iPMR

Learning and Inference

- Sometimes Bayesian learning and inference is intractable.
- I.e. computing the right posterior distribution or inference distribution cannot be done efficiently.
- Still need to do learning and inference!
- Approximate methods.

Divergences

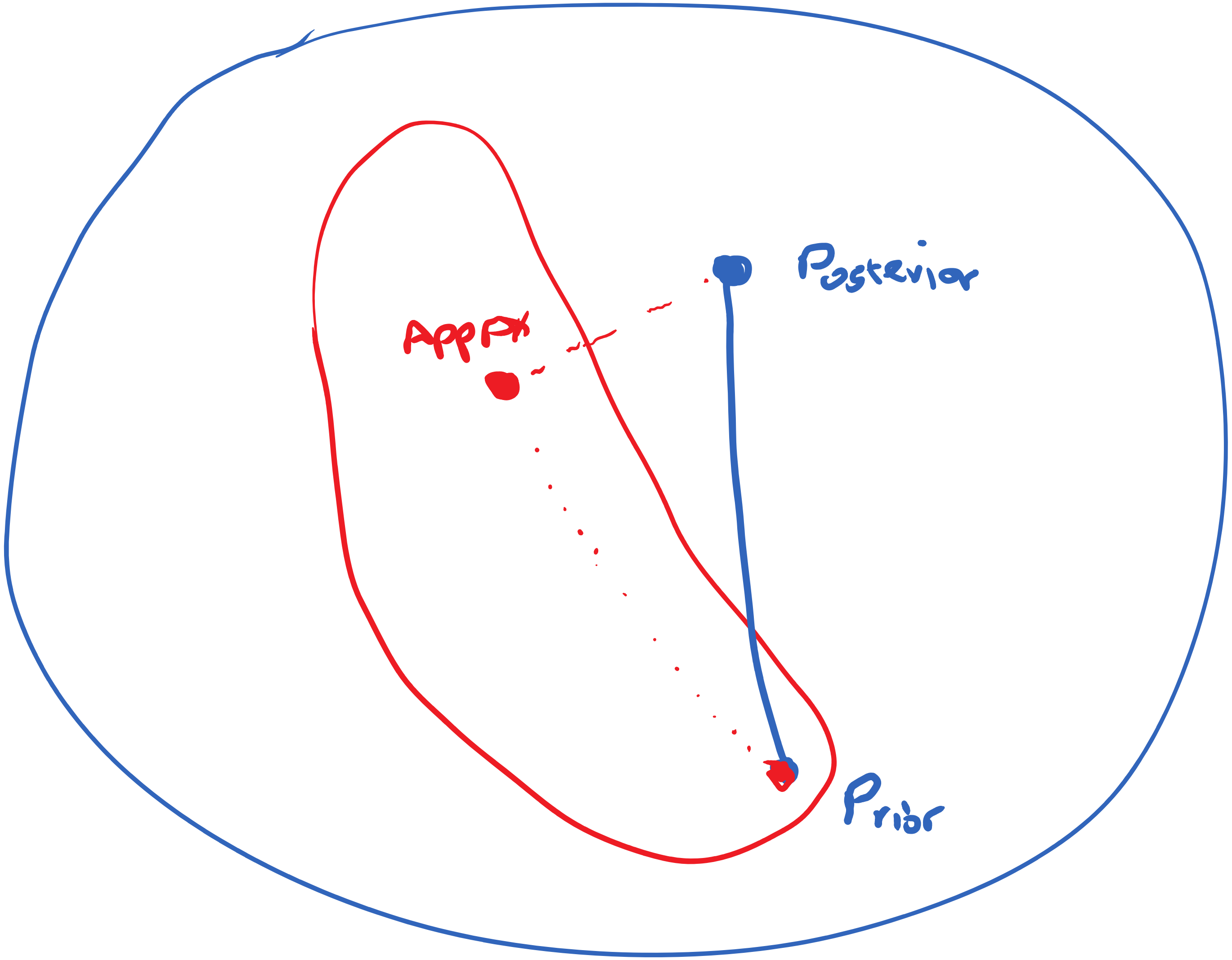
- Divergences measure a cost of using a wrong distribution instead of a correct one.
- E.g. KL(Q||P) – measures the coding cost of coding using the distribution P instead of the true distribution Q.

$$KL(Q(.)||P(.)) = \int d\mathbf{x}\; Q(\mathbf{x}) \log \frac{Q(\mathbf{x})}{P(\mathbf{x})}$$

- Sometimes by constraining the set of distributions we allow, and the ensuring the divergence to the required distribution is low, we can do the computation.

Note: placement of dx is for convenience, all logs are natural logs throughout

Picture



Inference to Optimization

- Simple case: consider using a delta function instead of the true distribution.
- E.g. In Learning:
  - Instead of computing a posterior distribution P, compute the best **delta function** approximation Q.
  - Minimizing KL(Q||P) means picking the location for the delta function that maximizes the posterior.
- We have turned learning from an inference to an optimizing function: maximize posterior.

Independent Component Analysis

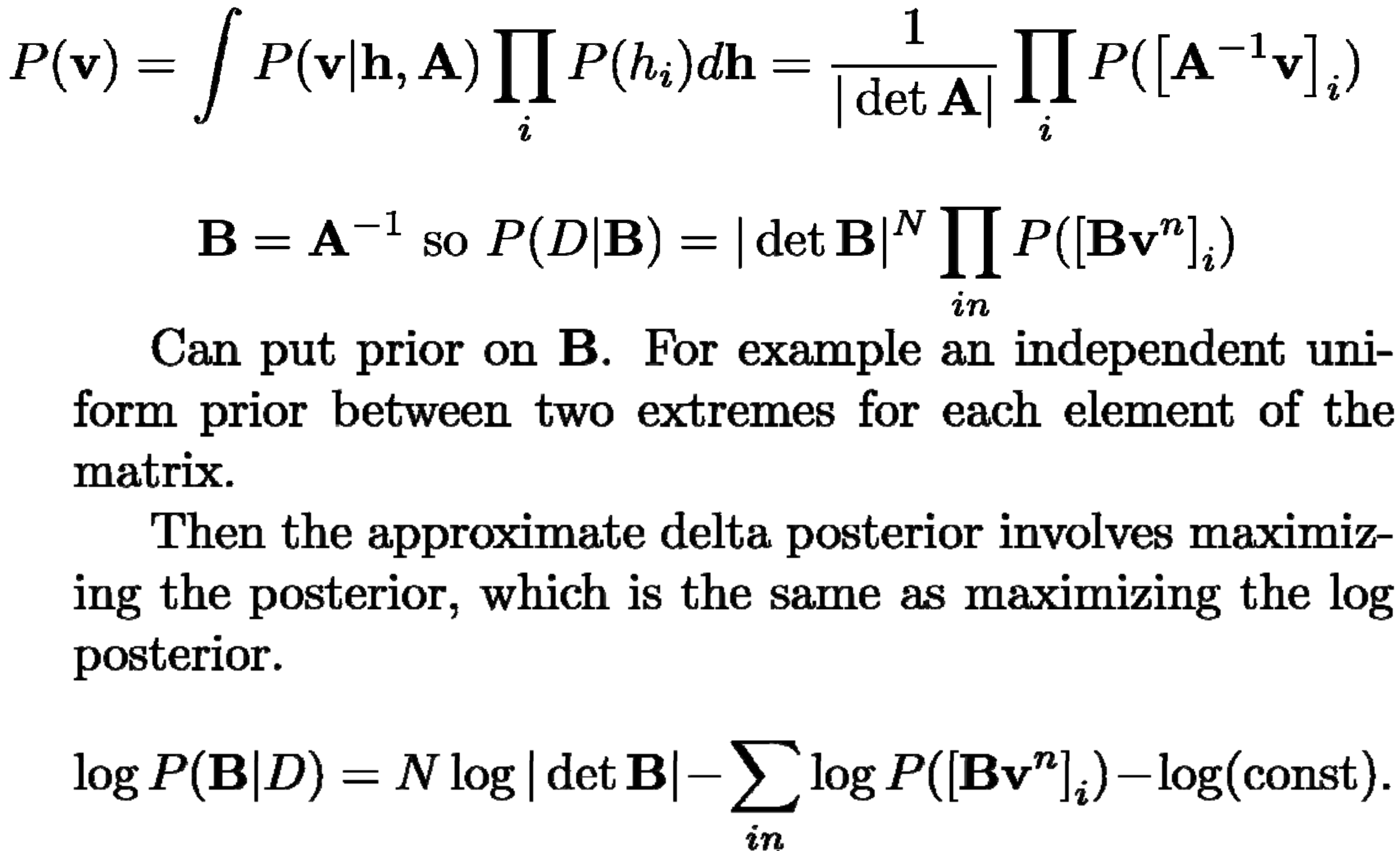
- By way of example
- Consider the problem of discovering independent sources in data.
- m independent sources (e.g. speakers)
- m different points of observation (e.g. microphones)
- Each microphone “hears” all the sources.
- But in different proportions depending on location.
- Want unmixed sources.

ICA

- Model

$$P(\mathbf{v},\mathbf{h}|\mathbf{A}) = P(\mathbf{v}|\mathbf{h},\mathbf{A})\prod_i P(h_i)$$

$$P(\mathbf{v}|\mathbf{h},\mathbf{A}) = \delta(\mathbf{v},\mathbf{A}\mathbf{h})$$



$$P(\mathbf{v}) = \int P(\mathbf{v}|\mathbf{h},\mathbf{A})\prod_i P(h_i)d\mathbf{h} = \frac{1}{|\det\mathbf{A}|}\prod_i P([\mathbf{A}^{-1}\mathbf{v}]_i)$$

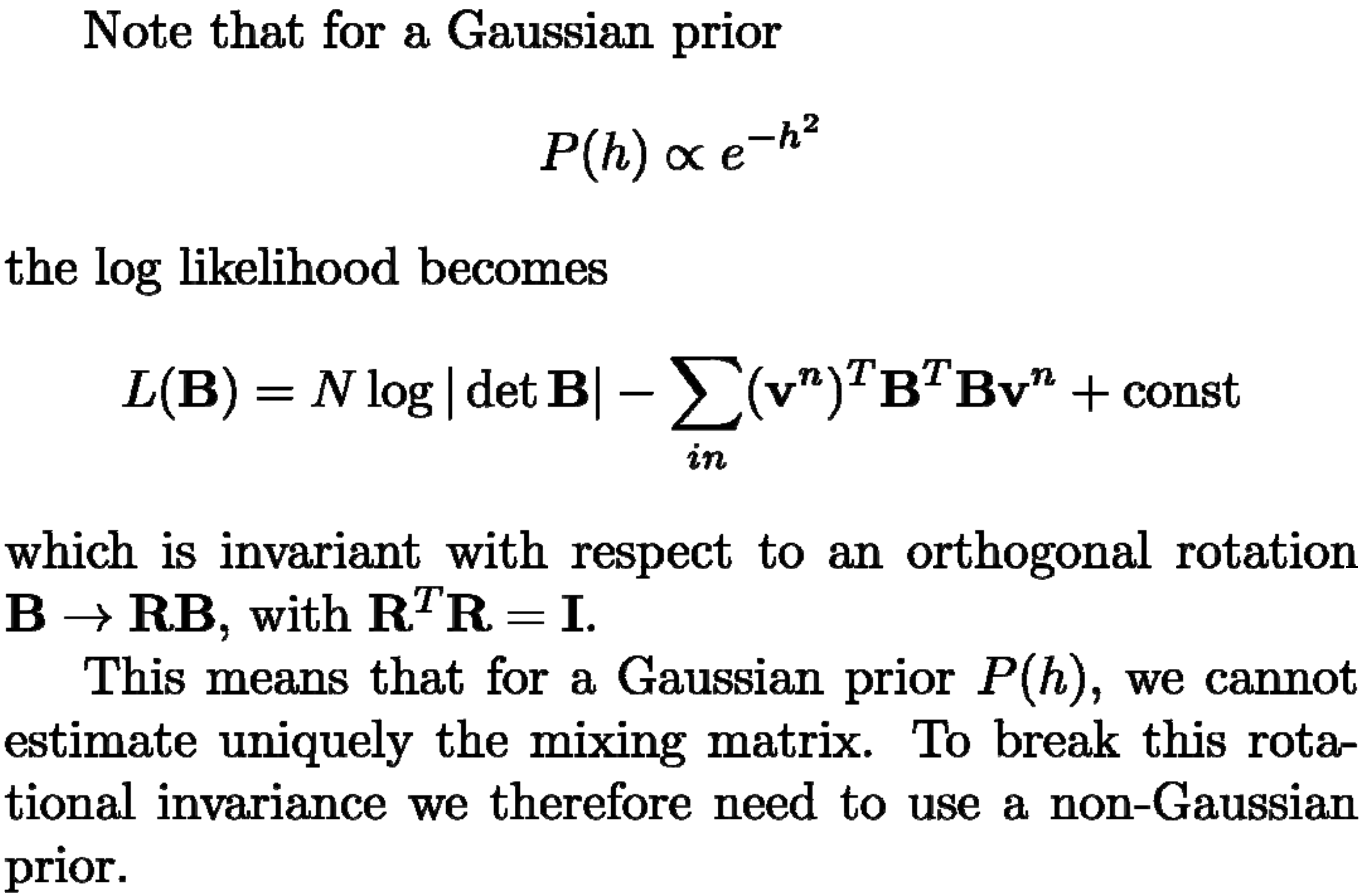
$$\mathbf{B} = \mathbf{A}^{-1} \text{ so } P(D|\mathbf{B}) = |\det\mathbf{B}|^N \prod_{in} P([\mathbf{B}\mathbf{v}^n]_i)$$

Can put prior on $\mathbf{B}$. For example an independent uniform prior between two extremes for each element of the matrix.

Then the approximate delta posterior involves maximizing the posterior, which is the same as maximizing the log posterior.

$$\log P(\mathbf{B}|D) = N\log|\det\mathbf{B}| - \sum_{in}\log P([\mathbf{B}\mathbf{v}^n]_i) - \log(\text{const}).$$

Non Gaussianity



Note that for a Gaussian prior

$$P(h) \propto e^{-h^2}$$

the log likelihood becomes

$$L(\mathbf{B}) = N \log|\det \mathbf{B}| - \sum_{in} (\mathbf{v}^n)^T \mathbf{B}^T \mathbf{B} \mathbf{v}^n + \text{const}$$

which is invariant with respect to an orthogonal rotation $\mathbf{B} \to \mathbf{RB}$, with $\mathbf{R}^T\mathbf{R} = \mathbf{I}$.

This means that for a Gaussian prior $P(h)$, we cannot estimate uniquely the mixing matrix. To break this rotational invariance we therefore need to use a non-Gaussian prior.

Optimizing

Assuming we have a non-Gaussian prior $p(h)$, taking the derivative *w.r.t.* B_{ab} we obtain

$$\frac{\partial}{\partial B_{ab}} L(\mathbf{B}) = N A_{ba} + \sum_n \phi([\mathbf{B}\mathbf{v}]_a) v_b^n$$

where

$$\phi(x) \equiv \frac{d}{dx} \log p(x) = \frac{1}{p(x)} \frac{d}{dx} p(x)$$

A simple gradient ascent learning rule for $\mathbf{B}$ is then

$$\mathbf{B}^{new} = \mathbf{B} + \eta \left(\mathbf{B}^{-\mathsf{T}} + \frac{1}{N} \sum_n \phi(\mathbf{B}\mathbf{v}^n) (\mathbf{v}^n)^{\mathsf{T}} \right)$$

Optimizing

An alternative ‘natural gradient’ algorithm that approximates a Newton update is given by multiplying the gradient by $\mathbf{B}^{\mathsf{T}}\mathbf{B}$ on the right to give the update

$$\mathbf{B}^{new} = \mathbf{B} + \eta \left(\mathbf{I} + \frac{1}{N} \sum_{n} \phi(\mathbf{B}\mathbf{v}^{n}) (\mathbf{B}\mathbf{v}^{n})^{\mathsf{T}} \right) \mathbf{B}$$

Here η is an empirically set learning rate.

fast ICA

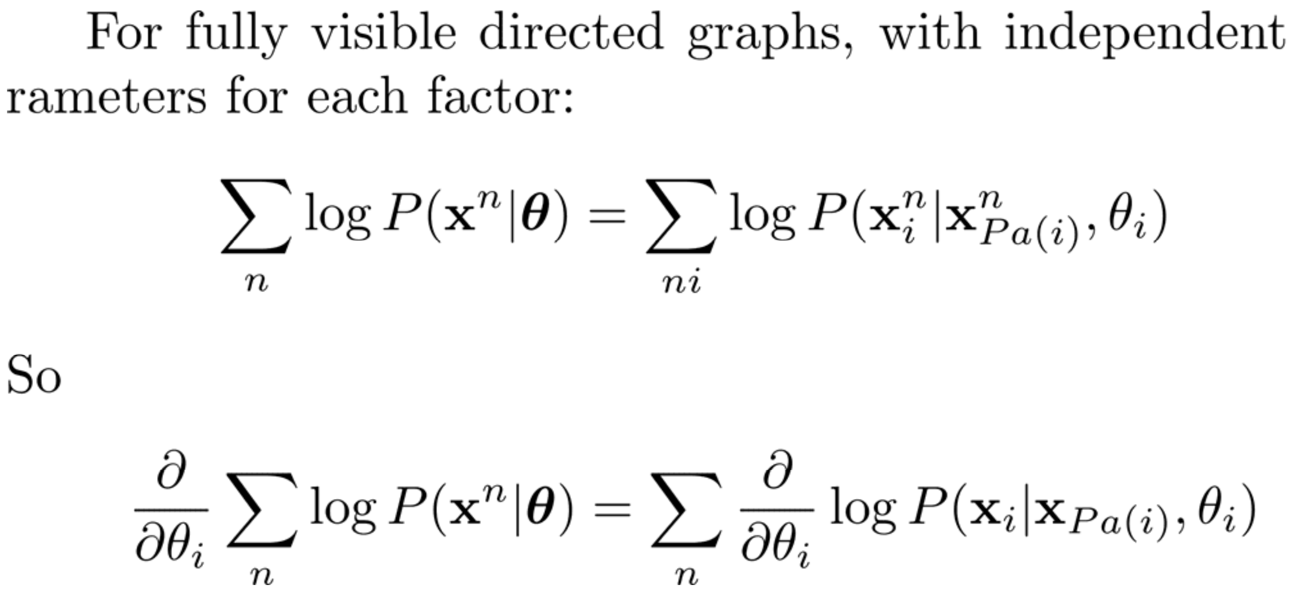
A popular alternative estimation method is FastICA and can be related to an iterative Maximum Likelihood optimisation procedure.

Summary

- Computing the maximum posterior can be thought of as approximating the posterior with the best choice of delta function.
- With a uniform prior maximum posterior is the same as maximum likelihood.
- Converts to an optimization problem.
- Note: Bayesian Inference is representation independent. Optimization is representation dependent.

ML and Graphical Models

- Maximum likelihood works fine for graphical models where all nodes are visible.



For fully visible directed graphs, with independent parameters for each factor:

$$\sum_n \log P(\mathbf{x}^n|\boldsymbol{\theta}) = \sum_{ni} \log P(\mathbf{x}_i^n|\mathbf{x}_{Pa(i)}^n, \theta_i)$$

So

$$\frac{\partial}{\partial\theta_i}\sum_n \log P(\mathbf{x}^n|\boldsymbol{\theta}) = \sum_n \frac{\partial}{\partial\theta_i} \log P(\mathbf{x}_i|\mathbf{x}_{Pa(i)}, \theta_i)$$

ML and Graphical Models

- Maximum likelihood not always straightforward in general graphs or graphs with hidden nodes.
- Need to be able to compute with both prior and posterior.

Let $\mathbf{x}^n = ((\mathbf{v}^n)^T, (\mathbf{h}^n)^T)^T$. Then we want

$$\sum_n \log \sum_{\mathbf{h}^n} P(\mathbf{x}^n|\boldsymbol{\theta}) \text{ (summing out over hidden part of } \mathbf{x}\text{)}$$

Useful result which holds for any split of variables into sets S and R. Note we split clique C_j into S_j, the part in S, and R_j (i.e. rest), the part not in S.

$$\frac{\partial}{\partial\theta_i}\log\sum_{\mathbf{x}_S}\exp\left(\sum_j \phi_j(\mathbf{x}_{C_j},\theta_j)\right) = \frac{\partial}{\partial\theta_i}\log\sum_{\mathbf{x}_S}\exp\left(\sum_j \phi_j(\mathbf{x}_{S_j},\mathbf{x}_{R_j},\theta_j)\right)$$

$$= \frac{\sum_{\mathbf{x}_S}\frac{\partial}{\partial\theta_i}\exp\left(\sum_j \phi_j(\mathbf{x}_{S_j},\mathbf{x}_{R_j},\theta_j)\right)}{\sum_{\mathbf{x}'_S}\exp\left(\sum_j \phi_j(\mathbf{x}'_{S_j},\mathbf{x}_{R_j}\theta_j)\right)}$$

$$= \sum_{\mathbf{x}_S}\frac{\exp(\sum_j \phi_j(\mathbf{x}_{S_j},\mathbf{x}_{R_j},\theta_j))}{\sum_{\mathbf{x}'_S}\exp(\sum_j \phi_j(\mathbf{x}'_{S_j},\mathbf{x}_{R_j},\theta_j))}\frac{\partial}{\partial\theta_i}\phi_i(\mathbf{x}_{S_i},\mathbf{x}_{R_i},\theta_i))$$

$$= \sum_{\mathbf{x}_S} P(\mathbf{x}_S|\mathbf{x}_R,\boldsymbol{\theta})\frac{\partial}{\partial\theta_i}\phi_i(\mathbf{x}_{S_i},\mathbf{x}_{R_i},\theta_i)) = \sum_{\mathbf{x}_S} P(\mathbf{x}_{S_i}|\mathbf{x}_R,\boldsymbol{\theta})\frac{\partial}{\partial\theta_i}\phi_i(\mathbf{x}_{C_i},\theta_i))$$

ML and Graphical Models

- Maximum likelihood not always straightforward in general graphs or graphs with hidden nodes.
- Need to be able to compute with both prior and posterior.

Let $\mathbf{x}^n = ((\mathbf{v}^n)^T, (\mathbf{h}^n)^T)^T$, and $P(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp(\sum_i \phi_i(\mathbf{x}_{C_i}|\theta_i))$
Using trick from previous slide.

$$\frac{\partial}{\partial\theta_i}\sum_n \log\sum_{\mathbf{h}^n} P(\mathbf{x}^n|\boldsymbol{\theta}) = \left[\sum_n\sum_{\mathbf{h}^n} P(\mathbf{x}^n_{C_i}|\boldsymbol{\theta},\mathbf{v}^n)\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}^n_{C_i}|\theta_i)\right] - N\frac{\partial}{\partial\theta}\log Z(\theta).$$

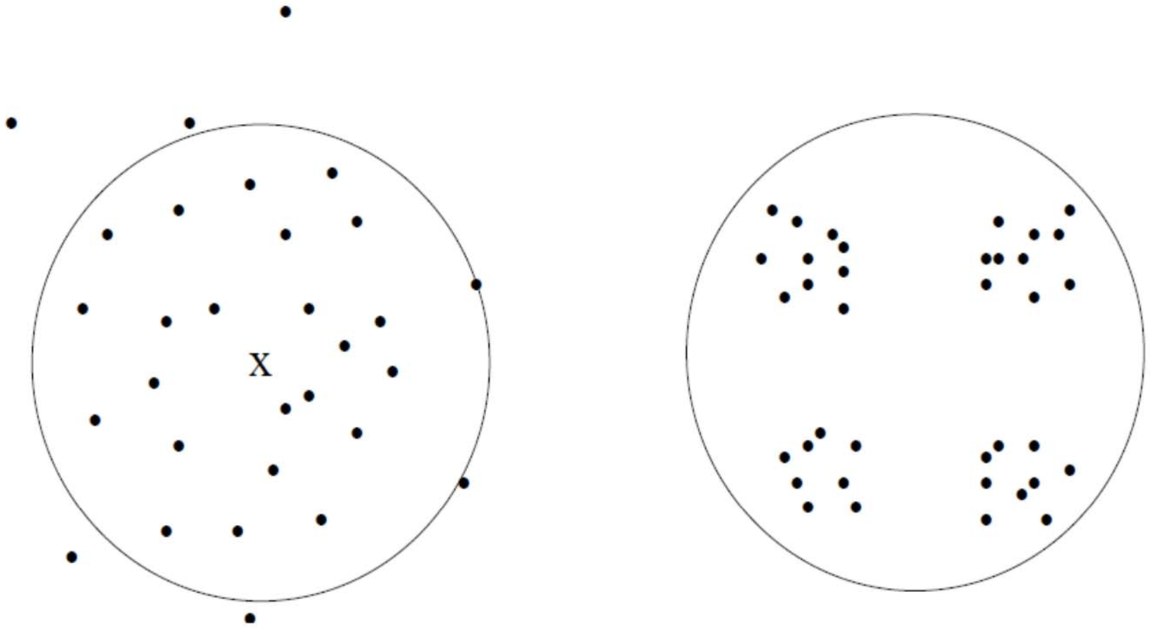
But $Z(\boldsymbol{\theta})$ is also a log sum as on previous slide, so we can rewrite

$$\frac{\partial}{\partial\theta_i}\sum_n \log\sum_{\mathbf{h}^n} P(\mathbf{x}^n|\boldsymbol{\theta}) = \sum_n \left[\sum_{\mathbf{h}^n_{C_i}} \overbrace{P(\mathbf{x}^n_{C_i}|\boldsymbol{\theta},\mathbf{v}^n)}^{\text{Posterior}}\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}^n_{C_i}|\theta_i)\right.$$

$$\left. - \sum_{\mathbf{x}'_{C_i}} \underbrace{P(\mathbf{x}'_{C_i}|\boldsymbol{\theta})}_{\text{Prior}}\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}'_{C_i}|\theta_i)\right]$$

Mixture Models

- A single Gaussian might be a poor fit



- Need mixture models for a *multimodal* density

Mixture Models

- Let ***z*** be a 1 of k indicator variable with $\sum_j z_j = 1$.
- $P(z_j = 1) = \pi_j$ is the probability of a given data point being created by component *j* (mixing proportion).
- $0 \leq \pi_j \leq 1 \; \forall j$ and $\sum_j \pi_j = 1$.
- $P_k(\mathrm{x}|\theta_k)$ is a mixing component

$$P(\mathbf{x}|\Theta) = \sum_{j=1}^{k} P(z_j = 1) P(\mathbf{x}|z_j = 1, \theta_k) = \sum_j \pi_j P_k(\mathbf{x}|\theta_k)$$

Maximizing Likelihood

$$P(D|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \left(\sum_{z^n} P(z^n|\boldsymbol{\pi}) P(\mathbf{x}^n|\boldsymbol{\theta}, z^n) \right)$$

$$= \sum_{n=1}^{N} \log \left\{ \sum_{j=1}^{N_J} \pi_j \exp \log P(\mathbf{x}^n|\theta_j) \right\}$$

Can use the same method as before. But now partion function $Z(\theta) = 1$, and so the second term falls out.

$$\frac{\partial P(D|\boldsymbol{\theta})}{\partial \theta_j} = \sum_n \sum_{z^n} P(z^n|\mathbf{x}^n) \frac{\partial \log p(\mathbf{x}^n|z^n)}{\partial \theta_j}$$

$$= \sum_n P(z^n = j|\mathbf{x}^n) \frac{\partial \log p(\mathbf{x}^n|\theta_j)}{\partial \theta_j}$$

$$\text{where } P(z^n = j|\mathbf{x}) = \frac{\pi_j p(\mathbf{x}^n|\theta_j)}{\sum_j \pi_j p(\mathbf{x}^n|\theta_j)}$$

are called the responsibilities, i.e. the probability of each component having generated a particular data point.

Iterating

- It turns out iteratively computing
  - the responsibilities
  - The best parameters for fixed responsibilities
- Is guaranteed to converge to a (local) maximum.
- Expectation Maximization (EM) Algorithm.
- More next lecture.

Our Journey

Graphical Models

Learning Probabilistic Models

- Introduction to Learning
- Learning exponential family models and Bayesian Set example.
- Approximate Learning and Maximum Posterior/Likelihood
- More Approximate Methods