Bayesian networks enable use of domain knowledge

\[ p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid x_{Pa(i)}) \]

Will my car start this morning?

Heckerman *et al.*, Decision-Theoretic Troubleshooting, 1995
Bayesian networks enable use of domain knowledge

\[ p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i \mid x_{\text{Pa}(i)}) \]

What is the differential diagnosis?

Beinlich et al., The ALARM Monitoring System, 1989
Bayesian networks are *generative models*

- Can sample from the joint distribution, top-down
- Suppose $Y$ can be “spam” or “not spam”, and $X_i$ is a binary indicator of whether word $i$ is present in the e-mail
- Let’s try generating a few emails!

- Often helps to think about Bayesian networks as a generative model when constructing the structure and thinking about the model assumptions
Inference in Bayesian networks

• Computing marginal probabilities in **tree** structured Bayesian networks is easy
  
  – The algorithm called “belief propagation” generalizes what we showed for hidden Markov models to arbitrary trees

• Wait... this isn’t a tree! What can we do?

---

**Diagrams**

1. A tree-structured Bayesian network with nodes labeled \( X_1, X_2, X_3, X_4, X_5, X_6 \) and \( Y_1, Y_2, Y_3, Y_4, Y_5, Y_6 \).
2. A non-tree structure with nodes labeled \( X_1, X_2, X_3, \ldots, X_{18} \) and a label node \( Y \).
Inference in Bayesian networks

- In some cases (such as this) we can *transform* this into what is called a “junction tree”, and then run belief propagation.

*Fig. 7*

Spiegelhalter’s algorithm re-arranges the ALARM network by triangulation and clique formation. The cliques are shaded differently to make them visible.
Approximate inference

- There is also a wealth of **approximate** inference algorithms that can be applied to Bayesian networks such as these
  
  - **Markov chain Monte Carlo algorithms** repeatedly sample assignments for estimating marginals
  
  - **Variational inference algorithms** (deterministic) find a simpler distribution which is “close” to the original, then compute marginals using the simpler distribution
Maximum likelihood estimation in Bayesian networks

- Suppose that we know the Bayesian network structure $G$
- Let $\theta_{x_i | x_{pa(i)}}$ be the parameter giving the value of the CPD $p(x_i | x_{pa(i)})$
- Maximum likelihood estimation corresponds to solving:

$$\max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \log p(x^M; \theta)$$

subject to the non-negativity and normalization constraints

- This is equal to:

$$\max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \log p(x^M; \theta) = \max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{N} \log p(x^M_i | x^M_{pa(i)}; \theta)$$

$$= \max_{\theta} \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \log p(x^M_i | x^M_{pa(i)}; \theta)$$

- The optimization problem decomposes into an independent optimization problem for each CPD! Has a simple closed-form solution.
Returning to clustering...

- Clusters may overlap
- Some clusters may be “wider” than others
- Can we model this explicitly?
- With what **probability** is a point from a cluster?
Probabilistic Clustering

• Try a probabilistic model!
  • allows overlaps, clusters of different size, etc.

• Can tell a **generative story** for data
  – \( P(Y)P(X|Y) \)

• **Challenge:** we need to estimate model parameters without labeled Ys
Gaussian Mixture Models

- \( P(Y) \): There are \( k \) components
- \( P(X | Y) \): Each component generates data from a **multivariate Gaussian** with mean \( \mu_i \) and covariance matrix \( \Sigma_i \)

Each data point assumed to have been sampled from a **generative process**:  

1. Choose component \( i \) with probability \( P(y = i) \) **[Multinomial]**  
2. Generate datapoint \( \sim N(m_i, \Sigma_i) \)

\[
P(X = x_j | Y = i) = \frac{1}{(2\pi)^{m/2} ||\Sigma_i||^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i) \right]
\]

**By fitting this model (unsupervised learning), we can learn new insights about the data**
Multivariate Gaussians

\[ P(X=x_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1} (x_j - \mu_i) \right] \]

\[ \Sigma \propto \text{identity matrix} \]
Multivariate Gaussians

\[ P(X=x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i) \right] \]

\[ \Sigma = \text{diagonal matrix} \]

\[ X_i \text{ are independent } a la \text{ Gaussian NB} \]
Multivariate Gaussians

$$P(X=x_j)= \frac{1}{(2\pi)^{m/2} ||\Sigma||^{1/2}} \exp\left[-\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1} (x_j - \mu_i)\right]$$

$$\Sigma = \text{arbitrary (semidefinite) matrix:}$$
- specifies rotation (change of basis)
- eigenvalues specify relative elongation
Multivariate Gaussians

Eigenvalue, $\lambda$, of $\Sigma$

Covariance matrix, $\Sigma$, = degree to which $x_i$ vary together

$$P(X=x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu)^T \Sigma^{-1} (x_j - \mu) \right]$$
Modelling eruption of geysers

Old Faithful Data Set

![Graph showing the relationship between time to eruption and duration of last eruption.](image)
Modelling eruption of geysers

Old Faithful Data Set

Single Gaussian

Mixture of two Gaussians
Marginal distribution for mixtures of Gaussians

\[ p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k) \]

\forall k : \pi_k \geq 0 \quad \sum_{k=1}^{K} \pi_k = 1

Component
Mixing coefficient

\[ K=3 \]

\[ p(x) \]

\[ x \]
Marginal distribution for mixtures of Gaussians
Learning mixtures of Gaussians

Original data (hypothesized)  Observed data (y missing)  Inferred y’s (learned model)

Shown is the *posterior probability* that a point was generated from \( i \)th Gaussian: \( \Pr(Y = i \mid x) \)
ML estimation in supervised setting

- Univariate Gaussian

$$\mu_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x_i$$
$$\sigma_{MLE}^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

- Mixture of Multivariate Gaussians

ML estimate for each of the Multivariate Gaussians is given by:

$$\mu_{ML}^k = \frac{1}{n} \sum_{j=1}^{n} x_n$$
$$\Sigma_{ML}^k = \frac{1}{n} \sum_{j=1}^{n} (x_j - \mu_{ML}^k)(x_j - \mu_{ML}^k)^T$$

Just sums over $x$ generated from the $k$'th Gaussian
What about with unobserved data?

- Maximize *marginal likelihood*:
  - \[ \text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{k=1}^K P(Y_j=k, x_j) \]

- Almost always a hard problem!
  - Usually no closed form solution
  - Even when \( \log P(X,Y) \) is convex, \( \log P(X) \) generally isn’t...
  - Many local optima
Expectation Maximization

1977: Dempster, Laird, & Rubin
The EM Algorithm

• A clever method for maximizing marginal likelihood:
  \[
  \arg\max_\theta \prod_j P(x_j) = \arg\max_\theta \prod_j \sum_{k=1}^K P(Y_j=k, x_j)
  \]
  – Based on coordinate descent. Easy to implement (eg, no line search, learning rates, etc.)

• Alternate between two steps:
  – Compute an expectation
  – Compute a maximization

• Not magic: *still optimizing a non-convex function with lots of local optima*
  – The computations are just easier (often, significantly so)
EM: Two Easy Steps

Objective: \( \arg\max_{\theta} \lg \prod_j \sum_{k=1}^K P(Y_j=k, x_j; \theta) = \sum_j \lg \sum_{k=1}^K P(Y_j=k, x_j; \theta) \)

Data: \( \{x_j \mid j=1..n\} \)

• **E-step:** Compute expectations to “fill in” missing y values according to current parameters, \( \theta \)
  – For all examples j and values k for \( Y_j \), compute: \( P(Y_j=k \mid x_j; \theta) \)

• **M-step:** Re-estimate the parameters with “weighted” MLE estimates
  – Set \( \theta^{new} = \arg\max_{\theta} \sum_j \sum_k P(Y_j=k \mid x_j; \theta^{old}) \log P(Y_j=k, x_j; \theta) \)

Particularly useful when the E and M steps have closed form solutions
Gaussian Mixture Example: Start
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
After 20th iteration
**EM for GMMs: only learning means (1D)**

**Iterate**: On the $t$'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

**E-step**

Compute “expected” classes of all datapoints

$$P(Y_j = k|x_j, \mu_1 \ldots \mu_K) \propto \exp\left(-\frac{1}{2\sigma^2}(x_j - \mu_k)^2\right)P(Y_j = k)$$

**M-step**

Compute most likely new $\mu$s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k|x_j) x_j}{\sum_{j=1}^{m} P(Y_j = k|x_j)}$$
What if we do hard assignments?

**Iterate:** On the $t$’th iteration let our estimates be 
\[ \lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \} \]

**E-step**

Compute “expected” classes of all datapoints
\[ P(Y_j = k \mid x_j, \mu_1 \ldots \mu_K) \propto \exp\left(-\frac{1}{2\sigma^2} (x_j - \mu_k)^2\right) P(Y_j = k) \]

**M-step**

Compute most likely new $\mu$s given class expectations
\[ \mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k \mid x_j) x_j}{\sum_{j=1}^{m} P(Y_j = k \mid x_j)} \]

$\delta$ represents hard assignment to “most likely” or nearest cluster

Equivalent to k-means clustering algorithm!!!
E.M. for General GMMs

Iterate: On the t’th iteration let our estimates be
\[ \lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_K^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_K^{(t)} \} \]

E-step

Compute “expected” classes of all datapoints for each class
\[ P(Y_j = k|x_j; \lambda_t) \propto p_k^{(t)} p(x_j; \mu_k^{(t)}, \Sigma_k^{(t)}) \]

M-step

Compute weighted MLE for \( \mu \) given expected classes above
\[ \mu_k^{(t+1)} = \frac{\sum_j P(Y_j = k|x_j; \lambda_t) x_j}{\sum_j P(Y_j = k|x_j; \lambda_t)} \quad \Sigma_k^{(t+1)} = \frac{\sum_j P(Y_j = k|x_j; \lambda_t) [x_j - \mu_k^{(t+1)}][x_j - \mu_k^{(t+1)}]^T}{\sum_j P(Y_j = k|x_j; \lambda_t)} \]
\[ p_k^{(t+1)} = \frac{\sum_j P(Y_j = k|x_j; \lambda_t)}{m} \]

\( p_k^{(t)} \) is shorthand for estimate of \( P(y=k) \) on t’th iteration

Evaluate probability of a multivariate a Gaussian at \( x_j \)

\( m = \#training \text{ examples} \)
The general learning problem with missing data

- **Marginal likelihood:** $X$ is observed,

  $Z$ (e.g. the class labels $Y$) is missing:

  $$
  \ell(\theta : D) = \log \prod_{j=1}^{m} P(x_j | \theta) \\
  = \sum_{j=1}^{m} \log P(x_j | \theta) \\
  = \sum_{j=1}^{m} \log \sum_z P(x_j, z | \theta)
  $$

- **Objective:** Find $\text{argmax}_\theta I(\theta : \text{Data})$

- **Assuming hidden variables are missing completely at random** (otherwise, we should explicitly model why the values are missing)
Properties of EM

• One can prove that:
  – EM converges to a local maxima
  – Each iteration improves the log-likelihood

• How? (Same as k-means)
  – Likelihood objective instead of k-means objective
  – M-step can never decrease likelihood
Derivation of EM algorithm

\[ L(\theta) = \sum_z P(z|X, \theta_n) \ln P(X|z, \theta_n) P(z|\theta_n) \]

We have now a function, \( l(\theta|\theta_n) \) which is bounded above by the likelihood function \( L(\theta) \). Additionally, observe that,

\[ l(\theta_n|\theta_n) = \sum_z P(z|X, \theta_n) \ln P(X|z, \theta_n) P(z|\theta_n) \]

so for \( \theta = \theta_n \) the functions \( l(\theta|\theta_n) \) and \( L(\theta) \) are equal. Therefore, any \( \theta \) which increases \( l(\theta|\theta_n) \) will also increase \( L(\theta) \). In order to achieve the greatest possible increase in the value of \( L(\theta_n) \), the EM algorithm calls for selecting \( \theta \) such that \( l(\theta|\theta_n) \) is maximized. We denote this updated value as \( \theta_{n+1} \). This process is illustrated in Figure 2.

(Figure from tutorial by Sean Borman)
What you should know

• Mixture of Gaussians

• EM for mixture of Gaussians:
  – How to learn maximum likelihood parameters in the case of unlabeled data
  – Relation to K-means
    • Two step algorithm, just like K-means
    • Hard / soft clustering
    • Probabilistic model

• Remember, EM can get stuck in local minima,
  – And empirically it *DOES*