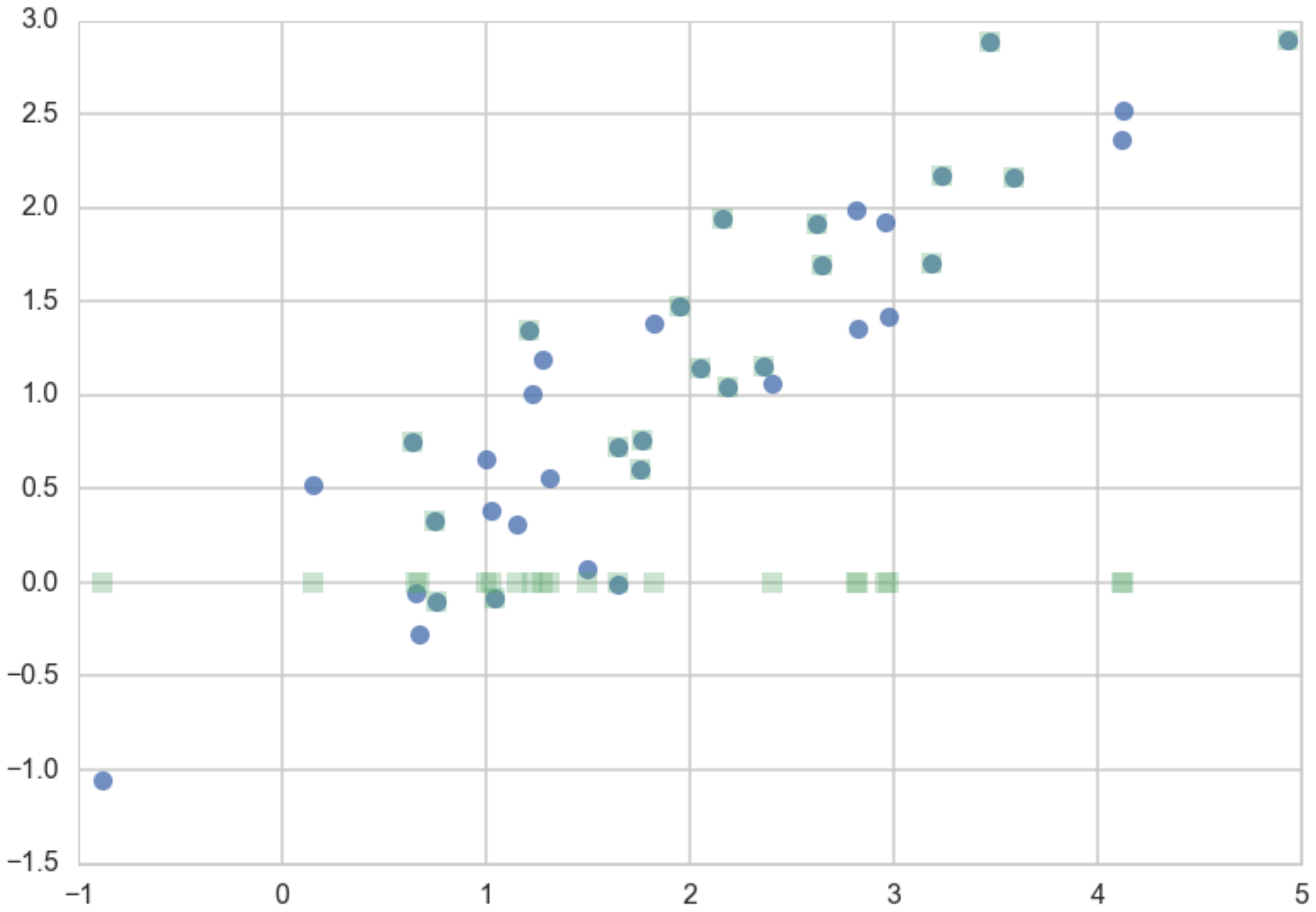
calculate MLE estimates for the incomplete data problem by using the complete-data likelihood. To create complete data, augment the observed data with manufactured data

Toy Example: 2D Gaussian

$$\begin{pmatrix} x_{1i} \\ x_{2i} \end{pmatrix} \overset{\text{ind}}{\sim} \mathcal{N}_2\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{pmatrix}\right)$$

```
sig1=1
sig2=0.75
mu1=1.85
mu2=1
rho=0.82
means=np.array([mu1, mu2])
cov = np.array([
    [sig1**2, sig1*sig2*rho],
    [sig2*sig1*rho, sig2**2]
])
```

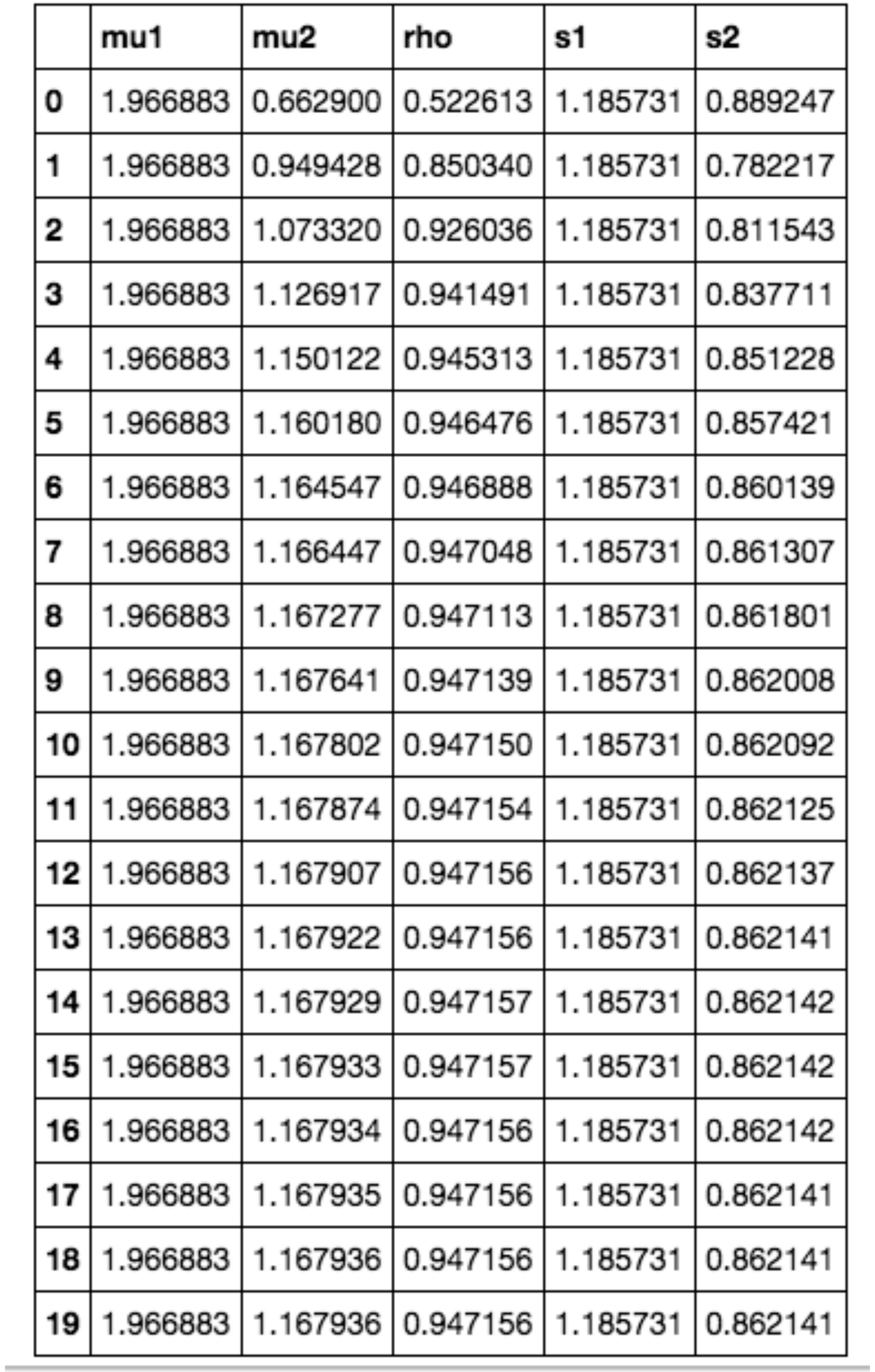
Lose z = 20 y-values. Set to 0.



Voila. We converge to stable values of our parameters. Initials:

```
sig1=1
sig2=0.75
mu1=1.85
mu2=1
rho=0.82
```

But they may not be the ones we seeded the samples with. The EM algorithm is only good upto finding local minima, and a finite sample size also means that the minimum found can be slightly different.



	mu1	mu2	rho	s1	s2
0	1.966883	0.662900	0.522613	1.185731	0.889247
1	1.966883	0.949428	0.850340	1.185731	0.782217
2	1.966883	1.073320	0.926036	1.185731	0.811543
3	1.966883	1.126917	0.941491	1.185731	0.837711
4	1.966883	1.150122	0.945313	1.185731	0.851228
5	1.966883	1.160180	0.946476	1.185731	0.857421
6	1.966883	1.164547	0.946888	1.185731	0.860139
7	1.966883	1.166447	0.947048	1.185731	0.861307
8	1.966883	1.167277	0.947113	1.185731	0.861801
9	1.966883	1.167641	0.947139	1.185731	0.862008
10	1.966883	1.167802	0.947150	1.185731	0.862092
11	1.966883	1.167874	0.947154	1.185731	0.862125
12	1.966883	1.167907	0.947156	1.185731	0.862137
13	1.966883	1.167922	0.947156	1.185731	0.862141
14	1.966883	1.167929	0.947157	1.185731	0.862142
15	1.966883	1.167933	0.947157	1.185731	0.862142
16	1.966883	1.167934	0.947156	1.185731	0.862142
17	1.966883	1.167935	0.947156	1.185731	0.862141
18	1.966883	1.167936	0.947156	1.185731	0.862141
19	1.966883	1.167936	0.947156	1.185731	0.862141

The EM algorithm, conceptually

- iterative method for maximizing difficult likelihood (or posterior) problems, first introduced by Dempster, Laird, and Rubin in 1977
- Sorta like, just assign points to clusters to start with and iterate.
- Then, at each iteration, replace the augmented data by its conditional expectation given current observed data and parameter estimates. (E-step)
- Maximize the full-data likelihood (M-step).

Why does it work?

$$p(x|\theta) = \sum_z p(x, z|\theta)$$

where the x and z range over the multiple points in your data set.

Then x-data log-likelihood $\ell(x|\theta) = log\, p(x|\theta) = log\, \sum_z p(x, z|\theta)$.

Hard to maximize for us.

Assume z has some normalized distribution:

$$z \sim q(z).$$

We wish to compute conditional expectations of the type:

$$E_{p(z|x,\theta)}\left[z\right]$$

but we dont know this "posterior" (henceforth p).

Lets say we somehow know q.

Consider KL loss function

$$\mathrm{KL}(q||p) = D_{KL}(q,p) = E_q[log\frac{q}{p}] = -E_q[log\frac{p}{q}]$$

$$D_{KL}(q,p) = -E_q[log\frac{p(x,z|\theta)}{q\,p(x|\theta)}]$$

$$D_{KL}(q,p) = -\left(E_q[log\frac{p(x,z|\theta)}{q}] - E_q[log\,p(x|\theta)]\right)$$

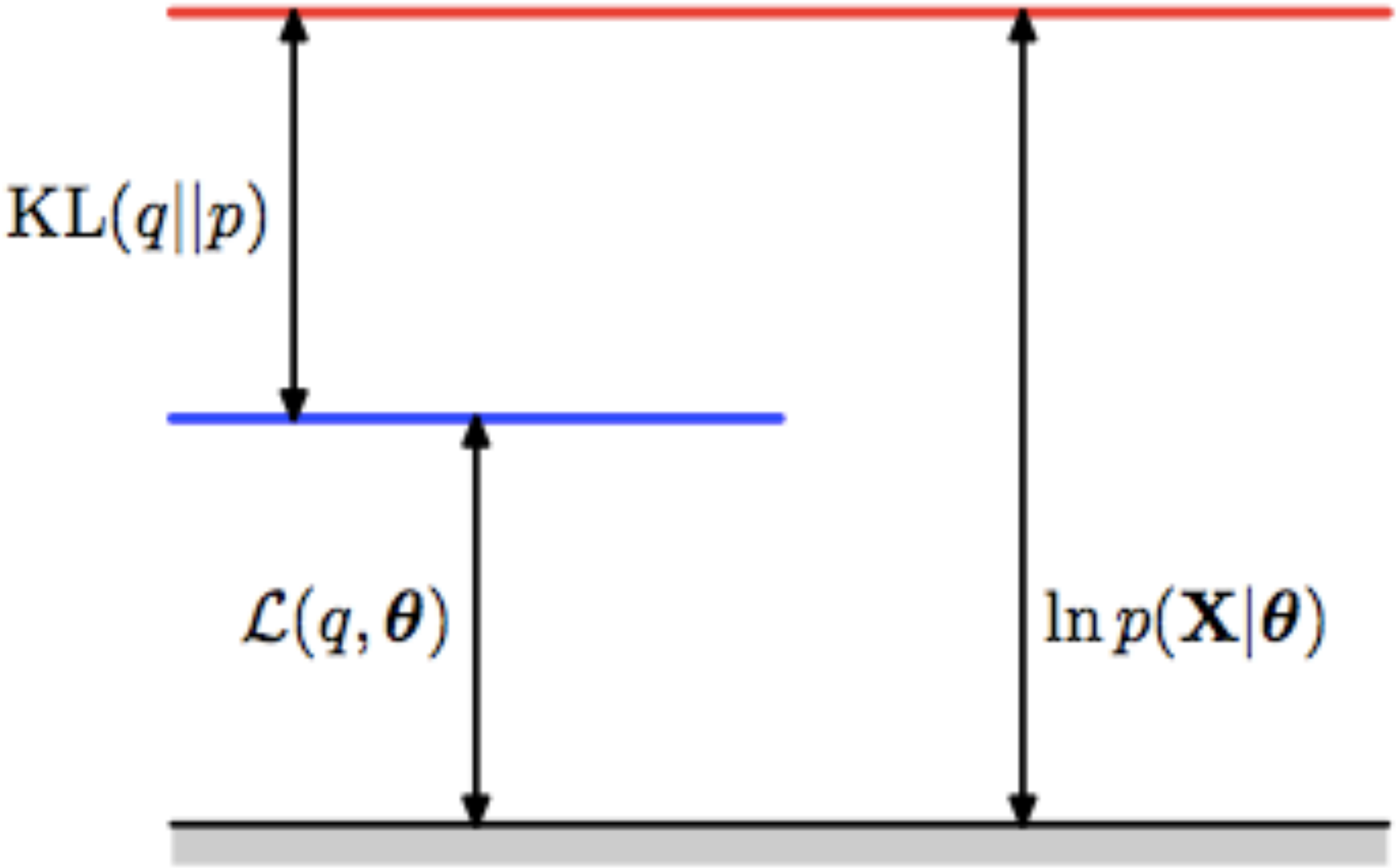
x-data likelihood

$$log\, p(x|\theta) = E_q[log \frac{p(x,z|\theta)}{q}] + D_{KL}(q,p)$$

If we define the ELBO or Evidence Lower bound as:

$$\mathcal{L}(q,\theta) = E_q[log \frac{p(x,z|\theta)}{q}]$$

then $log\, p(x|\theta)$ = ELBO + KL-divergence



$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

- KL divergence only 0 when $p = q$ exactly everywhere
- minimizing KL means maximizing ELBO
- ELBO $\mathcal{L}(q, \theta)$ is a lower bound on the log-likelihood.
- ELBO is average full-data likelihood minus entropy of q:

$$\mathcal{L}(q, \theta) = E_q[log \frac{p(x, z|\theta)}{q}] = E_q[logp(x, z|\theta)] - E_q[log\, q]$$

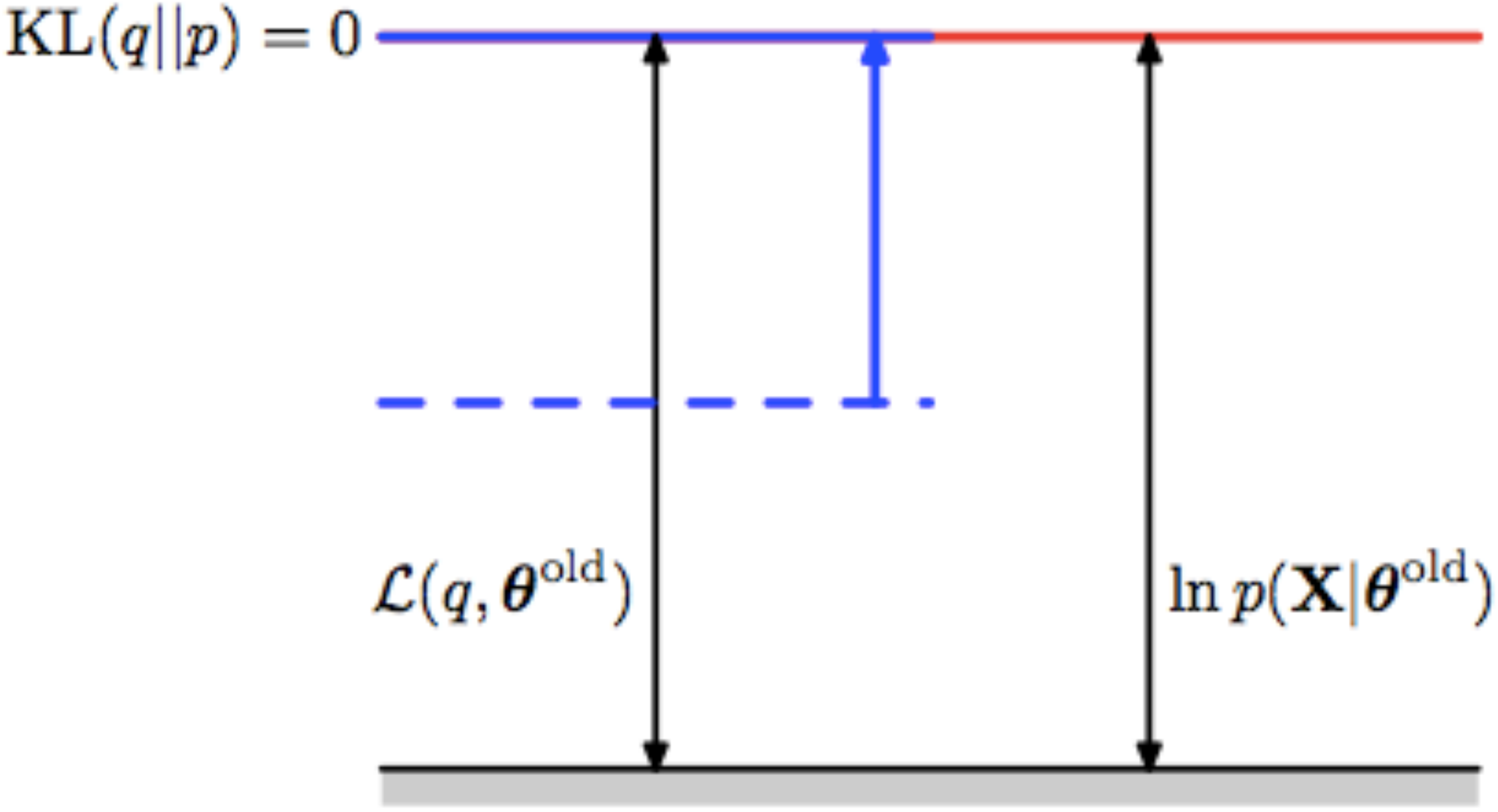
E-step conceptually

Choose at some (possibly initial) value of the parameters θ_{old},

$$q(z) = p(z|x, \theta_{old}),$$

then KL divergence = 0, and thus $\mathcal{L}(q, \theta)$ = log-likelihood at θ_{old}, maximizing the ELBO.

Conditioned on observed data, and θ_{old}, we use q to **conceptually** compute the expectation of the missing data.



$\mathrm{KL}(q||p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{old}})$

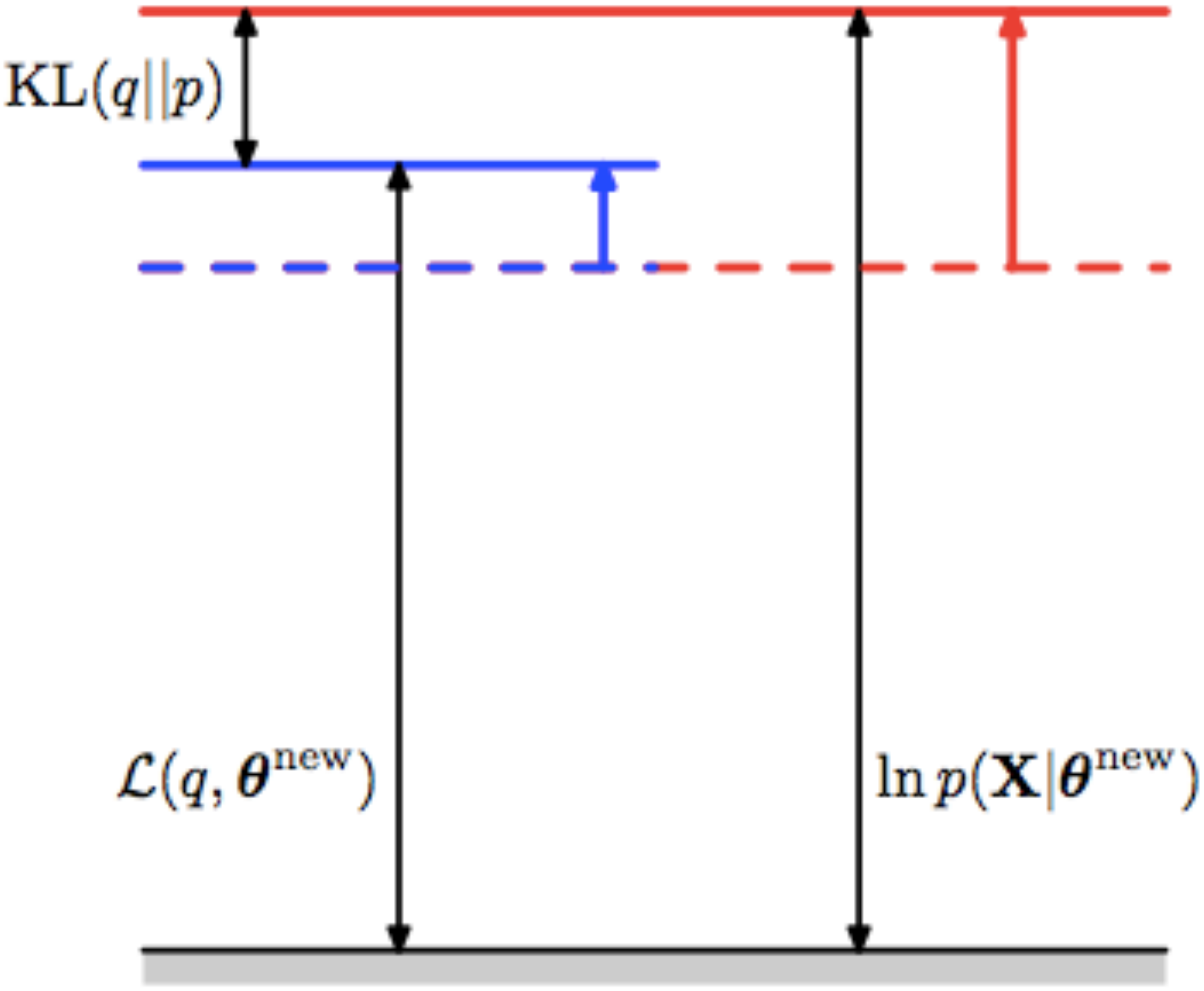
E-step: what we actually do

Compute the Auxilary function, $Q(\theta, \theta^{(t-1)})$, the expected (with respect to the z-posterior) complete(full) data log likelihood, defined by:

$$Q(\theta, \theta^{(t-1)}) = E_{Z|Y=y, \Theta=\theta^{t-1}}[logp(x, z|\theta)]$$

or the expectation of the ELBO instead of Q. Thus 2 parts:

(a) **Identify** $q = p(z \mid x, \theta_{old})$ (b) **compute** $Q = E_q[logp(x, z|\theta)]$.



$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{new}})$

M-step

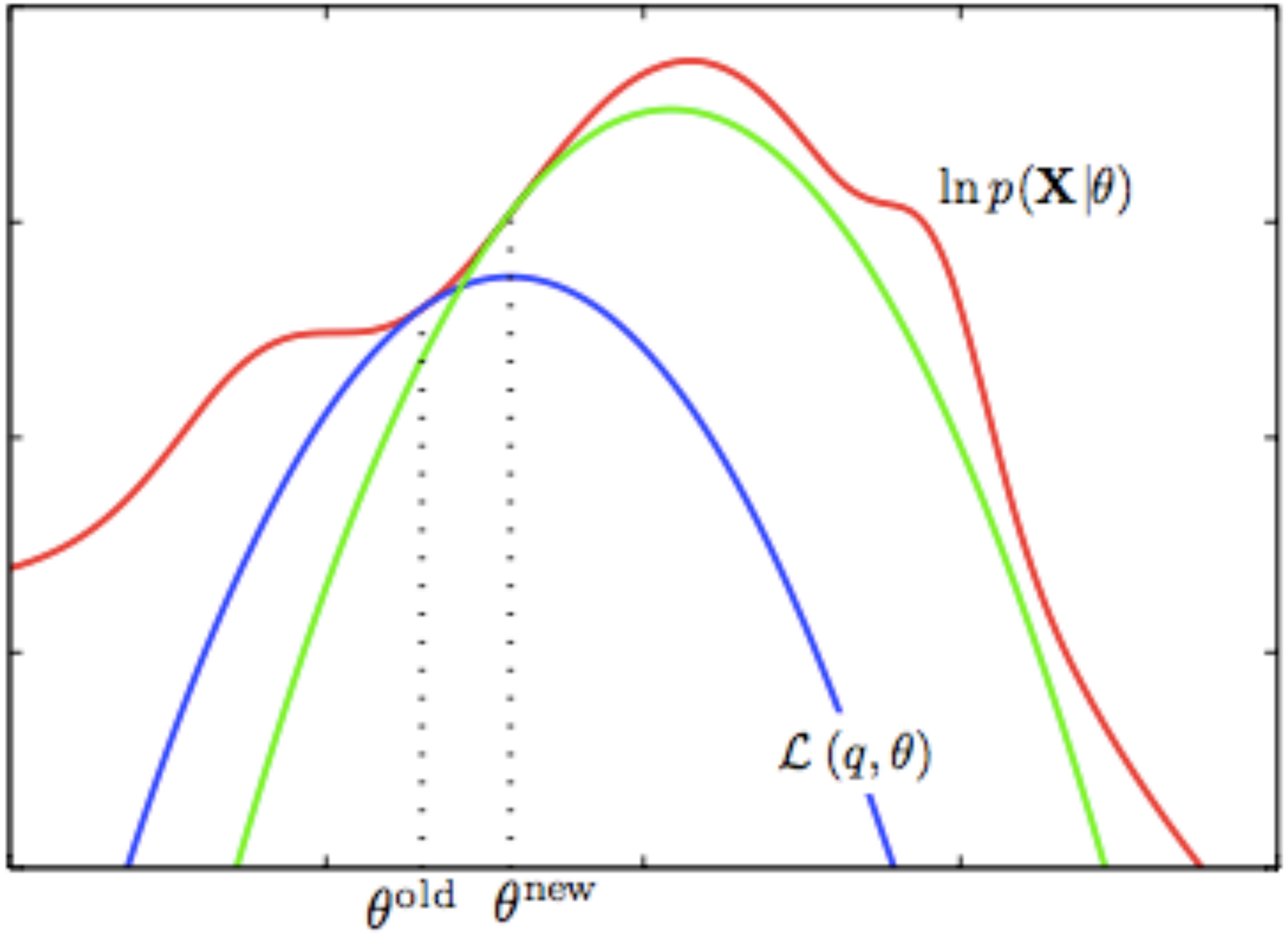
After E-step, ELBO touches $\ell(x|\theta)$, any maximization wrt θ will also “push up” on likelihood, thus increasing it.

Thus hold $q(z)$ fixed at the z-posterior calculated at θ_{old}, and maximize ELBO $\mathcal{L}(q, \theta, \theta_{old})$ or $Q(q, \theta, \theta_{old})$ wrt θ to obtain new θ_{new}.

In general $q(\theta_{old}0 \neq p(z|x, \theta_{new})$, hence KL $\neq 0$. Thus increase in $\ell(x|\theta) \geq$ increase in ELBO.

Process

1. Start with $p(x|\theta)$(red curve), θ_{old}.
2. Until convergence:
    1. E-step: Evaluate $q(z, \theta_{old}) = p(z|x, \theta_{old})$ which gives rise to $Q(\theta, \theta_{old})$ or $ELBO(\theta, \theta_{old})$(blue curve) whose value equals the value of $p(x|\theta)$ at θ_{old}.
    2. M-step: maximize Q or $ELBO$ wrt θ to get θ_{new}.
    3. Set $\theta_{old} = \theta_{new}$



An iteration:

$$\ell(\theta_{t+1}) \geq \mathcal{L}(q(z, \theta_t), \theta_{t+1}) \geq \mathcal{L}(q(z, \theta_t), \theta_t) = \ell(\theta_t)$$

The first equality follows since $\mathcal{L}$ is a lower bound on ℓ, the second from the M-step's maximization of $\mathcal{L}$, and the last from the vanishing of the KL-divergence after the E-step.

As a consequence, you **must** observe monotonic increase of the observed-data log likelihood ℓ across iterations. **This is a powerful debugging tool for your code**.

EM is local only!

Note that as shown above, since each EM iteration can only improve the likelihood, you are guaranteeing convergence to a local maximum. Because it IS local , you must try some different initial values of θ_{old} and take the one that gives you the largest ℓ.

GMM: E-step

E-step: Calculate $w_{i,j} = q_i(z_i = j) = p(z_i = j | x_i, \lambda_{old}, \mu_{old}, \Sigma_{old})$

Compute: $Q = \sum_i \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i | \lambda, \mu, \Sigma)}{q_i(z_i)}$

$$Q = \sum_i \sum_{j=i}^{k} q_i(z_i = j) \log \frac{p(x_i | z_i = j, \mu, \Sigma) p(z_i = j | \lambda)}{q_i(z_i = j)}$$

$$Q = \sum_{i=1}^{m} \sum_{j=i}^{k} w_{i,j} \log \left[\frac{\frac{1}{(2\pi)^{n/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2} (x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j) \right) \lambda_j}{w_{i,j}} \right]$$

E-step: calculate responsibilities

We are basically calculating the posterior of the z's given the x's and the current estimate of our parameters. We can use Bayes rule

$$w_{i,j} = p(z_i = j | x_i, \lambda_{old}, \mu_{old}, \Sigma_{old}) =$$

$$\frac{p(x_i | z_i = j, \mu_{old}, \Sigma_{old})\, p(z_i = j | \lambda_{old})}{\sum_{l=1}^{k} p(x_i | z_i = l, \mu_{old}, \Sigma_{old})\, p(z_i = l | \lambda_{old})}$$

M-step: mazimize Q

Taking derivatives yields following updating formulas:

$$\lambda_j = \frac{1}{m} \sum_{i=1}^{m} w_{i,j}$$

$$\mu_j = \frac{\sum_{i=1}^{m} w_{i,j}\, x_i}{\sum_{i=1}^{m} w_{i,j}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_{i,j}\, (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{m} w_{i,j}}$$

```python
def Estep(x, mu, sigma, lam):
    a = lam * norm.pdf(x, mu[0], sigma[0])
    b = (1. - lam) * norm.pdf(x, mu[1], sigma[1])
    return b / (a + b)

def Mstep(x, w):
    lam = np.mean(1.-w)

    mu = [np.sum((1-w) * x)/np.sum(1-w), np.sum(w * x)/np.sum(w)]

    sigma = [np.sqrt(np.sum((1-w) * (x - mu[0])**2)/np.sum(1-w)),
             np.sqrt(np.sum(w * (x - mu[1])**2)/np.sum(w))]

    return mu, sigma, lam
```

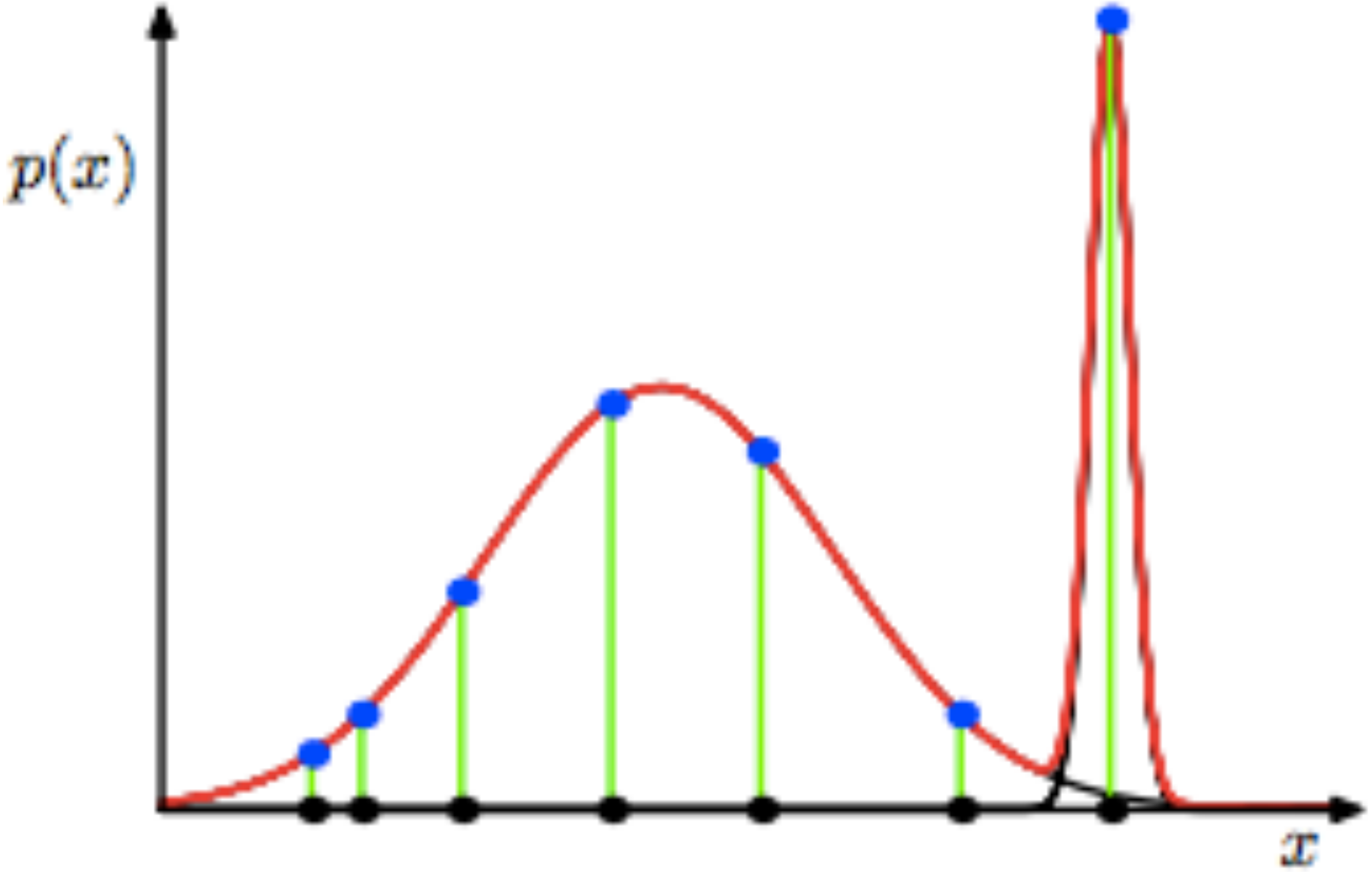
```
0.4 [2, 5] [0.6, 0.6]
Initials, mu: [-4.85176052  5.51133343]
Initials, sigma: [ 2.02807915  3.58912888]
Initials, lam: 0.5418931691319009
Iterations 71
A: N(2.0261, 0.5936)
B: N(5.0083, 0.6288)
lam: 0.5884

0.4 [2, 5] [0.6, 0.6]
Initials, mu: [ 11.09643621  -4.48315085]
Initials, sigma: [ 4.31750531  0.95518757]
Initials, lam: 0.5767814041950222
Iterations 103
A: N(5.0083, 0.6288)
B: N(2.0261, 0.5936)
lam: 0.4116
```

Compared to supervised classification and k-means

- M-step formulas vs GDA we can see that are very similar except that instead of using δ functions we use the w's.
- Thus the EM algorithm corresponds here to a weighted maximum likelihood and the weights are interpreted as the 'probability' of coming from that Gaussian
- Thus we have achieved a **soft clustering** (as opposed to k-means in the unsupervised case and classification in the supervised case).

- kmeans is HARD EM. Instead of calculating Q in e-step, use mode of z posterior. Also the case with classification
- finite mixture models suffer from multimodality, non-identifiability, and singularity. They are problematic but useful
- models can be singular if cluster has only one data point: overfitting
- add in prior to regularise and get MAP. Add log(prior) in M-step only



Exchangeability

Think of our poisson based college/no-college problem.

Lets assume that the number of children of a women in any one of these classes can me modelled as coming from ONE birth rate.

The in-class likelihood for these women is invariant to a permutation of variables.

This is really a statement about what is **exchangeable** and what is not.

It depends on how much knowledge you have...

Back to Joint Densities

- even if we never calculate it, we must consider the joint density $p(x_1, \ldots, x_n, \theta, \phi, etc)$
- Lets just focus for a bit on $p(x_1, \ldots, x_n)$. This must capture the type of dependence assumed among the x_i.
- one assumption could be that these data are IID. BUT more generally consider that the labels or subscripts are uninformative
- that is, $p(x_1, \ldots, x_n) = p(x_{\pi(1)}, \ldots, x_{\pi(n)})$, for all permutations π.
- A sequence of random quantities is said to be **exchangeable** if this property holds for every finite subset of them (Bernardo)

De-Finetti's Representation Theorem

For exchangeable $\{x_i\}$, there exists a parametric model, $p(x|\theta)$, labeled by some parameter $\theta \in \Theta$:

- which is the $n \to \infty$ limit of some function F of the x_i's, and
- There exists a probability distribution for θ, with density $p(\theta)$, such that we get an infinite mixture:

$$p(x_1, \ldots, x_n) = \int_\Theta \prod_{i=1}^{n} p(x_i \mid \theta) p(\theta) d(\theta)$$

That is, quoting Bernardo,

> *if a sequence of observations is judged to be exchangeable, then, any finite subset of them is a random sample of some* ***model*** $p(x_i \mid \theta)$*, and there exists a* ***prior*** *distribution* $p(\theta)$ *which has to describe the initially available information about the parameter which labels the model.*

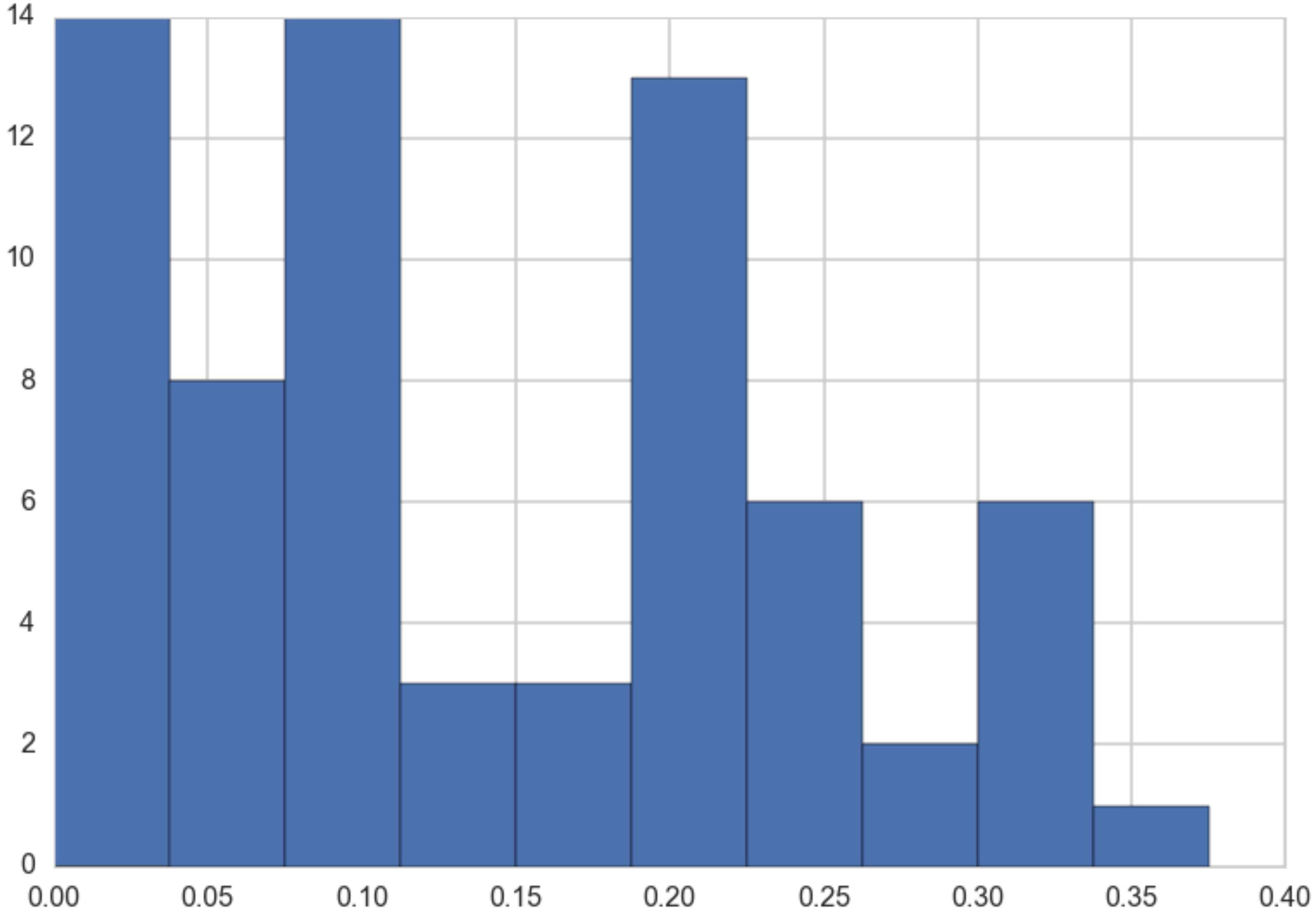
That is, exchageability demands a likelihood with conditionally independent observations, and these observations:

> *must indeed be a random sample from some model and there must exist a prior probability distribution over the parameter of the model, hence* ***REQUIRING*** *a Bayesian approach*

De Finetti's theorem helps dispel the mystery of where the prior belief over the chances comes from. From exchangeable degrees of belief, de Finetti recovers both the chance statistical model of coin flipping and the Bayesian prior probability over the chances. The mathematics of inductive inference is just the same. If you were worried about where Bayes' priors came from, if you were worried about whether chances exist, you can forget your worries. De Finetti has replaced them with a symmetry condition on degrees of belief. This is, we think you will agree, a philosophically sensational result.

From Diaconis, Persi; Skyrms, Brian. Ten Great Ideas about Chance (Page 124). Princeton University Press.

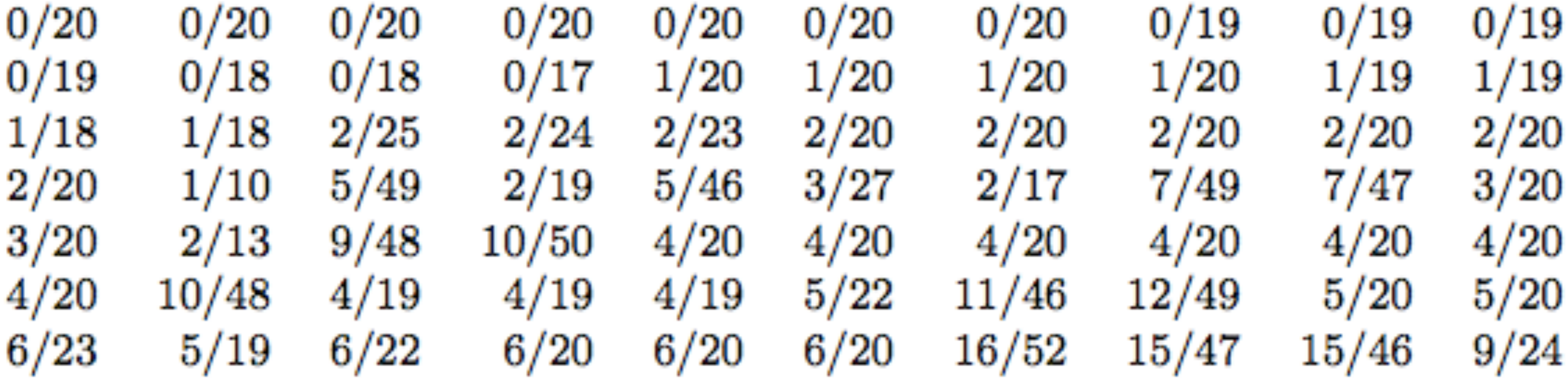
HIERARCHICAL MODELS



Rat Tumors

- tumors in female rats of type "F344" that recieve a particular drug, in 70 different experiments.
- mean and variance of tumor incidence: `0.13600653889043893, 0.010557640623609196`
- 71st experiment done: 4 out of 14 rats develop tumors. Estimate the risk of tumor in the rats in the 71st experiment

Tumors data



0/20	0/20	0/20	0/20	0/20	0/20	0/20	0/19	0/19	0/19
0/19	0/18	0/18	0/17	1/20	1/20	1/20	1/20	1/19	1/19
1/18	1/18	2/25	2/24	2/23	2/20	2/20	2/20	2/20	2/20
2/20	1/10	5/49	2/19	5/46	3/27	2/17	7/49	7/47	3/20
3/20	2/13	9/48	10/50	4/20	4/20	4/20	4/20	4/20	4/20
4/20	10/48	4/19	4/19	4/19	5/22	11/46	12/49	5/20	5/20
6/23	5/19	6/22	6/20	6/20	6/20	16/52	15/47	15/46	9/24

Modeling

$$p(y_i|\theta_i; n_i) = Binom(n_i, y_i, \theta_i)$$

$$p(Y|\Theta; \{n_i\}) = \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Need to choose a prior $p(\Theta)$.

No Pooling

Separate priors on each θ_i:

$$\theta_i \sim Beta(\alpha_i, \beta_i).$$

$$p(\Theta | \{\alpha_i\}, \{\beta_i\}) = \prod_{i=1}^{70} Beta(\theta_i, \alpha_i, \beta_i),$$

Very overfit model with 210 parameters. VARIANCE!

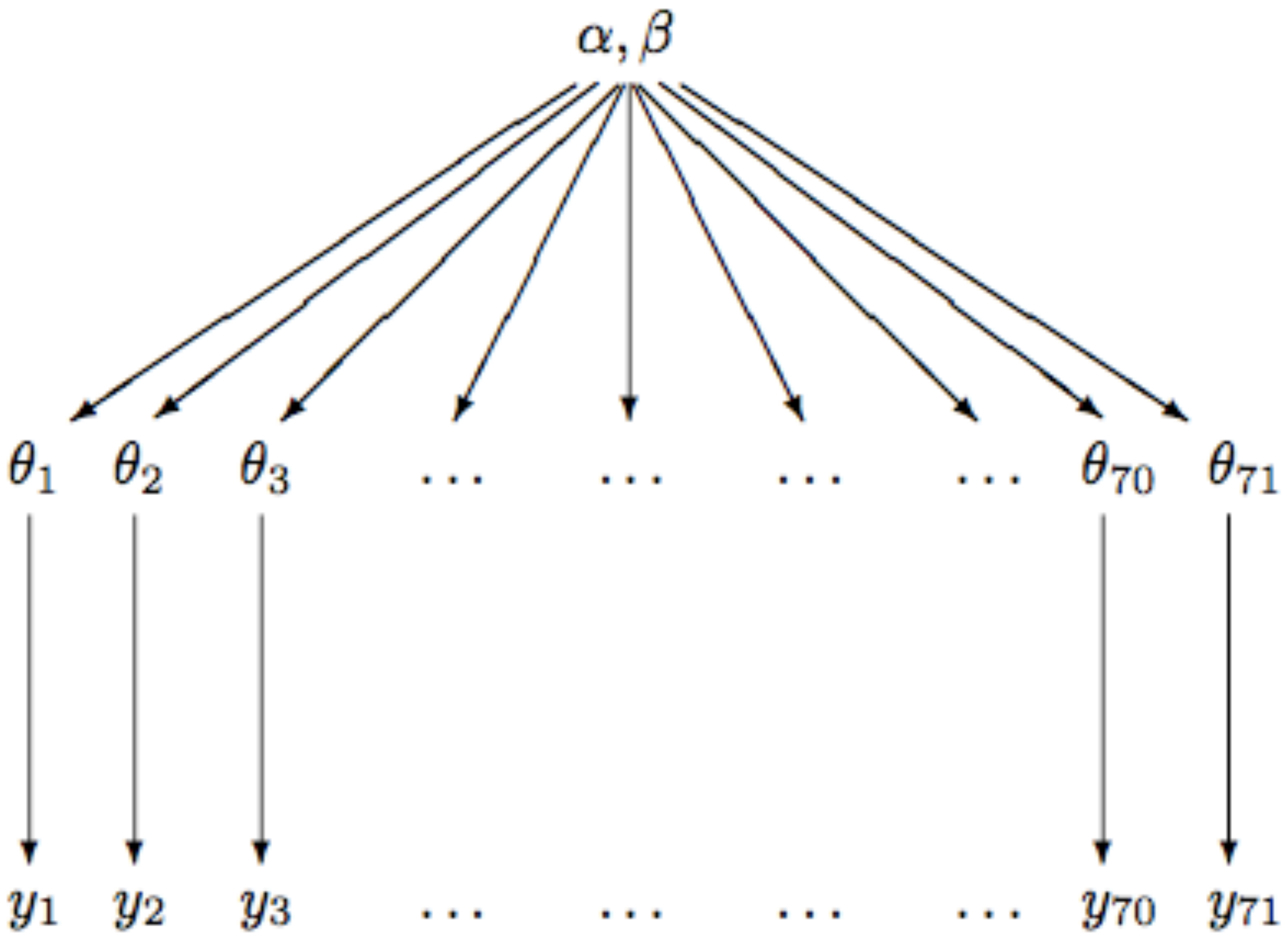
Full Pooling

Assume that there is only one θ in the problem, and set an prior on it.

Ignores any variation amongst the sampling units other than sampling variance.

Underfit model with 3 params. BIAS

Partial pooling: Hierarchical Model


α, β
θ_1 θ_2 θ_3 $\dots$ $\dots$ $\dots$ $\dots$ θ_{70} θ_{71}
y_1 y_2 y_3 $\dots$ $\dots$ $\dots$ $\dots$ y_{70} y_{71}

θ_is drawn from "population distribution" given by a conjugate Beta prior $Beta(\alpha, \beta)$ with **hyperparameters** α and β.

$$\theta_i \sim Beta(\alpha, \beta).$$

$$p(\Theta | \alpha, \beta) = \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta).$$

Why is this ok?

Suppose we have several sequences of data $\mathbf{x}_i$, each assumed to dependent separately on sufficient statistics $\mathbf{t}_i$, ie:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_i^m p(\mathbf{x}_i \mid \mathbf{t}_i)$$

Then the representation theorem looks like:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \int_\Theta \prod_{i=1}^m \prod_{j=1}^{n_i} p(\mathbf{x}_{ij} \mid \theta_i) p(\theta_i \ldots \theta_m) d\theta_i \ldots d\theta_m$$

If one has $p_i(x \mid \theta_i) = p(x \mid \theta_i)$, then one can recursively use De-Finetti's theorem for each θ_i:

$$p(\theta_1, \ldots, \theta_n) = \int_\Phi \prod_{i=1}^{n} p(\theta_i \mid \phi) p(\phi) d(\phi)$$

Bernardo:

hence, the parameter values which correspond to each sequence may be seen as a random sample from some parameter population with density $p(\theta \mid \phi)$, and there must exist a prior distribution $p(\phi)$ describing the initial information about the hyperparameter ϕ which labels $p(\theta \mid \phi)$.

- De-Finetti tells us that in the limit of infinite data points exchangeability at the θ level is captured as an infinite mixture of IID distributions

- First proved for infinite data points but subsequently proved approximately for finite number of points

- often observations are only partially or conditionally exchangeable. This helps when we want to model groups.

- if y_i has accompanying co-variates x_i, so that y_i are not exchangeable, but the pair are, then we make a joint model for the pair (x_i, y_i) or a conditional model for $y_i \mid x_i$.

- We write then: $p(\theta_1, .., \theta_n \mid x_i, .., x_n) = \int [\prod_j p(\theta_j \mid \phi, x_j)] p(\phi \mid x) d\phi$

Priors from data

Where do α and β come from?

Why are we calling them hyperparameters?

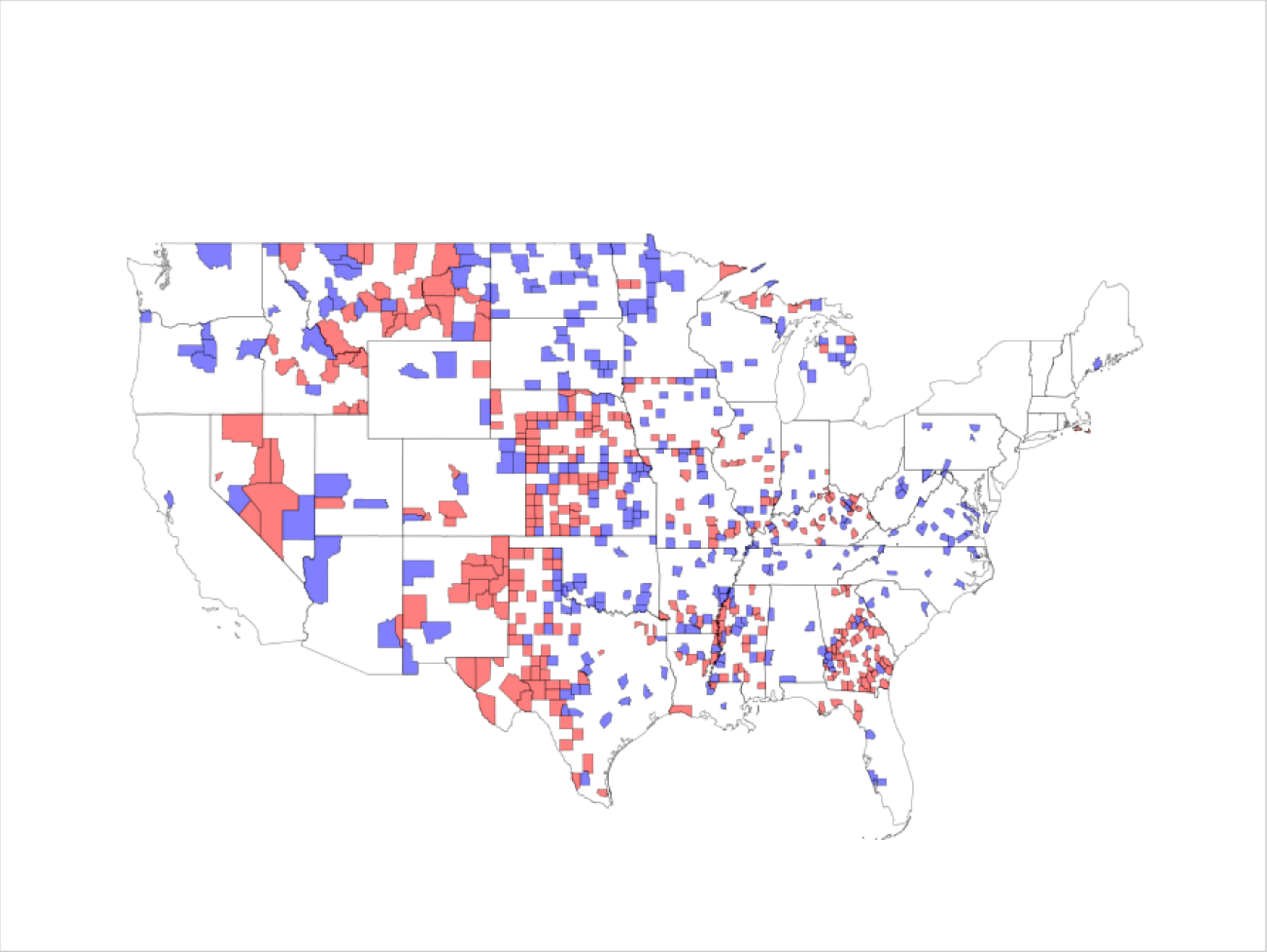
So far have assumed α and β known in priors to be weakly informative.

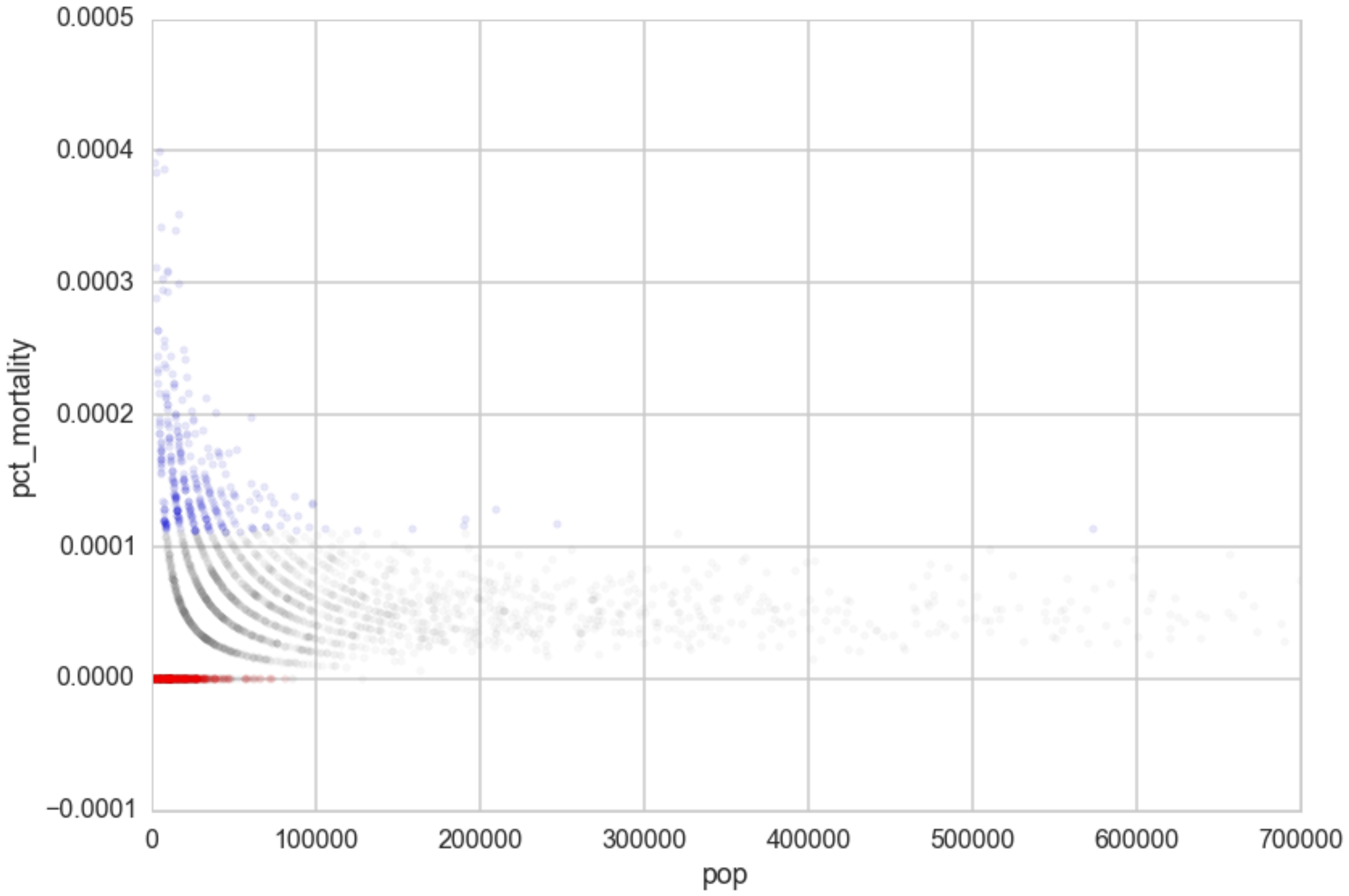
New idea: estimate priors from data. Looks like a cross-validation like setup.

Key Idea: Share statistical strength

- Some **units** (experiments) statistically more robust
- Non-robust experiments have smaller samples or outlier like behavior
- Borrow strength from all the data as a whole through the estimation of the hyperparameters
- **regularized partial pooling model** in which the "lower" parameters (θs) tied together by "upper level" hyperparameters.

Another Example: Kidney cancers





First idea: estimate directly from data

Posterior-predictive distribution, as a function of upper level parameters $\eta = (\alpha, \beta)$.

$$p(y^* \vert D, \eta) = \int d\theta \, p(y^* \vert \theta) \, p(\theta \vert D, \eta)$$

A likelihood with parameters η and simply use maximum-likelihood with respect to η to estimate these η using our "data" y^*

Called Empirical Bayes or Type-2 MLE

- MLE with respect to η
- involves an optimization
- unlike cross-validation, θs not-yet estimated on training set.
- indeed we marginalize over θs so can use training set.
- in practice often match moments of predictive or posterior

EB for rats: prior/prior predictive...

Consider the prior expectation and variance:

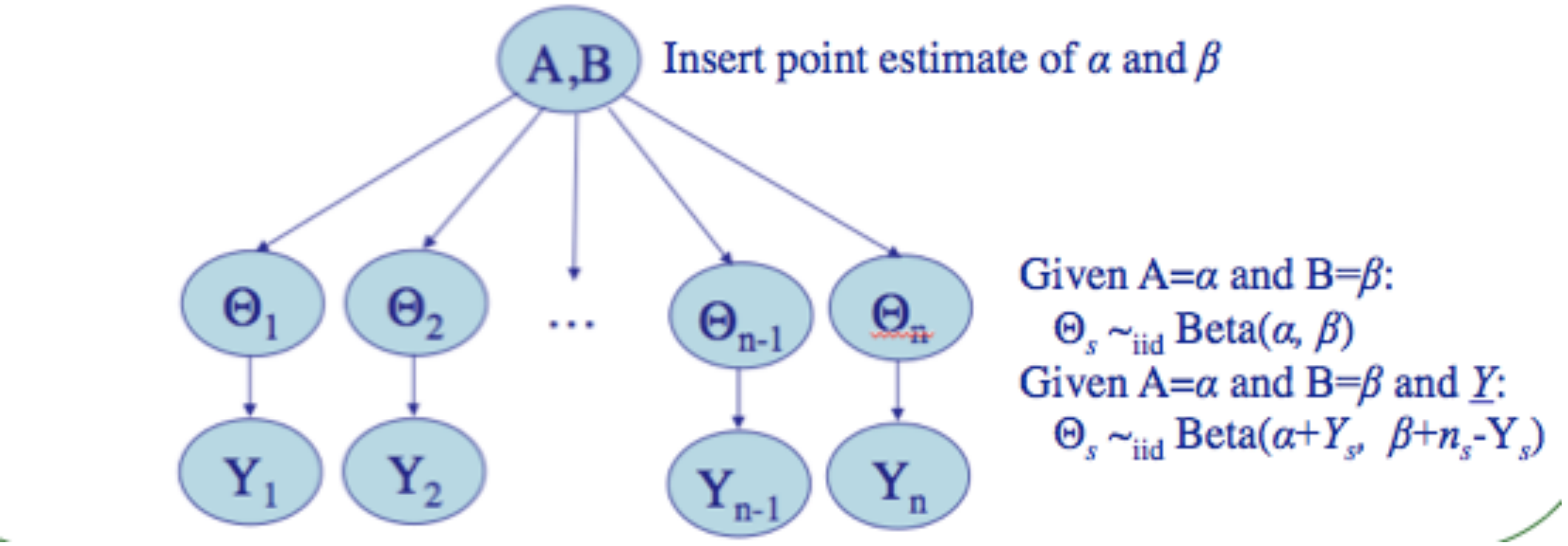
$$\mu = \frac{\alpha}{\alpha + \beta}, V = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Match empirical mean and variance on y_i/n_i

- Need to be careful what "space" you are working in, predictive (y) or not
- Use prior predictive if in a "predictive space":

  $$p(y^*) = E_{p(\theta}[p(y^*|\theta)] = \int d\theta p(y^*|\theta)p(\theta).$$

…to posterior/posterior predictive…

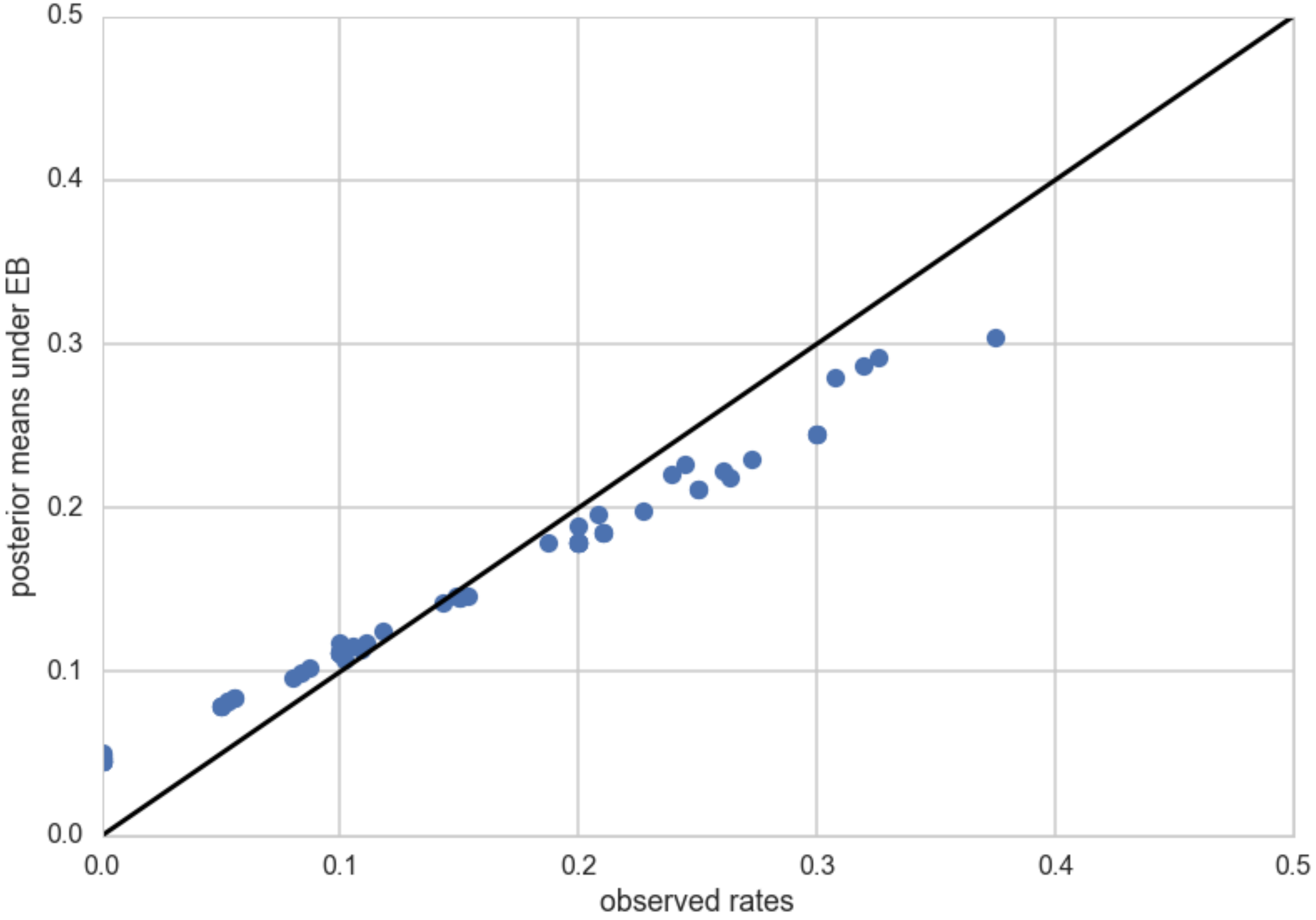


- (α, β) = `(1.3777748392916778, 8.7524354471531129)`
- Conditional posterior distribution for each of the θ_i, given everything else is Beta:.

$$p(\theta_i | y_i, n_i, \alpha, \beta) = Beta(\alpha + y_i, \beta + n_i - y_i)$$

$$\bar{\theta}_{post,i} = \frac{\alpha + y_i}{\alpha + \beta + n_i}$$

Shrinkage in rat (tumors)



Posterior estimates shrink towards full pooling.

Now, for the 71st experiment, we have 4 out of 14 rats having tumors. The posterior estimate for this would be

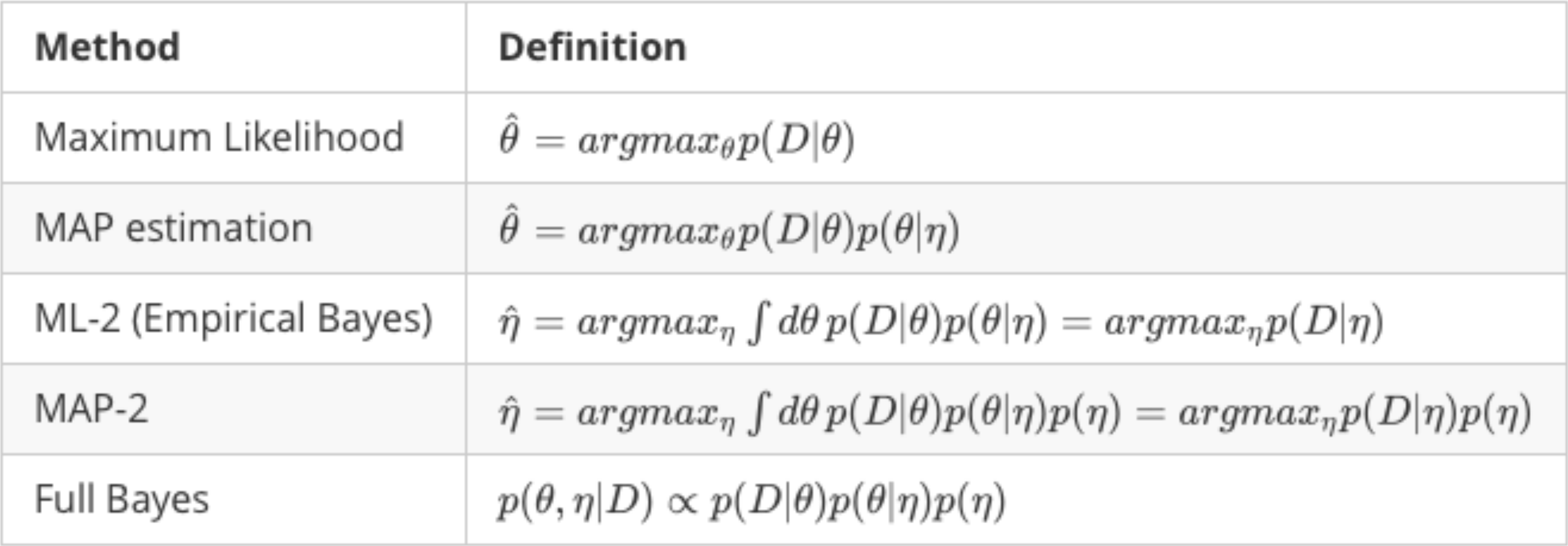
$$\frac{\alpha + y_{71}}{\alpha + \beta + n_{71}}$$

```
4/14, (4+a_est)/(14+a_est+b_est)
= (0.2857142857142857, 0.22286481449822493)
```

Hierarchy organizes exchangeability

- we use the notion of exchangeability at the level of 'units'.
- for our rats, the y_j were exchangeable since we had no additional information about experimental conditions.
- if specific groups of experiments came from specific laboratories, assume experiments interchangeable if from the same lab.
- lab specific α_{lab} and β_{lab} parameters
- add another level of hierarchy to draw these from hyperprior.

Levels of Bayes



Method	Definition
Maximum Likelihood	$\hat{\theta} = argmax_{\theta} p(D\|\theta)$
MAP estimation	$\hat{\theta} = argmax_{\theta} p(D\|\theta) p(\theta\|\eta)$
ML-2 (Empirical Bayes)	$\hat{\eta} = argmax_{\eta} \int d\theta \, p(D\|\theta) p(\theta\|\eta) = argmax_{\eta} p(D\|\eta)$
MAP-2	$\hat{\eta} = argmax_{\eta} \int d\theta \, p(D\|\theta) p(\theta\|\eta) p(\eta) = argmax_{\eta} p(D\|\eta) p(\eta)$
Full Bayes	$p(\theta, \eta\|D) \propto p(D\|\theta) p(\theta\|\eta) p(\eta)$