COMPGI18
PROBABILISTIC & UNSUPERVISED LEARNING

Introduction & Foundations

- Unsupervised: observe data, find pattern & underlying structure
- Supervised: observe input/output pairs, learn to predict for new input
- Reinforcement: find policy for action choice to maximise reward

- Probabilistic approach:

  $p(x|\theta)$ or $p(y|x,\theta)$

  - generative model / likelihood
  - Beliefs: isomorphic to probabilities
  - Basic rules of probability:

    $P(x) \geq 0$

    $\sum_{x \in X} P(x) = 1$

    $P(x) = \sum_y P(x,y)$

    $P(x|y) = P(x,y) / P(y)$

    $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

Dutch book theorem:
- if I bet according to my beliefs (ie if $b(x=T) = 0.9$, accept 1:9 bet)
- exist set of bets where I am guaranteed to lose money if my beliefs are not consistent

- Bayesian learning:
  - Data $D = \{x_1, \ldots, x_n\}$
  - Models $M_1, M_2, \ldots$
  - Parameters Θ_i (per model)

Parameter learning $\longrightarrow$ $P(\Theta_i | D, M_j) = \frac{P(D|\Theta_i, M_j) P(\Theta_i | M_j)}{P(D|M_j)}$

Model selection $\longrightarrow$ $P(M_i | D) = \frac{P(D|M_i) P(M_i)}{P(D)}$ $\quad \left(P(D|M_i) = \int d\theta \, p(D|M_i,\theta) p(\theta|M_i) \right)$

- Conjugate priors:
  - for a given likelihood, conjugate prior give posterior in same family as prior
  - Exponential family likelihood:

$$P(x|\theta) = g(\theta) f(x) e^{\phi(\theta)^T T(x)} \quad \left(\begin{array}{l} \phi(\theta) = \text{natural parameters} \\ T(x) = \text{sufficient statistic} \end{array} \right)$$

$$\Rightarrow p(\theta) \propto g(\theta)^{\nu} e^{\phi(\theta)^T \tau}$$

τ = pseudo observations

ν = 'scale' of prior

$$\left(\Rightarrow p(\theta | \underline{x}) \propto g(\theta)^{\nu + n} e^{\phi(\theta)^T (\tau + \sum T(x_i))} \right)$$

- ML learning for a Gaussian:

$$p(\underline{x}|\underline{\mu},\underline{\underline{\Sigma}})=N(\underline{\mu},\underline{\underline{\Sigma}})$$

log likelihood $\longrightarrow$ $L=\log\prod_i p(\underline{x}_i|\underline{\mu},\underline{\underline{\Sigma}})=\sum\log p(\underline{x}_i|\underline{\mu},\underline{\underline{\Sigma}})$

$$\frac{\partial L}{\partial\underline{\mu}}=0\Rightarrow\hat{\underline{\mu}}=\frac{1}{N}\sum\underline{x}_i\,,\quad\frac{\partial L}{\partial\underline{\underline{\Sigma}}^{-1}}=0\Rightarrow\hat{\underline{\underline{\Sigma}}}=\frac{1}{N}\sum(\underline{x}_i-\hat{\underline{\mu}})(\underline{x}_i-\hat{\underline{\mu}})^T$$

- Multivariate linear regression:

$$p(\underline{y}|\underline{x},\underline{\underline{W}},\underline{\underline{\Sigma}}_y)=N(\underline{\underline{W}}\underline{x},\underline{\underline{\Sigma}}_y)$$

$$\frac{\partial L}{\partial\underline{\underline{W}}}\Rightarrow\hat{\underline{\underline{W}}}_{ML}=\left(\sum_i\underline{y}_i\underline{x}_i^T\right)\left(\sum_i\underline{x}_i\underline{x}_i^T\right)^{-1}$$

- MAP learning:

$$p(\underline{w})=N(\underline{0},\underline{\underline{A}}^{-1}),\quad y_i\text{ scalar},\quad y_i|\underline{x},\underline{w},\sigma_y\sim N(\underline{w}^T\underline{x},\sigma_y^2)$$

$$\Rightarrow\log p(\underline{w}|D,\underline{\underline{A}},\sigma_y)=L+\log p(\underline{w}|\underline{\underline{A}})$$
$$=\log N(\underline{w}_{MAP},\underline{\underline{\Sigma}}_w)$$

$$\left[\underline{w}_{MAP}=\underline{\underline{\Sigma}}_w^{-1}\frac{1}{\sigma_y^2}\sum y_i\underline{x}_i\,,\quad\underline{\underline{\Sigma}}_w=\left(\underline{\underline{A}}+\frac{1}{\sigma_y^2}\sum\underline{x}_i\underline{x}_i^T\right)\right]$$

- Problems with multivariate Gaussian model:
  - no higher order structure
  - predicts very few outliers; not robust
  - $\frac{D(D+1)}{2}$ parameters

2: Latent Variable Models

- Explain correlations in $\underline{x}$ by assuming dependence on latent variables $\underline{y}$
  - can reduce parameters needed, capture an underlying generative process
  - $p(\underline{y})$, $p(\underline{x}|\underline{y})$, maybe $p(\underline{x},\underline{y})$ in exponential family ($p(\underline{x})$ rarely is)

- Probabilistic PCA:

  $D = \{x_1, x_2, \dots, x_N\}$, $\underline{x}_i \in \mathbb{R}^D$ $\quad (K<D)$

  $Y = \{y_1, \dots, y_N\}$, $\underline{y}_i \in \mathbb{R}^K$

  noise in any direction

  - Model:

  $$\underline{x}_i = \underline{\Lambda}\,\underline{y}_i + \underline{e}_i, \qquad \underline{y} \sim N(\underline{0}, \underline{\underline{I}}), \quad \underline{e} \sim N(\underline{0}, \psi \underline{\underline{I}})$$

  $$\Rightarrow \underline{x}|\underline{y} \sim N(\underline{\Lambda}\,\underline{y}, \psi\underline{\underline{I}}) \Rightarrow \underline{x} \sim N(\underline{0}, \underline{\Lambda}\,\underline{\Lambda}^T + \psi\underline{\underline{I}})$$

  $DK+1$ free parameters (vs $\frac{D(D+1)}{2}$ if $\underline{x} \sim N(\underline{0}, \underline{\underline{\Sigma}})$)

  (more interpretable, but harder to do inference)

noise $\perp$ to projection →

- PCA:
  - as $\psi \to 0$, can only capture K dimensions of variance

    $\underline{\Lambda}$ columns = eigenvectors of $\underline{\underline{S}}$, ordered by eigenvalue

    ($\underline{\underline{S}} = \underline{\underline{U}}\,\underline{\underline{W}}\,\underline{\underline{U}}^T$, $\underline{\underline{U}}$ columns = eigenvectors,

    $\underline{\underline{W}} = \mathrm{diag}(\lambda_1, \dots, \lambda_D)$)

- ML Learning
  - $l = \log \prod_{N} N(\underline{0}, p(\underline{x}|\underline{\Lambda}, \psi))$

  $$\frac{\partial l}{\partial \underline{\Lambda}} \Rightarrow \underline{\Lambda} = 0$$

  or, $\underline{\Lambda}\,\underline{\Lambda}^T + \psi\underline{\underline{I}} = \underline{\underline{S}}$

  or, $\underline{\Lambda} = \underline{\underline{U}}\,\underline{\underline{L}}\,\underline{\underline{V}}^T$, and $\underline{\underline{S}}\,\underline{\underline{U}} = \underline{\underline{U}}(\underline{\underline{L}}^2 + \psi\underline{\underline{I}})$

- Projection:

  $\tilde{\underline{x}}_n = \underline{\Lambda}\,\bar{\underline{y}}_n$, where $\bar{\underline{y}}_n = E(\underline{y}_n|\underline{x}_n)$

  - find $E(\underline{y}_n|\underline{x}_n)$:

    write $p(\underline{y}_n, \underline{x}_n) = p(\underline{y}_n)p(\underline{x}_n|\underline{y}_n)$, consider $\underline{x}_n$ fixed

    $\Rightarrow \underline{y}_n|\underline{x}_n \sim N(\beta \underline{x}_n, \underline{\underline{I}} - \beta\underline{\Lambda})$, $\beta = \underline{\underline{\Sigma}}\,\underline{\Lambda}^T \psi^{-1}$

- Factor Analysis:
  - don't assume equal noise in all dimensions; $\underline{e} \sim N(\underline{0}, \underline{\underline{\Psi}})$
  - $DK + D - \frac{K(K-1)}{2}$ parameters (not identifiable if $> \frac{D(D+1)}{2}$)
  - no closed form solution for $\underline{x} \sim N(\underline{0}, \underline{\Lambda}\underline{\Lambda}^T + \underline{\Psi})$
    $\Rightarrow$ need to use gradient ascent on log likelihood (or EM)
  - projection: same as for PPCA, except $\underline{\Psi}$ now (diagonal) matrix

- PCA & PPCA rotationally invariant, FA not
- FA measurement scale invariant, PCA/PPCA not
- FA & PPCA probabilistic model, PCA not
- all can only model very restricted class of densities (too many assumptions)

- Mixture Models:

$$s_i \overset{iid}{\sim} \text{Discrete}(\underline{\pi}), \quad \underline{x}_i | s_i \sim P_{s_i}[\Theta_{s_i}]$$

$$\Rightarrow p(\underline{x}_i) = \sum_m \pi_m p_m(\underline{x}_i | \Theta_m)$$ (convex combination of component densities)

  - Mixture of Gaussians:

$$\underline{x}_i | s_i \sim N(\underline{\mu}_{s_i}, \underline{\underline{\Sigma}}_{s_i})$$

$$p(D | \{\underline{\mu}_m\}, \{\underline{\underline{\Sigma}}_m\}, \underline{\pi}) = \prod_{i=1}^{n} \sum_{m=1}^{k} \pi_m N(\underline{x}_i | \underline{\mu}_m, \underline{\underline{\Sigma}}_m)$$

  - ML: $$L = \sum_{i=1}^{n} \log \sum_{m=1}^{k} \pi_m p_m(\underline{x}_i | \Theta_m)$$

(For MoG, $\Theta_m = \{\underline{\mu}_m, \underline{\underline{\Sigma}}_m\}$) $$\frac{\partial L}{\partial \Theta_m} \overset{*}{=} \sum_{i=1}^{n} r_{im} \frac{\partial}{\partial \Theta_m} \log p_m(\underline{x}_i | \Theta_i)$$ $$\left(r_{im} = P(s_i = m | \underline{x}_i) = \frac{p_m(\underline{x}_i)\pi_m}{\sum_k p_k(\underline{x}_i)\pi_k} \right)$$

$$\frac{\partial L}{\partial \pi_m} = \sum_{i=1}^{n} \frac{r_{im}}{\pi_m}$$

(* using $\frac{\partial p}{\partial \theta} = p \times \frac{\partial \log p}{\partial \theta}$)

  - K-means:

$$\pi_m = 1/m, \ \underline{\underline{\Sigma}}_m = \sigma^2 \underline{\underline{I}}, \ \sigma^2 \to 0$$

$\Rightarrow$ MoG approaches K-means $\left[r_{im} \to \delta(m, \underset{i}{\operatorname{argmin}} \|\underline{x}_i - \underline{\mu}_i\|^2) \right]$

3: Expectation Maximisation

(* Jenson's inequality)

- Maximisation hard when y is latent
- Idea:
  - find expectation of y, then update Θ w.r.t. this, & alternate
- E-step:
  - find expected values for latent variables
- M-step:
  - maximise likelihood as if latents were not hidden

- Proof we ~~always~~ never decrease likelihood:

$$\ell(\Theta) = \log p(X|\Theta) = \log \int p(Y,X|\Theta)\,dY$$

$$= \log \int q(Y) \frac{p(Y,X|\Theta)}{q(Y)}\,dY \overset{*}{\geq} \int q(Y) \log \frac{p(Y,X|\Theta)}{q(Y)}\,dY \triangleq \mathcal{F}(q,\Theta)$$

Free energy $\longrightarrow$ $\mathcal{F}(q,\Theta) = \int q(Y)\log p(Y,X|\Theta)\,dY - \int q(Y)\log q(Y)$

$$\mathcal{F}(q,\Theta) = E_{q(Y)}[\log p(Y,X|\Theta)] + H(q) \leftarrow \text{entropy}$$

- E-step:

$$q^{(k)}(Y) = \underset{q(Y)}{\operatorname{argmax}}\, \mathcal{F}(q(Y),\Theta^{(k-1)}) = p(Y|X,\Theta^{(k-1)})$$

- M-step:

$$\Theta^{(k)} = \underset{\Theta}{\operatorname{argmax}}\, \mathcal{F}(q^{(k)}(Y),\Theta) = \underset{\Theta}{\operatorname{argmax}}\, E_{q^{(k)}(Y)}[\log p(Y,X|\Theta)]$$

$$\mathcal{F}(q,\Theta) = \int q(Y)\log p(X|\Theta)\,dY + \int q(Y)\log\frac{p(Y|X,\Theta)}{q(Y)}\,dY$$

$$\mathcal{F}(q,\Theta) = \ell(\Theta) - KL[q(Y)\,\|\,p(Y|X,\Theta)]$$

$\Rightarrow$ E-step sets $q^{(k)}(Y) = p(Y|X,\Theta^{(k-1)})$ [minimize KL-divergence]

$$\ell(\Theta^{(k-1)}) \underset{\text{E-step}}{=} \mathcal{F}(q^{(k)},\Theta^{(k-1)}) \underset{\text{M-step}}{\leq} \mathcal{F}(q^{(k)},\Theta^{(k)}) \underset{\text{Jensen}}{\leq} \ell(\Theta^{(k)})$$

- EM for MoG:
  - E-step:

$$q(s_i) = p(s_i | x_i, \theta) \propto \pi_{s_i} \times N(\mu_{s_i}, \sigma^2_{s_i})$$

  - M-step:

$$\theta = \arg\max_\theta E_{q(s)}[\log p(x, s | \theta)]$$

$$E = \sum q(s) \log[p(s|\theta) p(x|s,\theta)]$$

$$E(\theta) = \sum_{i,m} r_{im} \left[\log \pi_m - \log \sigma_m - \frac{1}{2\sigma_m^2}(x_i - \mu_m)^2\right]$$

(find μ_m, σ_m, π_m by differentiating E)

- EM for FA:
  - E-step:

$$q_n(y_n) = p(y_n | x_n, \theta) \propto p(y_n, x_n | \theta) = p(y_n|\theta) p(x_n | y_n, \theta)$$

$$= N(\beta x_n, I - \beta \Lambda), \quad \beta = \Sigma \Lambda^T \Psi^{-1}$$

  - M-step:

$$\theta = \arg\max_\theta \sum_n E_{q_n(y_n)}[\log p(x_n, y_n | \theta)]$$

$$\log p(x_n, y_n | \theta) = \log p(y_n | \theta) + \log p(x_n | y_n, \theta)$$

do maths, take expectation by replacing y_n w/ μ_n, $y_n y_n^T$ w/ $\mu_n \mu_n^T + \Sigma$

(as $\Sigma \to 0$ these tend to equations for ML linear regression)

$$\frac{\partial F}{\partial \Lambda} = 0 \Rightarrow \hat{\Lambda} = \left(\sum_n x_n \mu_n^T\right)\left(N\Sigma + \sum_n \mu_n \mu_n^T\right)$$

$$\frac{\partial F}{\partial \Psi^{-1}} = 0 \Rightarrow \hat{\Psi} = \hat{\Lambda} \Sigma \hat{\Lambda} + \frac{1}{N}\sum_n (x_n - \Lambda \mu_n)(x_n - \Lambda \mu_n)^T$$

- EM for Exponential families:

$p(y, x | \theta)$ has exp-family form

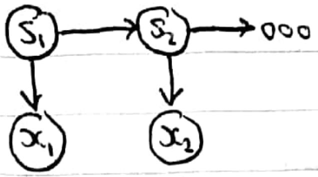
$$F(q, \theta) = \int q(y) \log p(y, x | \theta)\, dy + H(q)$$
$$= \theta^T E_{q(y)}[T(y, x)] - \log Z(\theta) + C$$

$\Rightarrow$ E-step: compute expected sufficient stats under q

M-step: $\frac{\partial F}{\partial \theta} = E_{q(y)}[T(y, x)] - E[T(y, x | \theta)] = 0$

Scanned by CamScanner

4: Latent Chain Models



- Hidden Markov Models:
  - $S_t \in \{1, \ldots, K\}$, x_t discrete or continuous

    $\phi_{ij} = P(S_{t+1} = j \mid S_t = i)$

    $\pi_j = P(S_1 = j)$

    $A_{jk} = P(x_t = k \mid S_t = j)$ [or $A_j(\underline{x}) = P(\underline{x}_t = \underline{x} \mid S_t = j)$ for continuous $\underline{x}_t$]

    $$p(x_{1:T}, S_{1:T}) = \pi_{S_1} A_{S_1 x_1} \prod_{t=2}^{T} \phi_{S_{t-1} S_t} A_{S_t x_t}$$

  - Learning: use EM
    - E-step: find $p(S_t \mid x_{1:T})$

      $$p(S_t = i \mid x_{1:T}) = \frac{P(S_t = i \mid x_{1:t}) P(x_{t+1:T} \mid S_t = i)}{P(x_{1:T})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

      - Forward:

        $\alpha_1(i) = \pi_i A_i(x_1)$, $\alpha_{t+1}(i) = \left(\sum_j \alpha_t(j)\phi_{ji}\right) A_i(x_{t+1})$

      - Backward:

        $\beta_t(i) = \sum_j \phi_{ij} A_j(x_{t+1}) \beta_{t+1}(j)$

      in practise, normalise as we go

      - $q(S_{1:T}) = p(S_{1:T} \mid x_{1:T}, \Theta)$

    - M-step:

      $\hat{\pi}_i = q(S_1 = i)$

      $$\hat{\phi}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i \to j)}{\sum_{t=1}^{T-1} q(S_t = i)} \qquad \varepsilon_t(i \to j) = P(S_t = i, S_{t+1} = j \mid x_{1:T}) = \frac{\alpha_t(i)\phi_{ij} A_j(x_{t+1}) \beta_{t+1}(j)}{P(x_{1:T})}$$

      $$\hat{A}_{ik} = \frac{\sum_{t: x_t = k} q(S_t = i)}{\sum_{t=1}^{T} q(S_t = i)}$$

$$y_1 \sim N(\mu_0, Q_0)$$
$$y_t | y_{t-1} \sim N(Ay_{t-1}, Q)$$
$$x_t | y_t \sim N(Cy_t, R)$$

- Linear Gaussian State-Space Models:

$$\underline{x}_t = C\underline{y}_t + \underline{v}_t$$
$$\underline{y}_t = A\underline{y}_{t-1} + \underline{w}_t$$

$\underline{v}_t, \underline{w}_t$ 0-mean uncorrelated Gaussian noise

$\underline{y}_1$ Gaussian,

$p(\underline{x}_{1:T}, \underline{y}_{1:T})$ multivariate Gaussian

- Learning:
  - E-step:

Kalman smoothing $\rightarrow$ $p(\underline{y}_t | \underline{x}_{1:T}) = \int p(\underline{y}_t | \underline{y}_{t+1}, \underline{x}_{1:t})\, p(\underline{y}_{t+1} | \underline{x}_{1:T})\, d\underline{y}_{t+1}$

- backwards recursion

  - M-step:

$$C_{new} = \arg\max_C \, E_q\left[\sum_t \ln p(\underline{x}_t | \underline{y}_t)\right] = \left(\sum_t \underline{x}_t E[\underline{y}_t]\right)\left(\sum_t E[\underline{y}_t \underline{y}\right.$$

$$A_{new} = \arg\max_A \, E_q\left[\sum_t \ln p(\underline{y}_{t+1} | \underline{y}_t)\right] = \left(\sum_t E[\underline{y}_{t+1} \underline{y}_t^T]\right)\left(\sum_t E[\underline{y}_t \underline{y}_t^T]\right)$$

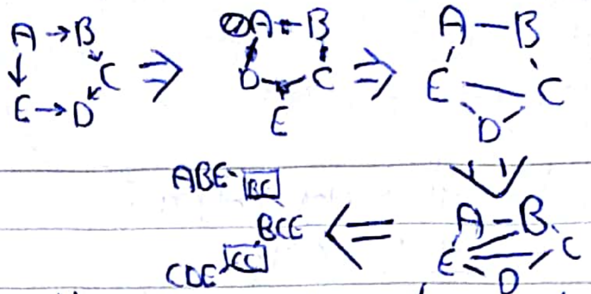
- Online:
  - $\Lambda = \sum_{t=1}^{T} \ln p(\underline{x}_t | \underline{x}_{1:t-1}) = \sum_{t=1}^{T} \Lambda_t$ ($p(x_t | x_{1:t-1})$ determined by Kalman filtering)
  - use gradient rules to update A, C, Q, R as we go (learning rate = expectation about non-stationarity)

- Slow Feature Analysis:
  - $A \rightarrow I$

    $Q = I - AA^T \rightarrow 0$

    $R \rightarrow 0$

    $\Rightarrow$ given series $x_{1:T}$, find C such that $y_{1:T}$ changes as slowly as possible



5: Graphical Models

- Junction Tree Algorithm:
  - DAG $\xrightarrow{①}$ Factor Graph $\xrightarrow{②}$ Undirected Graph $\xrightarrow{③}$ Chordal Graph $\xrightarrow{④}$ Junction Tree $\xrightarrow{⑤}$ Message Passing

(every step removes conditional independencies, ⇒ ↑ family of possible distributions)

moralising DAG does 1&2 in 1 step

1: Conditional distributions in DAG → Factors in factor graph
2: Replace factors by undirected clique
3: Add edges to graph so every loop of size ≥4 has 1+ chords

- Variable elimination:
  - eliminate variables in turn, connecting all nodes that variable is connected to
  - order of removal heuristic:
    - Choose variable ~~with~~ whose removal would add fewest new edges each step

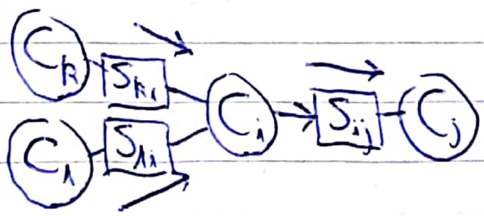
heuristic can significantly impact cost of future message passing

4: Each node in junction tree is a clique, edges are intersections between cliques

- Running intersection property:
  - if 2 cliques contain X, all cliques & separators between them do so too
- First, find graph where each node is a maximal clique
- Then, find maximal spanning tree (edge weight = no. variables in separator)

5:

most expensive stage; $O(\sum_i |X_{c_i}|)$



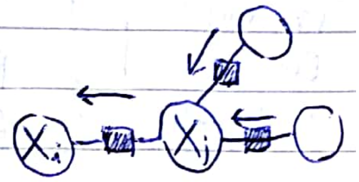
$$M_{i\to j}(X_{S_{ij}}) = \sum_{X_{C_i} \setminus S_{ij}} f_i(X_{C_i}) \prod_{k \in ne(i) \setminus j} M_{k\to i}(X_{S_{ki}})$$

$$P(X_{C_i}) = f_i(C_i) \prod_{k \in ne(i)} M_{k\to i}(X_{S_{ki}})$$

$$P(X_{S_{ij}}) = M_{i\to j}(X_{S_{ij}}) M_{j\to i}(X_{S_{ij}})$$

- Conditional independence:
  - $A \perp\!\!\!\perp B \mid X$ iff
    - moralise graph of A, B, X & ancestors
    - if X separates A & B, then true (else false)

- Belief propagation in trees:
  - undirected graph with no cycles
  - factors; $p(\mathcal{X}) \propto \prod_{ij} f_{ij}(X_i, X_j)$

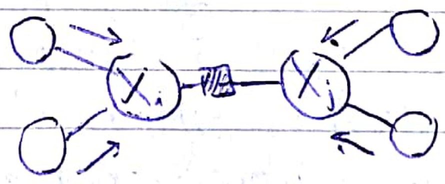


(Start from any leaf) $M_{j \to i}(X_i) = \sum_{X_j} f_{ij}(X_i, X_j) \prod_{X_k \in ne(X_j) \setminus X_i} M_{k \to j}(X_j)$

- Inference:

  $p(X_i) =$

  $p(X_i, X_j) = f_{ij}(X_i, X_j) \prod_{X_k \in ne(X_j) \setminus X_i} M_{k \to j}(X_j) \prod_{X_k \in ne(X_i) \setminus X_j} M_{k \to i}(X_i)$



If leaf node is observed, messages are conditioned, not marginalised;

$$M_{a \to i}(X_i) = f_{ai}(X_a = a, X_i)$$

6: Model Selection & GPs

- Model Selection:
  - cannot use ML/MAP; more complex models always have higher likelihood
  - can compare nested models, stopping when adding complexity doesn't improve some score
  - or cross-validation
  - or Bayesian model selection:

prob of random parameter values generating data

$$P(m|D) = \frac{p(D|m)p(m)}{p(D)}, \quad p(D|m) = \int p(D|\theta_m, m)\, p(\theta_m|m)\, d\theta_m$$

(can ignore) $p(D)$

- gives Bayesian Occams razor;
  - too simple models unlikely to generate data
  - too complex can generate too many datasets, so unlikely to generate this specific one
- Computing $p(D|m)$:
  - if likelihood $p(D|\theta_m, m)$ in exp fam, & using conjugate prior, then joint, $p(D, \theta_m|m)$ in exp family too thus can be tractable, but in general not
- Approximations:
  - Laplace: approximate posterior by Gaussian centred at $\hat{\theta}_{MAP}$
  - Bayesian info criterion: take Laplace & let $N \to \infty$
    $\log p(D|m) \approx \log p(D|\theta^*_m, m) - \frac{d}{2}\log N$
    + quick & easy to compute
    + can use θ_{ML} (as $N \to \infty$)
- Hyperparameters:
  - models may be continuously parameterised by hyperparameters η, need $p(D|\eta)$
  - can try * exact evidence, approximated, or sample η from $p(\eta|D) \propto p(D|\eta)p(\eta)$ (by MCMC)

*[gradient ascent in]

$\Rightarrow$ non-parametric model

- Gaussian Process Regression:
  - use prediction averaging; integrate out parameters, to predict conditional density at new data point
    $$p(y|\underline{x}, D, m) = \int p(y|\underline{x}, \theta, m)\, p(\theta|D, m)\, d\theta$$
  - eg, linear regression:
    $$\underline{w}|D \sim N(\hat{\underline{w}}, \underline{\underline{\Sigma}}_w)$$
    $$\Rightarrow y|\underline{x}, D \sim N(\hat{\underline{w}}^T\underline{x},\ \underline{x}^T\underline{\underline{\Sigma}}_w\underline{x} + \sigma^2)$$
    $\left(\begin{array}{l}\underline{w} \sim N(\underline{0}, \gamma^2\underline{\underline{I}}) \\ y|\underline{x} \sim N(\underline{w}^T\underline{x}, \sigma^2)\end{array}\right)$
  - Marginalised linear regression:
    $$\underline{Y}^T \sim N(\underline{0}, \gamma^2\underline{X}^T\underline{X} + \sigma^2\underline{\underline{I}})$$
    $\cancel{\underline{Y}|\underline{X} \sim N(\underline{0},}$
    $\left(\begin{array}{l}E[y_i] = E[\underline{w}^T\underline{x}_i] = \underline{0}^T\underline{x}_i = 0 \\ E[(y_i - \bar{y}_i)^2] = \gamma^2\underline{x}_i^T\underline{x}_i + \sigma^2 \\ E[(y_i - \bar{y}_i)(y_j - \bar{y}_j)] = \gamma^2\underline{x}_i^T\underline{x}_j\end{array}\right)$
  - predicting:
    $$\{\underline{Y}, y\}|\underline{X}, x \sim N\left(\begin{bmatrix}\underline{0} \\ 0\end{bmatrix}, \begin{bmatrix}\gamma^2\underline{X}^T\underline{X} + \sigma^2\underline{\underline{I}} & \gamma\underline{X}^T\underline{x} \\ \gamma\underline{x}^T\underline{X} & \gamma\underline{x}^T\underline{x} + \sigma^2\end{bmatrix}\right)$$
    $$= N\left(\begin{bmatrix}\underline{0} \\ 0\end{bmatrix}, \begin{bmatrix}K_{\underline{X}\underline{X}} & K_{\underline{X}x} \\ K_{x\underline{X}} & K_{xx}\end{bmatrix}\right)$$

(standard result) $\Rightarrow y|\underline{x}, \underline{X}, \underline{Y} \sim N(K_{x\underline{X}}K_{\underline{X}\underline{X}}^{-1}\underline{Y}^T,\ K_{xx} - K_{x\underline{X}}K_{\underline{X}\underline{X}}^{-1}K_{\underline{X}x})$

- can replace $\underline{x}$ with $\phi(\underline{x})$ & for non-linear regression
- as only inner products, use kernel trick;
  $$K(\underline{x}_i, \underline{x}_j) = \phi(\underline{x}_i)^T\phi(\underline{x}_j)$$
  $$\left([K_{XX}]_{ij} = K(\underline{x}_i, \underline{x}_j),\ [K_{Xx}]_i = K(\underline{x}_i, \underline{x}),\ K_{xx} = K(x, x)\right)$$

- can also sample vector from GP (parameterised by mean & covariance function)
- can optimise hyperparameters in kernel by gradient ascent in $\log p(Y|X, K)$