Lecture 12 - *Gaussian Processes*

OBJECTIVE: In this lecture we introduce gaussian processes (GPs) for regression. These nonparametric models allow us to generate nonlinear predictions. Yet, the models are very tractable and permit the application of experimental design and active learning ideas.

◇ GAUSSIAN PROCESSES

A Gaussian process, denoted $\mathbf{z}(\cdot) \sim GP(m(\cdot), K(\cdot,\cdot))$, is an infinite random process indexed by a variable $\mathbf{x}$ such that any realization $\mathbf{z}(\mathbf{x}_i)$ is Gaussian with mean $m(\mathbf{x}_i)$ and covariance (symmetric positive definite kernel) $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

The Gaussian process prior is denoted:

$$\mathbf{z}(\cdot) \;\sim\; GP(m, K)$$

The most popular kernels are the following:

- Polynomial

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \mathbf{x}_j^T + b)^p$$

- Gaussian

$$k(x_i, x_j) = e^{-\frac{1}{2\sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T}$$

- Sigmoid (logistic, neural network)

$$k(x_i, x_j) = tanh(\alpha \mathbf{x}_i \mathbf{x}_j^T - \beta)$$

On observing data $\{\mathbf{x}_{1:n}, \mathbf{z}_{1:n}\}$ and a test point $\mathbf{x}_{n+1}$ (or more test points), the vector of predictions on the test and training data is jointly Gaussian:

$$\begin{bmatrix} z_{n+1} \\ \mathbf{z}_{1:n} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}_{n+1}) \\ m(\mathbf{x}_{1:n}) \end{bmatrix}, \begin{bmatrix} k & \mathbf{k}^T \\ \mathbf{k} & \mathbf{K} \end{bmatrix} \right)$$

where $k = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})$,

$$\mathbf{k}^T = \begin{bmatrix} k(\mathbf{x}_{n+1}, \mathbf{x}_1) & \cdots & k(\mathbf{x}_{n+1}, \mathbf{x}_n) \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

The above expression for $p(\mathbf{z}_{1:n+1}|\mathbf{x}_{1:n+1})$ follows from the definition of GPs. We can now proceed to use standard manipulation of Gaussian distributions to obtain the predictive distribution: $p(z_{n+1}|\mathbf{x}_{1:n+1}, \mathbf{z}_{1:n})$, or simply

$$\begin{aligned} \mathbb{E}(z_{n+1}|\mathbf{x}_{1:n+1}, \mathbf{z}_{1:n}) &= m(\mathbf{x}_{n+1}) + \mathbf{k}^T\mathbf{K}^{-1}(\mathbf{z}_{1:n} - m(\mathbf{x}_{1:n})) \\ cov(z_{n+1}|\mathbf{x}_{1:n+1}, \mathbf{z}_{1:n}) &= k - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \end{aligned}$$

That is, we have a simple closed form expression to make nonlinear predictions!

◇ GPS WITH NOISE

Typically, the data is noisy, so one considers the regression model:

$$\mathbf{y}(\mathbf{x}) = \mathbf{z}(\mathbf{x}) + \mathcal{N}(0, \sigma^2 I)$$

This only involves a change in the covariance (kernel) function:

$$\mathbf{y}(\cdot) \sim GP(m, K + \sigma^2 I)$$

To do binary classification, we adopt a Bernoulli logistic model as we did when learning importance sampling.

◇ GP WITH UNKNOWN MEAN FUNCTION

Since the mean function is typically unknown, one can construct a GP for prediction as follows:

$$\mathbf{y}(\mathbf{x}) \;=\; \sum_{j=1}^{d} f_j(\mathbf{x})\theta_j + GP(0, \sigma^2\mathbf{K})$$

where the f's are some features of the input space.

On observing data $\{\mathbf{x}_{1:n}, \mathbf{y}_{1:n}\}$ and a test point $\mathbf{x}_{n+1}$ (or more test points), the vector of predictions on the test and training data is jointly Gaussian:

$$\begin{bmatrix} y_{n+1} \\ \mathbf{y}_{1:n} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{F} \end{bmatrix} \boldsymbol{\theta}, \sigma^2 \begin{bmatrix} k & \mathbf{k}^T \\ \mathbf{k} & \mathbf{K} \end{bmatrix} \right)$$

where, again, $k = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})$,

$$\begin{aligned} \mathbf{k}^T &= \begin{bmatrix} k(\mathbf{x}_{n+1}, \mathbf{x}_1) & \cdots & k(\mathbf{x}_{n+1}, \mathbf{x}_n) \end{bmatrix} \\ \mathbf{K} &= \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \\ \mathbf{f} &= \begin{bmatrix} f_1(\mathbf{x}_{n+1}) & \cdots & f_d(\mathbf{x}_{n+1}) \end{bmatrix} \\ \mathbf{F} &= \begin{bmatrix} f_1(\mathbf{x}_1) & \cdots & f_d(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ f_1(\mathbf{x}_n) & \cdots & f_d(\mathbf{x}_n) \end{bmatrix} \end{aligned}$$

As in the linear model, we can assign a Gaussian prior $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}_0, \sigma^2 R^{-1})$. This prior results in a Gaussian posterior with mean and covariance:

$$\begin{aligned} \boldsymbol{\mu} = \mathbb{E}(\boldsymbol{\theta}|\mathbf{F}, \mathbf{y}) &= (\mathbf{F}^T\mathbf{K}^{-1}\mathbf{F} + R)^{-1}(\mathbf{F}^T\mathbf{K}^{-1}\mathbf{y} + R\boldsymbol{\theta}_0) \\ cov(\boldsymbol{\theta}|\mathbf{F}, \mathbf{y}) &= (\mathbf{F}^T\mathbf{K}^{-1}\mathbf{F} + R)^{-1}\sigma^2 = \Sigma\sigma^2 \end{aligned}$$

Under this posterior distribution, the predictive distribution of the Gaussian process becomes:

$$\mathbb{E}(\mathbf{y}_{n+1}|\mathbf{y}_{1:n}) \;=\; \mathbf{f}\boldsymbol{\mu} + \mathbf{k}^T\mathbf{K}^{-1}(\mathbf{y}_{1:n} - \mathbf{F}\boldsymbol{\mu})$$
$$cov(\mathbf{y}_{n+1}|\mathbf{y}_{1:n}) = \sigma^2\left\{k - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} + (\mathbf{f} - \mathbf{F}^T\mathbf{K}^{-1}\mathbf{k})^T\Sigma(\mathbf{f} - \mathbf{F}^T\mathbf{K}^{-1}\mathbf{k})\right\}$$

◇ ACTIVE LEARNING WITH GAUSSIAN PROCESSES

We choose an experiment (point x) so as to minimize the variance of the prediction. That is,

$$u^\star(\mathbf{x}) = \min_{\mathbf{x}} cov(\mathbf{y}_{n+1}|\mathbf{y}_{1:n})$$

That is, we should choose to experiment in parts of the space where there is high uncertainty (also known as entropy).
In the 1D case the covariance is really a univariate variance as we only have a single test point. In general we can try to minimize either the determinant or trace (better) of the predictive covariance for a block of test points.

Lecture 12 - *Unsupervised Learning*

OBJECTIVE: When there are no labels $\mathbf{y}$, there are many useful patterns we can still find in the data. Here we introduce two of the most popular algorithms in machine learning and data mining: K-means and EM.

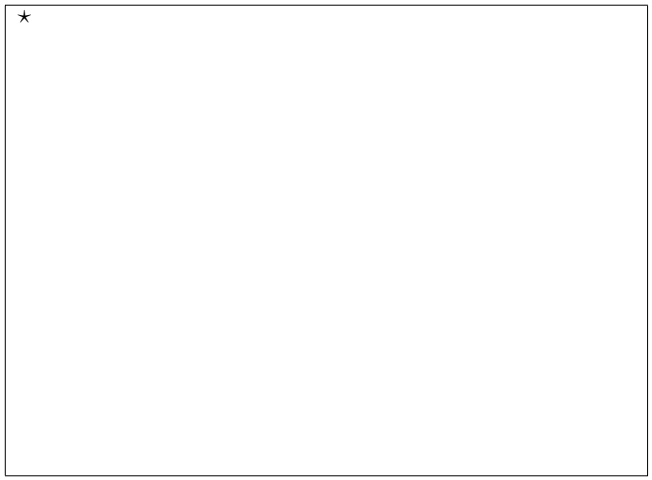
Unsupervised Learning

Our goal is to automatically discover patterns and structure in the data. Some examples include

- Novelty detection (e.g. detecting new strands of HIV).
- Data association (e.g. machine translation, multi-target tracking, object recognition from annotated images).
- Clustering.

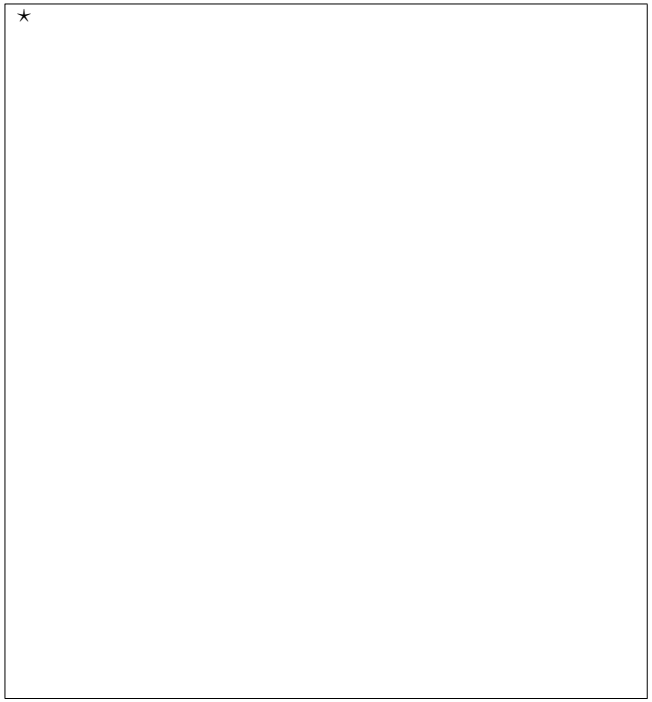
Clustering

Assume we are given the data $x_{1:n}$, with $x_i \in \mathbb{R}^2$.



We want to find clusters (groups where data items are similar).

K-means



K-means is a simple iterative algorithm for clustering data.

In our 2D example, it proceeds as follows

1. **Initialisation:** Choose $k = 2$ means $\mu_{1:2}$ at random.
2. **Compute distances:** For $c = 1, \ldots, k$ and $i = 1, \ldots, n$ compute the distance $\|x_i - \mu_c\|^2$.
3. **Assign data to nearest mean:** To keep track of assignments, introduce the indicator variable z_i, such that

$$\mathbb{I}_c(z_i) = \begin{cases} 1 & \text{if } c = \arg\min_{c'} \|x_i - \mu_{c'}\|^2 \\ 0 & \text{otherwise} \end{cases}$$

   That is, $\mathbb{I}_2(z_i) = 1$ if observation x_i is closer to cluster 2. $\mathbb{I}_c(z_i)$ end up being the entries of an $n \times k$ matrix with only one 1 per row and many zeros.
4. **Update means:**

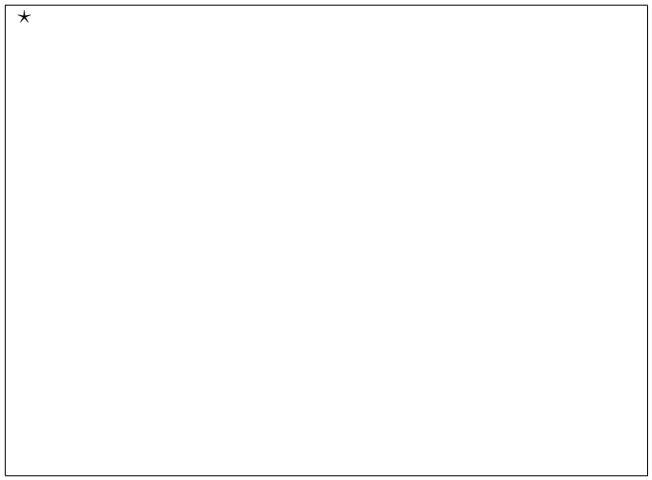
$$\mu_c = \frac{\sum_{i=1}^n \mathbb{I}_c(z_i) x_i}{\sum_{i=1}^n \mathbb{I}_c(z_i)}$$

5. **Repeat:** Go back to step 2. until the means and assignments stop changing.

The problem with this algorithm is that the assignments are hard. Something is either this or that. Sometimes, however, we would like to say that something is this with probability 0.7 or that with probability 0.3.

Finite Mixtures of Gaussians

We would like to find not only the means, but also the variances of each cluster and the probabilities of belonging to each cluster.



In the 2 cluster case we have:

$$\begin{aligned} p(x_i|\mu_{1,2}, \sigma^2_{1:2}) &= p(z_i = 1)\mathcal{N}(x_i|\mu_1, \sigma^2_1) \\ &+ p(z_i = 2)\mathcal{N}(x_i|\mu_2, \sigma^2_2) \end{aligned}$$

In general, we have

$$p(x_i|\theta) = \sum_{c=1}^{k} p(c)\mathcal{N}(x_i|\mu_c, \sigma^2_c)$$

where $\theta = (\mu_{1:c}, \sigma^2_{1:c})$ summarises the model parameters and $p(c) = p(z_i = c)$. Clearly, $\sum_{c=1}^{k} p(c) = 1$. It will be convenient for future derivations, to consider the following alternative way of writing a mixture model:

$$p(x_i, z_i = c|\theta) = \prod_{c=1}^{k} \left[p(c)\mathcal{N}(x_i|\mu_c, \sigma^2_c)\right]^{\mathbb{I}_c(z_i)}$$

EM for Mixtures of Gaussians

In this section, we use intuition to introduce the expectation-maximisation (EM) (later we will derive it formally). If we know $\mathbb{I}_c(z_i)$, then it is easy to compute (μ_c, σ^2_c) by maximum likelihood. We repeat this for each cluster. The problem is that we have a chicken and egg situation. To know the cluster memberships, we need the parameters of the Gaussians. To know the parameters, we need the cluster memberships.
One solution is to approximate $\mathbb{I}_c(z_i)$ with our expectation of it given the data and our current estimate of the parameters

θ. That is, we replace $\mathbb{I}_c(z_i)$ with

$$\xi_{ic} \triangleq \mathbb{E}\left[\mathbb{I}_c(z_i)|x_i, \theta\right]$$

⋆

$$\xi_{ic} \triangleq \mathbb{E}\left[\mathbb{I}_c(z_i)|x_i, \theta\right] =$$

Note that ξ_{ic} is a soft-assignments matrix:

⋆

Once we know ξ_{ic}, we can compute the Gaussian mixture parameters:

$$\begin{aligned}
\mu_c &= \frac{\sum_{i=1}^n \xi_{ic} x_i}{\sum_{i=1}^n \xi_{ic}} \\
\Sigma_c &= \frac{\sum_{i=1}^n \xi_{ic}(x_i - \mu_c)(x_i - \mu_c)'}{\sum_{i=1}^n \xi_{ic}} \\
p(c) &= \frac{1}{n}\sum_{i=1}^n \xi_{ic}
\end{aligned}$$

The EM for Gaussians is as follows:

1. **Initialise.**
2. **E Step:** At iteration t, compute the expectation of the

indicators for each i and c:

$$\xi_{ic}^{(t)} = \frac{p(c)^{(t)}\mathcal{N}(x_i|\mu_c^{(t)}, \Sigma_c^{(t)})}{\sum_{c'=1}^{k} p(c')^{(t)}\mathcal{N}(x_i|\mu_{c'}^{(t)}, \Sigma_{c'}^{(t)})}$$

and normalise it (divide by sum over c).

3. **M Step:** Update the parameters $p(c)^{(t)}, \mu_c^{(t)}, \Sigma_c^{(t)}$.

EM for Categorical Data

Matrix data with categorical entries is ubiquitous on the web. Examples include word-document matrices used for information retrieval and item-judge ratings for collaborative

filtering:

$$\begin{array}{c} \\ documents_{1:n} \end{array} \begin{array}{c} words_{1:L} \\ \begin{bmatrix} 3 & 1 & \dots & 6 \\ 7 & 4 & \dots & 2 \\ \vdots & & \ddots & \end{bmatrix} \end{array} = T_{1:n}$$

$$\begin{array}{c} \\ movies_{1:n} \end{array} \begin{array}{c} judges_{1:L} \\ \begin{bmatrix} 1 & ? & \dots & 5 \\ 5 & 4 & \dots & ? \\ \vdots & & \ddots & \end{bmatrix} \end{array} = M_{1:n}$$

Each row of N can be modelled with a multinomial distribution.

$$p(T_i) \propto \prod_{w=1}^{n_w} \delta_w^{T_{iw}}$$

We assume that the text documents are i.i.d.

$$p(T_{1:n}) = \prod_{i=1}^{n} p(T_i)$$

Suppose the documents come from n_c distinct clusters. Then

they are distributed according to a mixture of multinomials:

$$p(T_i|\theta) = \sum_{c=1}^{n_c} p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

The maximum likelihood EM for Categorical data involves iterating the following two steps:

1. **E Step:** At iteration t, compute the expectation of the indicators for each i and c:

$$\xi_{ic} \propto p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

   and normalise it (divide by sum over c).

2. **M Step:** Update the parameters:

$$\begin{aligned} \delta_{wc} &= \frac{\sum_{i=1}^{n} \xi_{ic} T_{iw}}{\sum_{i=1}^{n} \sum_{w=1}^{n_w} T_{iw} \xi_{ic}} \\ p(c) &= \frac{1}{n} \sum_{i=1}^{n} \xi_{ic} \end{aligned}$$

EM for Multimodal Data

A document with an image (represented by a vector $x_i \in \mathbb{R}^d$) and text T_i can be modelled as follows:

$$p(d_i|\theta) = \sum_{c=1}^{n_c} p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

The maximum likelihood EM is as follows:

1. **E Step:** At iteration t, compute the expectation of the indicators for each i and c:

$$\xi_{ic} \propto p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

   and normalise it (divide by sum over c).

2. **M Step:** Update the parameters:

$$\begin{aligned}
\mu_c &= \frac{\sum_{i=1}^n \xi_{ic} x_i}{\sum_{i=1}^n \xi_{ic}} \\
\Sigma_c &= \frac{\sum_{i=1}^n \xi_{ic}(x_i - \mu_c)(x_i - \mu_c)'}{\sum_{i=1}^n \xi_{ic}} \\
\delta_{wc} &= \frac{\sum_{i=1}^n \xi_{ic} T_{iw}}{\sum_{i=1}^n \sum_{w=1}^{n_w} T_{iw} \xi_{ic}} \\
p(c) &= \frac{1}{n} \sum_{i=1}^n \xi_{ic}
\end{aligned}$$

General Derivation of EM for ML

Our goal, in general, is to maximise the (log)-likelihood of the data x given the parameters θ. That is, we want to solve the optimisation problem:

$$\theta^\star = \arg\max_\theta \quad \log p(x_{1:n}|\theta)$$

For notational simplicity, let $x \triangleq x_{1:n}$.

Sometimes it helps to introduce the variables $z \triangleq z_{1:n}$. These may simplify the optimisation problem:

They may also decouple a complex optimisation problem into simpler ones:

⋆

We now prove that the problem:

$$\theta^\star = \arg\max_\theta \ \log p(x_{1:n}|\theta)$$

can be solved by solving the following problem:

$$\begin{aligned}\theta^\star &= \arg\max_\theta \ \mathbb{E}[\log p(x,z|\theta)] \\ &= \arg\max_\theta \ \int [\log p(x,z|\theta)] p(z|x,\theta^{\text{old}}) dz\end{aligned}$$

where θ^{old} denotes the old (previous) estimate of θ.

Step 1: We prove

$$\log p(x|\theta) = \mathbb{E}[\log p(x,z|\theta)] - \mathbb{E}[\log p(z|x,\theta)]$$

⋆

$$p(x|\theta) = p(x|\theta)\frac{p(z|x,\theta)}{p(z|x,\theta)}$$

$$\log p(x|\theta) =$$

Step 2: We prove

$$\mathbb{E}[\log p(z|x, \theta^{\text{old}})] \geq \mathbb{E}[\log p(z|x, \theta)]$$

⋆ Hint: $\log x \leq x - 1$

$$\mathbb{E}[\log p(z|x, \theta)] - \mathbb{E}[\log p(z|x, \theta^{\text{old}})]$$

$$=$$

Step 3: We note

$$\begin{aligned}\log p(x|\theta) - \log p(x|\theta^{\text{old}}) &= \left\{\mathbb{E}[\log p(x,z|\theta)] - \mathbb{E}[\log p(x,z|\theta^{\text{old}})]\right\} \\ &+ \left\{\mathbb{E}[\log p(z|x,\theta^{\text{old}})] - \mathbb{E}[\log p(z|x,\theta)]\right\}\end{aligned}$$

Hence to increase $\log p(x|\theta)$, we only have to increase $Q(\theta, \theta^{\text{old}}) \triangleq \mathbb{E}[\log p(x,z|\theta)]$. This completes the proof.

The **general EM** algorithm is as follows:

1. **Initialise.**
2. **E Step:** Given θ^{old}, compute the expectation $Q(\theta, \theta^{\text{old}})$.
3. **M Step:** Maximise $Q(\theta, \theta^{\text{old}})$ by differentiating it with respect to θ and equating it to zero.

The E and M steps are iterated until a local maximum of the likelihood is reached.

EM for Mixture Models

Assume that we are given the i.i.d. data $x_{1:n}$. The Q function for a mixture model of this data is:

$$\begin{aligned}Q(\theta, \theta^{\text{old}}) &= \mathbb{E}_{p(z|x,\theta^{\text{old}})}\left\{\log\left[\prod_{i=1}^{n} p(x_i, z_i = c|\theta)\right]\right\} \\ &= \mathbb{E}_{p(z|x,\theta^{\text{old}})}\left\{\log\left[\prod_{i=1}^{n}\prod_{c=1}^{n_c} (p(c)p(x_i|\theta_c))^{\mathbb{I}_c(z_i)}\right]\right\} \\ &= \mathbb{E}_{p(z|x,\theta^{\text{old}})}\left\{\sum_{i=1}^{n}\sum_{c=1}^{n_c} \mathbb{I}_c(z_i)\log\left[p(c)p(x_i|\theta_c)\right]\right\} \\ &= \sum_{c_1=1}^{n_c}\cdots\sum_{c_n=1}^{n_c}\left\{\sum_{i=1}^{n}\sum_{c=1}^{n_c} \mathbb{I}_c(z_i)\log\left[p(c)p(x_i|\theta_c)\right]\right\}\prod_{j=1}^{n} p(z_j|x_j,\theta^{\text{old}}) \\ &= \sum_{i=1}^{n}\sum_{c=1}^{n_c}\log\left[p(c)p(x_i|\theta_c)\right]\sum_{c_1=1}^{n_c}\cdots\sum_{c_n=1}^{n_c}\mathbb{I}_c(z_i)\prod_{j=1}^{n} p(z_j|x_j,\theta^{\text{old}})\end{aligned}$$

This expression can be greatly simplified by noticing that

$$\sum_{c_1=1}^{n_c} \cdots \sum_{c_n=1}^{n_c} \mathbb{I}_c(z_i) \prod_{j=1}^{n} p(z_j|x_j, \theta^{\text{old}})$$

$$= \left[\sum_{c_1=1}^{n_c} \cdots \sum_{c_{i-1}=1}^{n_c} \sum_{c_{i+1}=1}^{n_c} \cdots \sum_{c_n=1}^{n_c} \prod_{j=1, j\neq i}^{n} p(z_j|x_j, \theta^{\text{old}}) \right] p(z_i = c|x_i, \theta^{\text{old}})$$

$$= \prod_{j=1, j\neq i}^{n} \left[\sum_{c_j=1}^{n_c} p(z_j|x_j, \theta^{\text{old}}) \right] p(z_i = c|x_i, \theta^{\text{old}})$$

$$= \quad p(z_i = c|x_i, \theta^{\text{old}})$$

Consequently, the Q function simplifies to:

$$Q(\theta, \theta^{\text{old}}) \;=\; \sum_{i=1}^{n} \sum_{c=1}^{n_c} p(z_i = c|x_i, \theta^{\text{old}}) \log \left[p(c) p(x_i|\theta_c) \right]$$

The **EM algorithm for mixtures** is as follows:

1. **Initialise.**
2. **E Step:** Given θ^{old}, compute $p(z_i = c|x_i, \theta^{\text{old}})$.
3. **M Step:** Maximise $Q(\theta, \theta^{\text{old}})$ by differentiating it with respect to $p(c)$ and θ_c. One should add constraints to ensure that all probabilities sum to 1.

EM for Gaussian Mixtures

The data $x_{1:n}$, with $x_i \in \mathbb{R}^d$, is distributed according to the following normal mixture

$$p(x_i|\theta) = \sum_{c=1}^{n_c} p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c)$$

Hence, the Q function is:

$$Q(\theta, \theta^{\text{old}}) \;=\; \sum_{i=1}^{n} \sum_{c=1}^{n_c} \xi_{ic} \log \left[p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c) \right]$$

We know that the E step consists of computing the ξ_{ic}:

⋆

$$\xi_{ic} =$$

The M step requires that we maximise Q subject to the constraint $\sum_c p(c) = 1$. To estimate $p(c)$, we have:

⋆

To compute the remaining parameters, we expand the Q

function:

$$\begin{aligned} Q(\theta, \theta^{\text{old}}) &= \sum_{i=1}^{n}\sum_{c=1}^{n_c} \xi_{ic} \log\left[|\Sigma_c|^{-\frac{1}{2}} e^{-\frac{1}{2}(x_i-\mu_c)'\Sigma_c^{-1}(x_i-\mu_c)}\right] \\ &= \sum_{i=1}^{n}\sum_{c=1}^{n_c} \xi_{ic}\left[\frac{1}{2} log|\Sigma_c|^{-1} - \frac{1}{2}(x_i-\mu_c)'\Sigma_c^{-1}(x_i-\mu_c)\right] \end{aligned}$$

Note that we got rid of constant terms as these don't affect the location of the maxima for μ_c and Σ_c.

⋆

$$\frac{\partial Q}{\partial \mu_c} =$$

⋆

$$\frac{\partial Q}{\partial \Sigma_c^{-1}} =$$

EM for Discrete Mixtures

Suppose the matrix data (e.g. documents) comes from n_c distinct clusters. That is, it is distributed according to a

mixture of multinomials:

$$p(T_i|\theta) = \sum_{c=1}^{n_c} p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

where T_{iw} means the number of words w in document i. The Q function is:

⋆

$$Q(\theta, \theta^{\text{old}}) =$$

The E step is as follows:

⋆

$$\xi_{ic} =$$

In the M step for categorical variables, we need to maximise Q subject to the constraints $\sum_{c=1}^{n_c} p(c) = 1$ and $\sum_{w}^{n_w} \delta_{wc} = 1$. To compute $p(c)$, we introduce Lagrange multipliers, μ,

and maximise the Lagrangian

$$\mathcal{L}(p(c), \mu) = Q + \mu \left(1 - \sum_{c=1}^{n_c} p(c)\right)$$

by differentiating with respect to $p(c)$ and equating to zero. That is, we want to compute

$$\frac{\partial}{\partial p(c)} \left\{ \sum_{i=1}^{n} \sum_{c=1}^{n_c} \xi_{ic} \log \left[p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}} \right] + \mu \left(1 - \sum_{c=1}^{n_c} p(c)\right) \right\} = 0$$

$$\sum_{i=1}^{n} \xi_{ic} \frac{1}{p(c)} - \mu = 0$$

Summing both sides over c, we get $\mu = n$. Therefore, the estimate for $p(c)$ is

$$p(c) = \frac{1}{n} \sum_{i=1}^{n} \xi_{ic}$$

The estimate for δ is obtained following the same steps:

$$\frac{\partial}{\partial \delta_{wc}} \left\{ \sum_{i=1}^{n} \sum_{c=1}^{n_c} \xi_{ic} \log \left[p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}} \right] + \mu \left(1 - \sum_{w}^{n_w} \delta_{wc} \right) \right\} = 0$$

$$\sum_{i=1}^{n} \xi_{ic} T_{iw} \frac{1}{\delta_{wc}} - \mu = 0$$

Summing both sides over w, we get $\mu = \sum_{i=1}^{n} \sum_{w=1}^{n_w} \xi_{ic} T_{iw}$ and, hence, the estimate of δ_{wc} is

$$\delta_{wc} = \frac{\sum_{i=1}^{n} \xi_{ic} T_{iw}}{\sum_{i=1}^{n} \sum_{w=1}^{n_w} \xi_{ic} T_{iw}}$$

EM MAP Estimation

The EM formulation for MAP estimation is straightforward. One simply has to augment the objective function in the M step, Q^{ML}, by adding to it the log prior densities. That is,

the MAP objective function is

$$\begin{aligned} Q^{\text{MAP}} &= \mathbb{E}_{p(z|x,\theta^{(\text{old})})} [\log p(z, x, \theta)] \\ &= \mathbb{E}_{p(z|x,\theta^{(\text{old})})} [\log p(z, x|\theta) + \log p(\theta)] \\ &= Q^{\text{ML}} + \log p(\theta) \end{aligned}$$

EM MAP Estimates for Categorical Mixtures

The unconstrained objective function is

$$Q^{\text{MAP}} = Q^{\text{ML}} + \log p(\theta)$$

Since the distributions of z_i and T_{iw} are multinomial, we adopt conjugate Dirichlet priors for $p(c)$ and $\delta_{w,c}$. That is, our priors are:

$$\begin{aligned} p(p(c)) &= \frac{\Gamma(\sum_{c=1}^{n_c} \alpha_c)}{\prod_{c=1}^{n_c} \Gamma(\alpha_c)} \prod_{c=1}^{n_c} p(c)^{\alpha_c - 1} \\ p(\delta_{wc}) &= \prod_{c=1}^{n_c} \frac{\Gamma(\sum_{w}^{n_w} \beta_{wc})}{\prod_{w=1}^{n_w} \Gamma(\beta_{wc})} \prod_{w=1}^{n_w} \delta_{wc}^{\beta_{wc} - 1} \end{aligned}$$

To compute $p(c)$, we proceed as in the previous section by

differentiating the augmented Lagrangian with respect to $p(c)$ and equating to zero

$$\frac{\partial}{\partial p(c)}\left\{\sum_{i=1}^{n}\sum_{c=1}^{n_c}\xi_{ic}\log\left[p(c)\prod_{w=1}^{n_w}\delta_{wc}^{T_{iw}}\right]+\mu\left(1-\sum_{c=1}^{n_c}p(c)\right)\right\}+ \log\left[\frac{\Gamma(\sum_{c=1}^{n_c}\alpha_c)}{\prod_{c=1}^{n_c}\Gamma(\alpha_c)}\prod_{c=1}^{n_c}p(c)^{\alpha_c-1}\right]\Bigg\} = 0$$
$$\sum_{i=1}^{n}(\xi_{ic}+\alpha_c-1)\frac{1}{p(c)}-\mu = 0$$

Summing both sides over c, we get $\mu = n - n_c + \sum_c \alpha_c$. Hence, the estimate for $p(c)$ is

$$p(c)=\frac{\sum_{i=1}^{n}\xi_{ic}+\alpha_c-1}{n+\sum_c\alpha_c-n_c}$$

Similarly, the constrained maximisation for δ_{wc} requires that

we compute

$$\frac{\partial}{\partial\delta_{wc}}\left\{\sum_{i=1}^{n}\sum_{c=1}^{n_c}\xi_{ic}\log\left[p(c)\prod_{w=1}^{n_w}\delta_{wc}^{T_{iw}}\right]+\mu\left(1-\sum_{w}^{n_w}\delta_{wc}\right)\right\}+ \log\left[\prod_{c=1}^{n_c}\frac{\Gamma(\sum_{w}^{n_w}\beta_{wc})}{\prod_{w=1}^{n_w}\Gamma(\beta_{wc})}\prod_{w=1}^{n_w}\delta_{wc}^{\beta_{wc}-1}\right]\Bigg\} = 0$$
$$\sum_{i=1}^{n}(\xi_{ic}T_{iw}+\beta_{wc}-1)\frac{1}{\delta_{wc}}-\mu = 0$$

thus yielding

$$\delta_{wc}=\frac{\sum_{i=1}^{n}\xi_{ic}T_{ic}+\beta_{wc}-1}{\sum_{w=1}^{n_w}(\sum_{i=1}^{n}\xi_{ic}T_{ic}+\beta_{wc}-1)}$$

Note that if the α and β hyper-paramemeters are set to 1, we obtain the ML estimates. Typically, α is set to 1 and β is set to a number higher than 2. This choice of hyper-parameters allows for smoothing. It forces all the δ_{wc} to have have a significant value (each word has a significant probability of belonging to each cluster) and hence many of the $p(c)$ go to zero. This is an easy way of automatically choosing the number of mixture components.

Lecture 13 - *Dynamic Models*

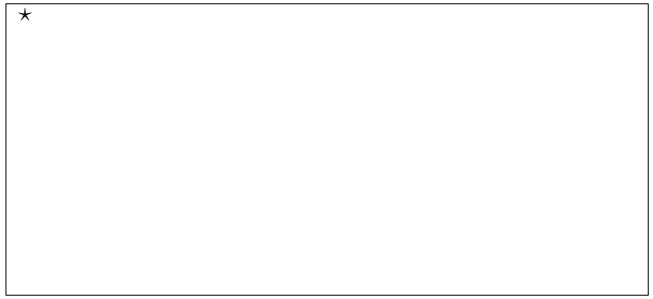
OBJECTIVE: In this lecture, we introduce dynamic models, discuss some applications, and derive Kalman, HMM and particle filtering.

◇ DYNAMIC MODELS

A dynamic model consists of three equations: the initial probability, the transition model and the observation model. The unobserved signal (hidden states or unknown parameters) $\{x_t; t \in \mathbb{N}\}$, $x_t \in \mathcal{X}$, is modelled as a *Markov process* of initial distribution $p(x_0)$ and transition equation $p(x_t|\, x_{t-1})$. The observations $\{y_t; t \in \mathbb{N}\}$, $y_t \in \mathcal{Y}$, are assumed to be conditionally independent given the process $\{x_t; t \in \mathbb{N}\}$ and of marginal distribution $p(y_t|\, x_t)$.

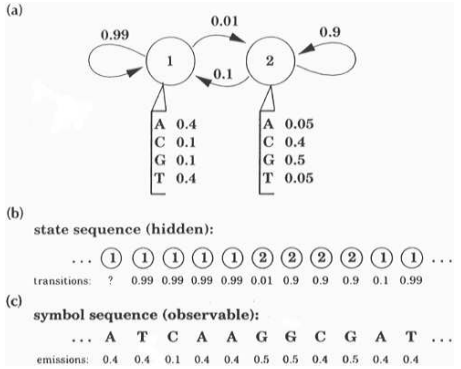
$$\begin{aligned} p(x_0) & \\ p(x_t|\, x_{t-1}) & \quad \text{for } t \geq 1 \\ p(y_t|\, x_t) & \quad \text{for } t \geq 1 \end{aligned}$$

The corresponding graphical models is as follows:

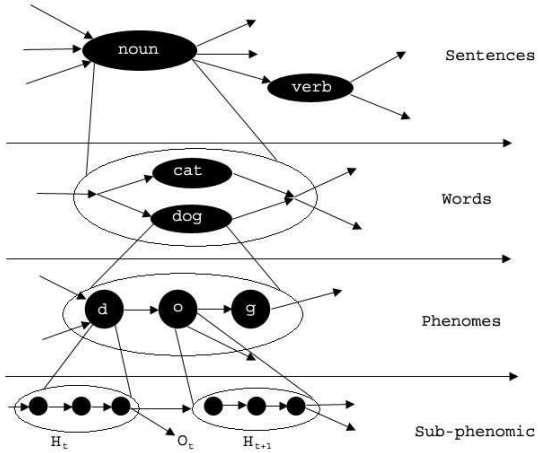


Examples include:

- **Bioinformatics**:

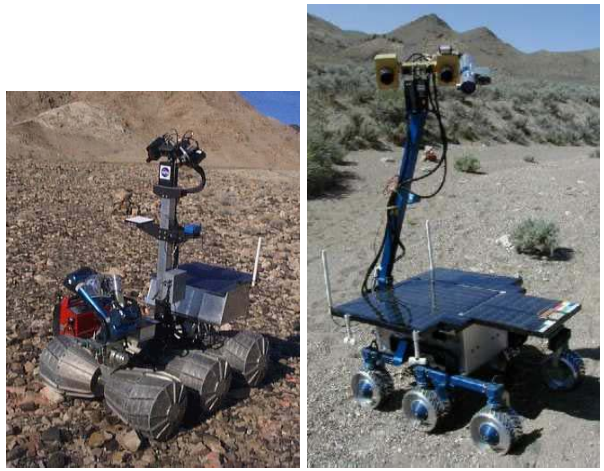


- **Speech processing**: Here y_t are acoustic vectors and x_t correspond to the phonemes/words/sentences we're trying to recognise.



- **Target tracking**: y_t are observations (typically noisy and subject to clutter) and x_t corresponds to our estimate of the position, velocity and acceleration of the entity being tracked.

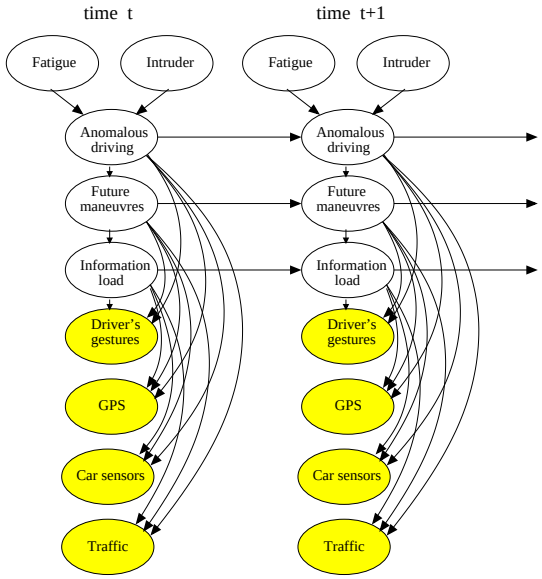
- **Self-Diagnosis in Robots**: y_t are the robot's observations and x_t its internal states. If the robot knows its internal state it can carry out diagnosis and self repair.



- **Optimal Control**

- **Localisation and Map Learning in Robots**: y_t are observations and x_t corresponds to the estimate of the robot's location and the map of the environment.

- **HCI**



- **Graphics**: y_t are observations of a character's movement (joint angles) and x_t is a compressed version of the motion. Components of x_t can be used to switch between different types of motions, e.g. running, walking, dancing.

- **Dynamic Data Compression**
- **Econometrics**: y_t could, for example, correspond to forex data and x_t to the volatility ("variance") of the market.
- **Digital Communications**
- **Ecology Models**

Bayesian Solution

The inference tasks in dynamic settings can be classified as follows:

- **Filtering**: Compute $p(x_t|y_{1:t})$.
- **Prediction**: Compute $p(x_{t+\tau}|y_{1:t})$.
- **Smoothing**: Compute $p(x_{t-\tau}|y_{1:t})$.

where τ is positive. We focus on the filtering problem. Our task is, therefore, to obtain a recurse estimator of $p(x_t|y_{1:t})$. This is done in two steps:

$$\textit{Prediction: } p\left(x_t|\, y_{1:t-1}\right) = \int p\left(x_t|\, x_{t-1}\right) p\left(x_{t-1}|\, y_{1:t-1}\right) dx_{t-1}$$

$$\textit{Updating: } p\left(x_t|\, y_{1:t}\right) = \frac{p\left(y_t|\, x_t\right) p\left(x_t|\, y_{1:t-1}\right)}{\int p\left(y_t|\, x_t\right) p\left(x_t|\, y_{1:t-1}\right) dx_t}$$

These expressions and recursions are deceptively simple as one cannot typically compute the integrals. However, if the distributions are Gaussian or discrete, we can solve the integrals. In the Gaussian case, the answer is known as the **Kalman filter**. In the discrete case, the answer gives rise to the **HMM filter**. In general, we need to do sampling (**Particle filtering**) to solve this problem.

Kalman Filtering

We consider the following dynamic state space model:

$$\begin{aligned} x_t &= Ax_{t-1} + Bw_t + Fu_t \\ y_t &= Cx_t + Dv_t + Gu_t, \end{aligned}$$

where $y_t \in \mathbb{R}^{n_y}$ denotes the observations, $x_t \in \mathbb{R}^{n_x}$ denotes the unknown Gaussian states, $u_t \in \mathcal{U}$ is a known control signal, the parameters (A, B, C, D, F, G) are known matrices and the initial mean and covariance of x_t are μ_0, Σ_0. The noise processes are *i.i.d* Gaussian: $w_t \sim \mathcal{N}(0, I)$ and $v_t \sim \mathcal{N}(0, I)$. Our model implies the continuous densities

⋆

$p(x_t|x_{t-1}) =$

$p(y_t|x_t) =$

Since the likelihood and prior are Gaussian, the posterior (filtering distribution) is also Gaussian. Its mean μ and covariance Σ can be computed using the following recursions (Kalman filter):

$$
\begin{aligned}
\mu_{t|t-1} &= A\mu_{t-1|t-1} + Fu_t \\
\Sigma_{t|t-1} &= A\Sigma_{t-1|t-1}A' + BB' \\
S_t &= C\Sigma_{t|t-1}C' + DD' \\
y_{t|t-1} &= C\mu_{t|t-1} + Gu_t \\
\mu_{t|t} &= \mu_{t|t-1} + \Sigma_{t|t-1}C^{\mathrm{T}}S_t^{-1}(y_t - y_{t|t-1}) \\
\Sigma_{t|t} &= \Sigma_{t|t-1} - \Sigma_{t|t-1}C'S_t^{-1}C\Sigma_{t|t-1}
\end{aligned}
$$

where

$$\mu_{t|t-1} \triangleq \mathbb{E}\left(x_t \middle| y_{1:t-1}\right)$$

$$\mu_{t|t} \triangleq \mathbb{E}\left(x_t \middle| y_{1:t}\right)$$

$$y_{t|t-1} \triangleq \mathbb{E}\left(y_t \middle| y_{1:t-1}\right)$$

$$\Sigma_{t|t-1} \triangleq cov\left(x_t \middle| y_{1:t-1}\right)$$

$$\Sigma_{t|t} \triangleq cov\left(x_t \middle| y_{1:t}\right)$$

$$S_t \triangleq cov\left(y_t \middle| y_{1:t-1}\right)$$

The predictive density is

$$p\left(y_t \middle| y_{1:t-1}\right) = \mathcal{N}\left(y_t; y_{t|t-1}, S_t\right)$$

Learning the Model Parameters

One can use the EM algorithm to learn the model parameters: $\theta = (A, B, C, D, F, G, \mu_0, \Sigma_0)$:

Initialisation : Start with a guess for θ^0.

E-step : Determine the expected log-likelihood density function of the complete data given the current estimate θ^{old}:

$$Q = \int \log \mathrm{p}(x_{0:T}, y_{1:T}|\theta)\mathrm{p}(x_{0:T}|y_{1:T}, \theta^{\text{old}})dx_{0:T}$$

M-step : The new θ can be found by simple differentiation of the Q function with respect to θ.

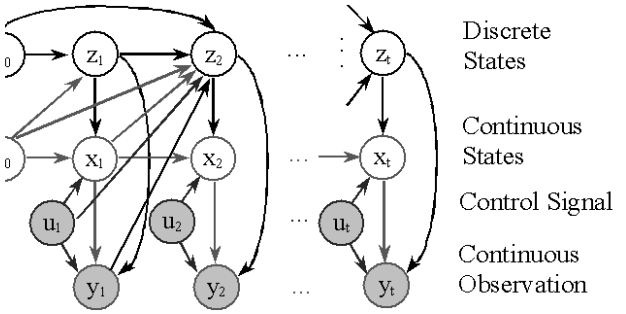
An excellent reference for this is:

`http://www.gatsby.ucl.ac.uk/~zoubin/papers/tr-96-2.ps.gz`

Zoubin's Matlab code is available at:

`http://www.gatsby.ucl.ac.uk/~zoubin/software.html`

Jump Markov Linear Models



For Markov transitions, the model is simpler:

We represent the complex nonlinear process (robot system) with a dynamic mixture of linear processes. In addition to the continuous state variables corresponding to each linear process, we have a discrete state variable that determines the linear regime of operation. In the fault diagnosis setting, each regime corresponds to a particular fault. Different regimes could, however, be used to represent levels of the fault or different internal states of the robot for the purposes of controlling it automatically. We acquire data for each regime separately. This data enables us to do off-line identification with the EM algorithm.

Once the stationary parameters have been identified, real-time Rao-Blackwellised particle filtering (RBPF) algorithms are used to estimate the continuous and discrete states of the system on-line. These estimates are used to determine the type of fault and control policies.

In mathematical terms, we adopt the following state space representation:

$$\begin{aligned} z_t &\sim P(z_t|z_{t-1}) \\ x_t &= A(z_t)x_{t-1} + B(z_t)w_t + F(z_t)u_t \\ y_t &= C(z_t)x_t + D(z_t)v_t + G(z_t)u_t, \end{aligned}$$

where,

- $y_t \in \mathbb{R}^{n_y}$ denotes the measurements.
- $x_t \in \mathbb{R}^{n_x}$ denotes the unknown continuous states.
- $u_t \in \mathcal{U}$ is a known control signal.
- $z_t \in \{1, \ldots, n_z\}$ denotes the unknown discrete states (normal operation and faulty conditions).
- The noise processes are $i.i.d$ Gaussian: $w_t \sim \mathcal{N}(0, I)$ and $v_t \sim \mathcal{N}(0, I)$.
- Typically, the (A, B, C, D, F, G) matrices are estimated with the EM algorithm for linear dynamic systems. That

is, one simply fixes z and obtains data from the physical process. With z known, it is straightforward to estimate these matrices from the measured data using the EM algorithm. We have to repeat this process for each possible value of z.

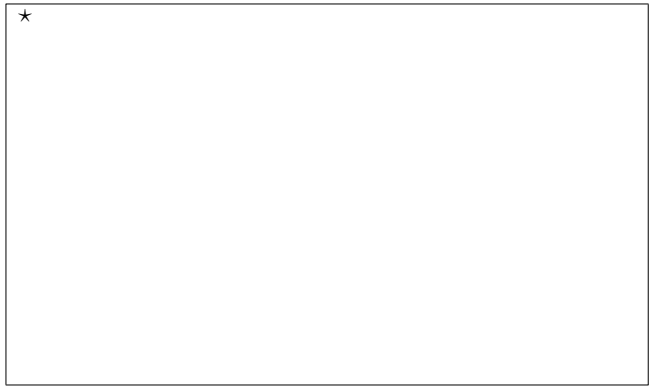
- The initial states are $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and $z_0 \sim p(z_0)$.
- The transition kernel $P(z_t|z_{t-1})$ is a discrete Markov transition matrix.

We should notice that for each realization of z_t, we have a single linear-Gaussian model. If we knew z_t, we could solve for x_t exactly using the Kalman filter algorithm. The directed acyclic graphical model representation of our model is shown below:

An good reference for this is:

`http://www.cs.ubc.ca/~nando/papers/rbpf.ps`

HMMs



In HMMs **the states are discrete** and the observations can be either continuous or discrete. Let's consider continuous observations. The discrete case is discusses in Jordan's book. The joint probability of the model is:

$$p(x, y|\theta) = p(x_0) \prod_{t=0}^{T-1} p(x_{t+1}|x_t) \prod_{t=1}^{T} p(y_t|x_t)$$

We adopt a notation in which state variables can be used as indices. When x_t takes on its i^{th} value and x_{t+1} takes

on its j^{th} value, we let $a_{d_t,d_{t+1}}$ denote the $(i,j)^{th}$ entry of the transition matrix $p(x_{t+1}|x_t)$. That is, $p(x_{t+1} = j|x_t = i) = a_{i,j}$. Formally, this interpretation is achieved via the following definition:

$$p(x_{t+1}|x_t) = \prod_{i,j=1}^{n_x} [a_{ij}]^{x_t^i x_{t+1}^j}$$

That is, we use a multinomial to describe the state transitions. Similarly

$$p(x_0) = \prod_i^{n_x} \pi_i^{x_0^i}$$

The likelihood (a.k.a. obsevation model, emmission model) is Gaussian

$$p(y_t|x_t = i) = \mathcal{N}(\mu_i, \Sigma_i)$$

These expressions allow us to write an expression for $p(x, y|\theta)$. In the E step, we compute the expectation of the complete log likelihood

$$\mathbf{Q}^{\text{ML}} = \mathbb{E}_{p(x|y,\theta^{(old)})}[\log\, p(x, y|\theta)]$$

That is, in the E step we need to be able to compute $p(x|y, \theta^{(old)})$. Differentiating $\mathbf{Q}^{\text{ML}}$, give us the update equations for the parameters: a, π, μ, Σ.

E: Forward-Backward Algorithm

we have:

$$\begin{aligned} p(x_t|y) &= \frac{p(y|x_t)p(x_t)}{p(y)} \\ &= \frac{p(y_1, ..., y_t|x_t)p(y_{t+1}, ..., y_T|x_t)p(x_t)}{p(y)} \\ &= \frac{\alpha(x_t)\beta(x_t)}{p(y)} \end{aligned}$$

where

$$p(y) = \sum_{x_t} \alpha(x_t)\beta(x_t)$$

Alpha and Beta are the messages propagated in the graph. We also define:

$$\gamma(x_t) = \frac{\alpha(x_t)\beta(x_t)}{p(y)}$$

That is, $\gamma(x_t) = p(x_t|y)$ is the smoothing distribution.

In the Forward-Backward algorithm, the forward pass allows us to compute α recursively and the backward pass allows us to compute β_t. Combining these estimates, we obtain $p(x_t|y)$ and $p(y)$.

In the forward direction, we have:

$$\alpha(x_0) = p(x_0)$$

and

$$\begin{aligned}
\alpha(x_{t+1}) &= p(y_1, ..., y_{t+1}, x_{t+1}) \\
&= p(y_1, ..., y_{t+1}|x_{t+1})p(x_{t+1}) \\
&= p(y_1, ..., y_t|x_{t+1})p(y_{t+1}|x_{t+1})p(x_{t+1}) \\
&= \sum_{x_t} p(y_1, ...y_t, x_t, x_{t+1})p(y_{t+1}|x_{t+1}) \\
&= \sum_{x_t} p(y_1, ...y_t|x_t)p(x_{t+1}|x_t)p(x_t)p(y_{t+1}|x_{t+1}) \\
&= \sum_{x_t} \alpha(x_t)p(x_{t+1}|x_t)p(y_{t+1}|x_{t+1})
\end{aligned}$$

Note that the algorithm proceeds "forward" in time.

For the beta variables we can obtain a "backward" recursion:

$$\begin{aligned}
\beta(x_t) &= p(y_{t+1}, ..., y_T|x_t) \\
&= \sum_{x_{t+1}} p(y_{t+1}, ..., y_T|x_{t+1}, x_t)p(x_{t+1}|x_t) \\
&= \sum_{x_{t+1}} p(y_{t+2}, ..., y_T|x_{t+1})p(x_{t+1}|x_t)p(y_{t+1}|x_{t+1}) \\
&= \sum_{x_{t+1}} \beta(x_{t+1})p(x_{t+1}|x_t)p(y_{t+1}|x_{t+1})
\end{aligned}$$

The beta recursion is a backward recursion, i.e. we start at the final time step T and proceed back-wards to the initial time step. For the initialization of the beta recursion , we define $\beta(x_T)$ to be a vector of ones.

Then we have

$$\gamma(x_t) = \frac{\alpha(x_t)\beta(x_t)}{\sum_{x_t} \alpha(x_t)\beta(x_t)}$$

For the M step we will also need the clique probability

$$\xi(x_t, x_{t+1}) = p(x_t, x_{t+1}|z)$$

which is given by:

$$\xi(x_t, x_{t+1}) \;=\; \frac{\alpha(x_t)p(y_{t+1}|x_{x+1})\gamma(x_{t+1})p(x_{t+1}|x_t)}{\alpha(x_{t+1})}$$

M Step

The complete log-likelihood is given by:

$$\begin{aligned}
\log p(x,y) \;&=\; \log p(x_0)\prod_{t=0}^{T-1} p(x_t|x_{t-1})\prod_{t=1}^{T} p(y_t|x_t) \\
&= \log\left\{\prod_i \pi_i^{x_0^i}\prod_{t=0}^{T-1}\prod_{i=1,j=1}^{n_x}[a_{ij}]^{x_t^i x_{t+1}^j}\prod_{t=1}^{T}\prod_i^{n_x}\mathcal{N}(\mu_i,\Sigma_i)^{x_t^i}\right\} \\
&= \sum_i^{n_x} x_0^i \log\pi_i + \sum_{t=0}^{T-1}\sum_{i=1,j=1}^{n_x} x_t^i x_{t+1}^j \log a_{ij} + \sum_{t=1}^{T}\sum_i^{n_x} x_t^i \log\mathcal{N}(\mu_i,\Sigma_i)
\end{aligned}$$

Taking expectations, we obtain the Q function:

$$\begin{aligned}
Q \;&=\; \sum_i^{n_x} p(x_0 = i|y,\theta^{old})\log\pi_i \\
&\quad + \sum_{t=0}^{T-1}\sum_{i=1,j=1}^{n_x} p(x_t = i, x_{t+1} = j|y,\theta^{old})\log a_{ij} \\
&\quad + \sum_{t=1}^{T}\sum_i^{n_x} p(x_t = i|y,\theta^{old})\log\mathcal{N}(\mu_i,\Sigma_i)
\end{aligned}$$

Now we simply take derivative with respect to each individual parameter to obtain the M step updates:

$$\widehat{a}_{ij} = \frac{\sum_{t=0}^{T-1}\xi(x_t = i, x_{t+1} = j)}{\sum_{t=0}^{T-1}\gamma(x_t = i)}$$

$$\widehat{\pi}_i = \gamma(x_0 = i)$$

$$\begin{aligned}
\widehat{\mu}_i \;&=\; \frac{\sum_{t=1}^{T}\gamma(x_t = i)y_t}{\sum_{t=0}^{T}\gamma(x_t = i)} \\
\widehat{\Sigma}_i \;&=\; \frac{\sum_{t=1}^{T}\gamma(x_t = i)(y_t - \mu_i)(y_t - \mu_i)'}{\sum_{t=0}^{T}\gamma(x_t = i)}
\end{aligned}$$

Particle Filtering

We begin with a review of the sequential Monte Carlo method for approximating probability distributions and carrying out integration in high-dimensional spaces. Assume we have a distribution $\pi(\mathbf{x}_{1:n})$ over a sequence of random vectors, $\mathbf{x}_{1:n} \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, which is only known up to a normalization constant:

$$\pi(\mathbf{x}_{1:n}) = Z_n^{-1} f(\mathbf{x}_{1:n})$$

where $Z_n \triangleq \int f(\mathbf{x}_{1:n}) d\mathbf{x}_{1:n}$ is the partition function. We are often interested in computing this partition function and other expectations, such as

$$I(g(\mathbf{x}_{1:n})) = \int g(\mathbf{x}_{1:n}) \pi(\mathbf{x}_{1:n}) d\mathbf{x}_{1:n}$$

If we had a set of samples $\left\{\mathbf{x}_{1:n}^{(i)}\right\}_{i=1}^{N}$ from π, we could approximate this integral with the following Monte Carlo esti-

mator

$$\widehat{\pi}(d\mathbf{x}_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_{1:n}^{(i)}}(d\mathbf{x}_{1:n})$$

and consequently approximate the expectations of interest with

$$\widehat{I}(g(\mathbf{x}_{1:n})) = \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{x}_{1:n}^{(i)})$$

It is typically hard to sample from π directly. Instead, we can sample from a proposal distribution q and weight the samples according to

$$w_n = \frac{f(\mathbf{x}_{1:n})}{q(\mathbf{x}_n|\mathbf{x}_{1:n-1}) f(\mathbf{x}_{1:n-1})} w_{n-1}$$

⋆ Proof:

The set of weighted samples from q allows us to construct the following estimate of the partition function

$$\widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n^{(i)}$$

⋆ Proof:

Given a set of N particles (samples) $\mathbf{x}_{1:n-1}^{(i)}$, we obtain a set of particles $\mathbf{x}_n^{(i)}$ by sampling from $q(\mathbf{x}_n|\mathbf{x}_{1:n-1}^{(i)})$ and applying the recursive importance weights. To overcome slow drift in the particle population, a resampling (selection) step chooses the fittest particles.

Sequential importance sampling step

- For $i = 1, ..., N$, sample from the proposal

$$\widetilde{\mathbf{x}}_n^{(i)} \sim q(\mathbf{x}_n|\mathbf{x}_{1:n-1}^{(i)})$$

  and set $\widetilde{\mathbf{x}}_{1:n}^{(i)} \triangleq \left(\widetilde{\mathbf{x}}_n^{(i)}, \mathbf{x}_{1:n-1}^{(i)}\right)$.

- For $i = 1, ..., N$, evaluate the importance weights

$$w_n^{(i)} = \frac{f(\widetilde{\mathbf{x}}_{1:n}^{(i)})}{q(\widetilde{\mathbf{x}}_n^{(i)}|\widetilde{\mathbf{x}}_{1:n-1}^{(i)}) f(\widetilde{\mathbf{x}}_{1:n-1}^{(i)})} w_{n-1}^{(i)}$$

- Normalise the importance weights

$$\widetilde{w}_n^{(i)} = \frac{w_n^{(i)}}{\sum_j w_n^{(j)}}$$

Selection step

- Resample the discrete weighted measure $\left\{\widetilde{\mathbf{x}}_{0:n}^{(i)}, \widetilde{w}_n^{(i)}\right\}_{i=1}^{N}$ to obtain an unweighted measure $\left\{\mathbf{x}_{0:n}^{(i)}, \frac{1}{N}\right\}_{i=1}^{N}$ of N new particles.

In Bayesian estimation, the target distribution is the posterior distribution

$$\pi(\mathbf{x}_{1:n}) = p(\mathbf{x}_{1:n}|\mathbf{y}_{1:n}) = Z_n^{-1} f(\mathbf{x}_{1:n})$$

where $Z_n = p(\mathbf{y}_{1:n})$ and

$$f(\mathbf{x}_{1:n}) = p(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \prod_{k=1}^{n} p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1})$$

Hence

$$w_n = \frac{p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1})}{q(\mathbf{x}_n|\mathbf{x}_{1:n-1}, \mathbf{y}_{1:n})} w_{n-1}$$

⋆ Proof:

In particular, if we choose the proposal to be the transition prior $p(\mathbf{x}_n|\mathbf{x}_{n-1})$, then the importance weights are simply the likelihood funtions $p(\mathbf{y}_n|\mathbf{x}_n)$.

⋆