Probabilistic & Unsupervised Learning
Approximate Inference

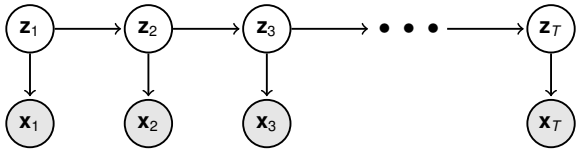
Graphical Models

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2023

Independence, factorisation and graphs



The (Markov) independence structure of a latent chain model implied that the joint-data likelihood factorised:

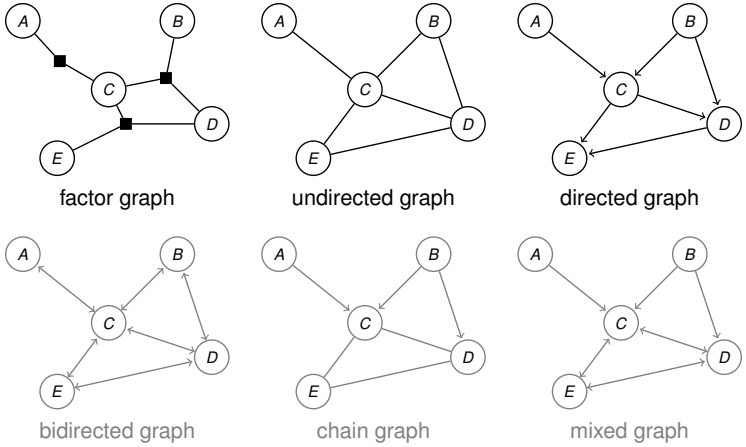
$$P(\mathcal{X}, \mathcal{Z}) = P(\mathbf{z}_1)P(\mathbf{x}_1|\mathbf{z}_1)\prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{z}_{t-1})P(\mathbf{x}_t|\mathbf{z}_t)$$

We exploited the factored form to obtain local $O(T)$ learning algorithms.

- Learning: requires only local marginals of posterior
- Inference: local marginals found by passing local messages

The independence structure of the model (and the factorisation of its likelihood) is encoded in its graph.

Varieties of graphical model



- Nodes in the graph correspond to random variables.
- Edges in graph indicate statistical dependence between the variables.
- (Absent edges signal (conditional) independence between variables).

Why the graph?

- Gives an intuitive representation of the relationships amongst many variables, possibly embodying prior beliefs or knowledge about causal relationships.
  (Examples: inheritance in family trees, noise in electric circuits, neural networks)
- Provides a precise syntax to describe these relationships, and to infer any implied (in)dependencies amongst larger groups of variables.

  Is $A \perp\!\!\!\perp E | \{B, C\}$?
- Each graphical structure corresponds to a parametric family of distributions that satisfy all the implied (in)dependencies.
- Graph-based manipulations allow us to identify the sufficient statistics of these distributions needed for learning, and to construct general-purpose message-passing algorithms that implement inference efficiently.

  Find $P(A|C = c)$ without enumerating all settings of $B, D, E \ldots$

Types of independence

For events or random variables X, Y, V:

Conditional Independence:

$$X \perp\!\!\!\perp Y|V \quad \Leftrightarrow \quad P(X|Y,V) = P(X|V) \qquad \text{[provided, for events, } P(Y,V) > 0\text{]}$$

Thus,

$$X \perp\!\!\!\perp Y|V \quad \Leftrightarrow \quad P(X,Y|V) = P(X|Y,V)P(Y|V) = P(X|V)P(Y|V)$$

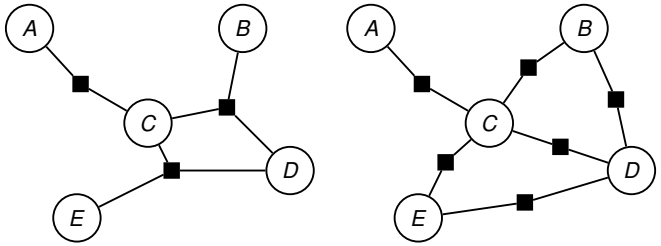
We can generalise to conditional independence between **sets of random variables**:

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{V} \quad \Leftrightarrow \quad \{X \perp\!\!\!\perp Y|\mathcal{V}, \ \forall X \in \mathcal{X} \text{ and } \forall Y \in \mathcal{Y}\}$$

Marginal Independence:

$$X \perp\!\!\!\perp Y \quad \Leftrightarrow \quad X \perp\!\!\!\perp Y|\varnothing \quad \Leftrightarrow \quad P(X,Y) = P(X)P(Y)$$

Factor graphs



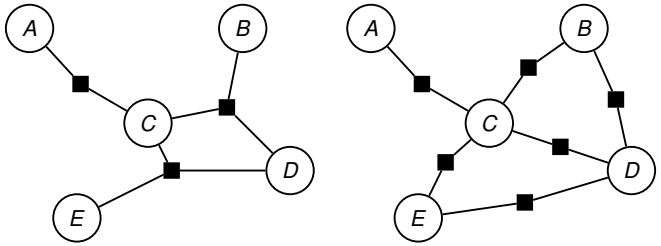
(a) (b)

A factor graph is a direct graphical representation of the factorised model structure: each square indicates a factor over the linked variables.

$$P(\mathcal{X}) = \frac{1}{Z}\prod_j f_j(\mathcal{X}_{C_j})$$

where $\mathcal{X} = \{X_1, \ldots, X_K\}$, $\mathcal{X}_S = \{X_i : i \in S\}$, j indexes the factors, C_j contains the indices of variables adjacent to factor j, f_j is the factor function (also called the factor potential or clique potential) and Z is a normalisation constant.

Factor graphs: examples



(a) (b)

Examples:

(a) $P_a(A,B,C,D,E) = \frac{1}{Z_a} f_1(A,C) f_2(B,C,D) f_3(C,D,E)$

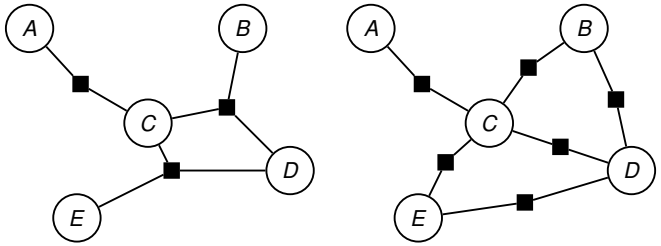
(b) $P_b(A,B,C,D,E) = \frac{1}{Z_b} f_1(A,C) f_2(B,C) f_3(C,D) f_4(B,D) f_5(C,E) f_6(D,E)$

and [e.g.]:

$$Z_a = \sum_{a\in\mathcal{A}} \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}} \sum_{d\in\mathcal{D}} \sum_{e\in\mathcal{E}} f_1(a,c) f_2(b,c,d) f_3(c,d,e)$$

where $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$ are the domains of the corresponding random variables.

Factor graphs: conditional independence



(a) (b)

Conditional independence: $X \perp\!\!\!\perp Y|\mathcal{V}$ if **every** path between X and Y contains some $V \in \mathcal{V}$.

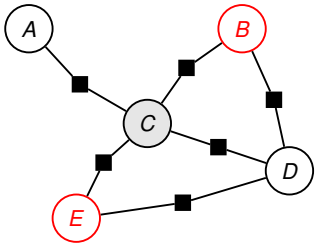
In both graphs:

$A \perp\!\!\!\perp D|C$

$B \not\perp\!\!\!\perp E|C$

$B \perp\!\!\!\perp E|\{C,D\}$

Factorisation and conditional independence



Every path between X and Y contains some $V \in \mathcal{V} \Rightarrow$ there exists a factorisation:

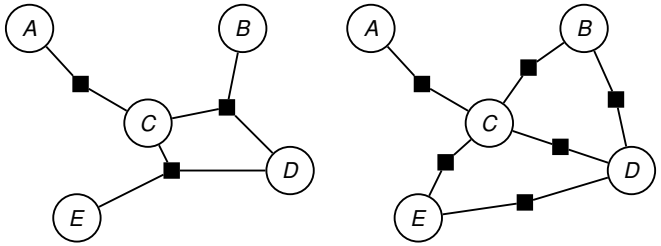
$$P(X, Y, \mathcal{V}, \dots) = \frac{1}{Z} g_X(X, \mathcal{V}_X, \mathcal{Q}_X) g_Y(Y, \mathcal{V}_Y, \mathcal{Q}_Y) g_R(\mathcal{Q}_R, \mathcal{V}_R)$$

where $\mathcal{V}_x, \mathcal{V}_y, \mathcal{V}_R \subseteq \mathcal{V}$ and the sets of remaining variables $\mathcal{Q}_X$, $\mathcal{Q}_Y$ and $\mathcal{Q}_R$ are disjoint.

$$\Rightarrow P(X|Y, \mathcal{V}, \dots) = \frac{P(X, Y, \mathcal{V}, \dots)}{P(Y, \mathcal{V}, \dots)} = \frac{\frac{1}{Z} g_X(X, \mathcal{V}_X, \mathcal{Q}_X) g_Y(Y, \mathcal{V}_Y, \mathcal{Q}_Y) g_R(\mathcal{Q}_R, \mathcal{V}_R)}{\sum_{X'} \frac{1}{Z} g_X(X', \mathcal{V}_X, \mathcal{Q}_X) g_Y(Y, \mathcal{V}_Y, \mathcal{Q}_Y) g_R(\mathcal{Q}_R, \mathcal{V}_R)}$$
$$= \frac{g_X(X, \mathcal{V}_X, \mathcal{Q}_X)}{\sum_{X'} g_X(X', \mathcal{V}_X, \mathcal{Q}_X)}$$

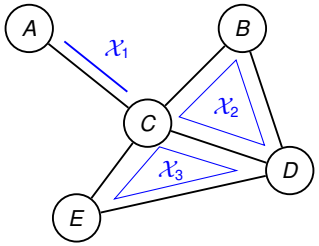
Since the RHS does not depend on Y, it follows that $X \perp\!\!\!\perp Y|\mathcal{V}$.

Factor graphs: neighbourhoods and Markov boundaries



- Variables are neighbours if they share a common factor; the neighbourhood $\mathrm{ne}(X)$ is the set of all neighbours of X.
- Each variable X is **conditionally independent** of all non-neighbours given its neighbours: $X \perp\!\!\!\perp Y|\,\mathrm{ne}(X), \quad \forall Y \notin \{X \cup \mathrm{ne}(X)\}$
  $\Rightarrow \mathrm{ne}(X)$ is a Markov blanket for X.
- In fact, the neighbourhood is the minimal such set: the Markov boundary.

Undirected graphical models: Markov networks



An **undirected graphical model** is a direct representation of conditional independence structure. Nodes are connected iff they are conditionally dependent given all others.

$\Rightarrow$ neighbours (connected nodes) in a Markov net share a factor.

$\Rightarrow$ non-neighbours (disconnected nodes) in a Markov net *cannot* share a factor.

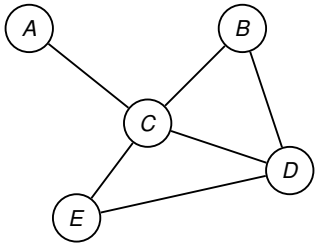
$\Rightarrow$ /cliques/ the joint probability factors over the **maximal cliques** C_j of the graph:

$$P(\mathcal{X}) = \frac{1}{Z} \prod_j f_j(\mathcal{X}_{C_j})$$

It may also factor more finely (as we will see in a moment).

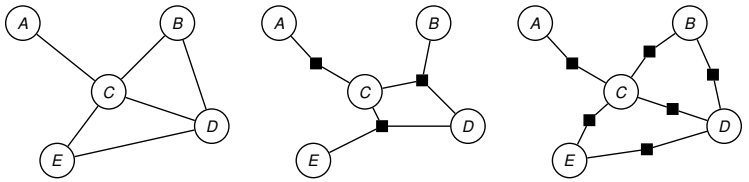
[Cliques are fully connected subgraphs, maximal cliques are cliques not contained in other cliques.]

Undirected graphs: Markov boundaries



- $X \perp\!\!\!\perp Y|\mathcal{V}$ if every path between X and Y contains some node $V \in \mathcal{V}$
- Each variable X is conditionally independent of all non-neighbours given its neighbours: $X \perp\!\!\!\perp Y|\,\mathrm{ne}(X), \quad \forall Y \notin \{X \cup \mathrm{ne}(X)\}$
- $\mathcal{V}$ is a **Markov blanket** for X iff $X \perp\!\!\!\perp Y|\mathcal{V}$ for all $Y \notin \{X \cup \mathcal{V}\}$.
- **Markov boundary:** minimal Markov blanket. For undirected graphs (like factor graphs) this is the set of neighbours of X.

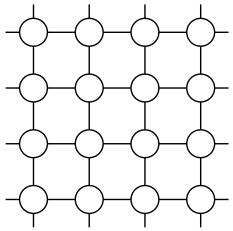
Undirected graphs and factor graphs



- Each node has the same neighbours in each graph, so (a), (b) and (c) represent exactly the same conditional independence relationships.
- The implied maximal factorisations differ: (b) has two three-way factors; (c) has only pairwise factors; (a) cannot distinguish between these (so we have to adopt factorisation (b) to be safe).
- Suppose all variables are discrete and can take on K possible values. Then the functions in (a) and (b) are tables with $\mathcal{O}(K^3)$ cells, whereas in (c) they are $\mathcal{O}(K^2)$.
- Factor graphs have richer expressive power than undirected graphical models.
- Factors cannot be determined solely by testing for conditional independence.

Some examples of undirected graphical models

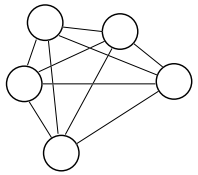
- Markov random fields (used in computer vision)



- Maximum entropy language models (used in speech and language modelling)

$$P(\mathcal{X}) = \frac{1}{Z} p_0(\mathcal{X}) \exp\left\{\sum_j \lambda_j g_j(\mathcal{X})\right\}$$

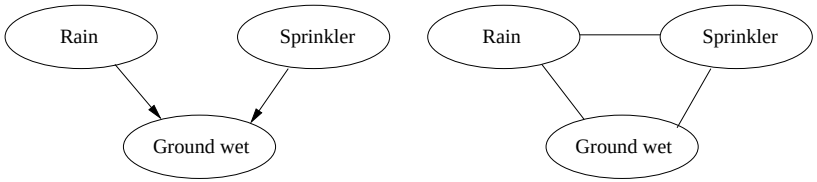
- Conditional random fields are undirected graphical models (conditioned on the input variables).
- Boltzmann machines (a kind of neural network/Ising model)



Limitations of undirected and factor graphs

Undirected and factor graphs fail to capture some useful independencies—a pair of variables may be connected merely because some other variable depends on them:

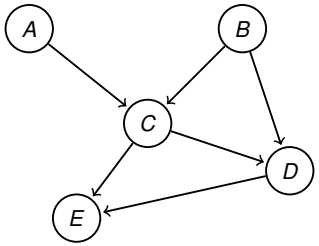
The classic example (due to Pearl):



- Most sprinklers switch on come rain or shine; and certainly the weather pays no heed to the state of the spinklers.
- **Explaining away**: Damp ground suggests that it has rained; but if we also see a running sprinkler this explains away the damp, returning our belief about rain to the prior.
- $R \perp\!\!\!\perp S | \varnothing$ but $R \not\perp\!\!\!\perp S | G$.

This highlights the difference between **marginal** and **conditional independence**.

Directed acyclic graphical models



A **directed acyclic graphical (DAG) model** represents a factorization of the joint probability distribution in terms of local (Markov) conditionals:

$$\begin{aligned} P(A,B,C,D,E) &= P(A)P(B|A)P(C|A,B)P(D|A,B,C)P(E|A,B,C,D) \\ &= P(A)P(B)P(C|A,B)P(D|B,C)P(E|C,D) \end{aligned}$$

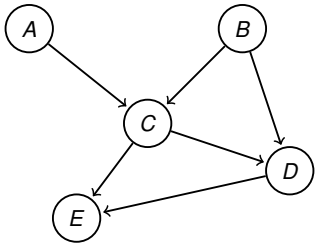
In general:

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i|X_{\text{pa}(i)})$$

where $\text{pa}(i)$ are the parents of node i.

DAG models are also known as **Bayesian networks** or **Bayes nets**.

Conditional independence in DAGs

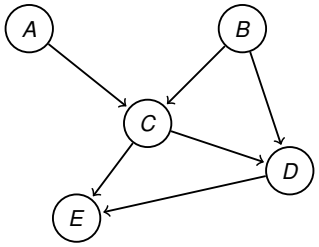


Reading conditional independence from DAGs is more complicated than in undirected graphs.

- $A \perp\!\!\!\perp E \mid \{B, C\}$: conditioning nodes block paths
- $A \perp\!\!\!\perp B \mid \varnothing$: *other* nodes block reflected paths
- $A \not\perp\!\!\!\perp B \mid C$: conditioning node *creates* a reflected path by explaining away
- $A \not\perp\!\!\!\perp E \mid C$: the created path extends to E via D
- $A \perp\!\!\!\perp E \mid \{C, D\}$: but is blocked by observing D

So conditioning on (i.e. observing) nodes can both create and remove dependencies.

The Bayes-ball algorithm



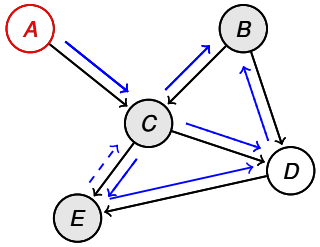
Game: can you get a ball from X to Y without being blocked by $\mathcal{V}$? If so, $X \not\perp\!\!\!\perp Y \mid \mathcal{V}$.

Rules: ball follow edges, and are passed on or bounced back from nodes according to:

- Nodes $V \notin \mathcal{V}$ pass balls down or up chains: $\rightarrow V \rightarrow$ or $\leftarrow V \leftarrow$.
- Nodes $V \notin \mathcal{V}$, bounce balls **from** children **to** children.
- Nodes $V \in \mathcal{V}$, bounce balls **from** parents **to** parents (including returning the ball whence it came).

Otherwise the ball is blocked. (So $V \in \mathcal{V}$ blocks all balls from children, and stops balls from parents reaching children.)

D-separation



So when is $X \perp\!\!\!\perp Y \mid \mathcal{V}$?

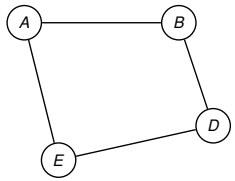
Consider *every* undirected path (i.e. ignoring arrows) between X and Y. The path is **blocked** by $\mathcal{V}$ if there is a node V on the path such that either:

- V has convergent arrows ($\rightarrow V \leftarrow$) on the path (*i.e.*, V is a "collider node") and **neither** V **nor its descendents** $\in \mathcal{V}$.
- V does not have convergent arrows on the path ($\rightarrow V \rightarrow$ or $\leftarrow V \rightarrow$) and $V \in \mathcal{V}$. This is similar to the undirected graph semantics.

If all paths are blocked, we say $\mathcal{V}$ **d-separates** X from Y (d for directed), and $X \perp\!\!\!\perp Y \mid \mathcal{V}$.

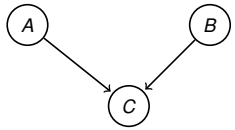
Markov boundary for X: $\{\mathsf{pa}(X) \cup \mathsf{ch}(X) \cup \mathsf{pa}(\mathsf{ch}(X))\}$.

Expressive power of directed and undirected graphs



No DAG can represent these and only these independencies

No matter how we direct the arrows there will always be two non-adjacent parents sharing a common child $\Rightarrow$ dependence in DAG but independence in undirected graph.



No undirected or factor graph can represent these and only these independencies

One three-way factor, but this does not encode marginal independence.

Graphs, conditional independencies, and families of distributions

Each graph G implies a set of conditional independence statements $\mathcal{C}(G) = \{X_i \perp\!\!\!\perp Y_i | \mathcal{V}_i\}$.

Each such set $\mathcal{C}$, defines a family of distributions that satisfy all the statements in $\mathcal{C}$:

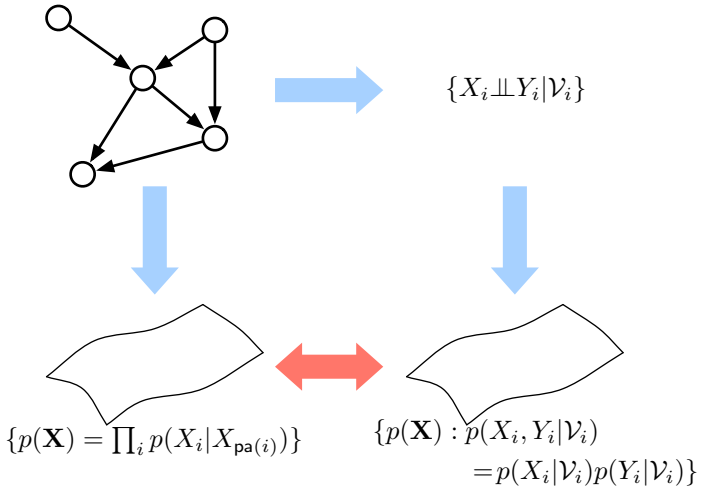
$$\mathcal{P}_{\mathcal{C}(G)} = \{P(\mathcal{X}) \ : \ P(X_i, Y_i|\mathcal{V}_i) = P(X_i|\mathcal{V}_i)P(Y_i|\mathcal{V}_i) \text{ for all } X_i \perp\!\!\!\perp Y_i|\mathcal{V}_i \text{ in } \mathcal{C} \ \}$$

G may also encode a family of distributions by their functional form, e.g. for a factor graph

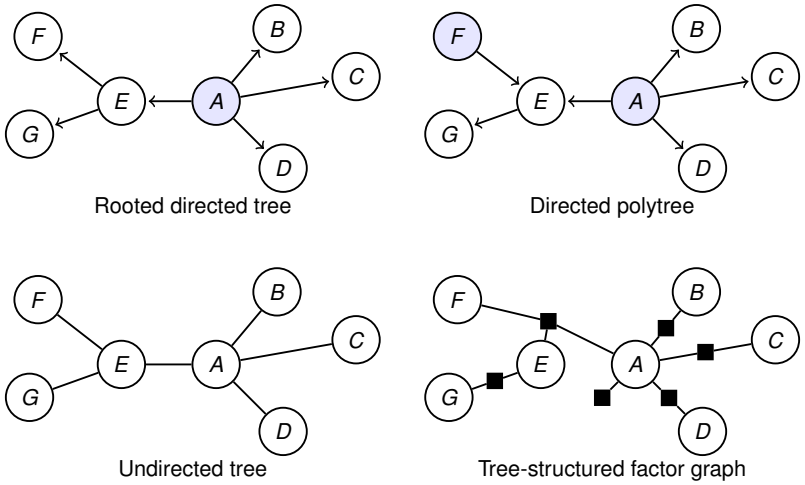
$$\mathcal{P}_G = \{P(\mathcal{X}) \ : \ P(\mathcal{X}) = \tfrac{1}{Z}\textstyle\prod_j f_j(\mathcal{X}_{C_j}), \text{ for some non-negative functions } f_j \ \}$$

- For directed graphs, $\mathcal{P}_G = \mathcal{P}_{\mathcal{C}(G)}$.
- For undirected graphs, $\mathcal{P}_G = \mathcal{P}_{\mathcal{C}(G)}$ if all distributions are positive, i.e. $P(\mathcal{X}) > 0$ for all values of $\mathcal{X}$ (Hammersley-Clifford Theorem).
- There are factor graphs for which $\mathcal{P}_G \neq \mathcal{P}_{\mathcal{C}_G}$.
- Factor graphs are more expressive than undirected graphs: for every undirected graph G_1 there is a factor graph G_2 with $\mathcal{P}_{G_1} = \mathcal{P}_{G_2}$ but not vice versa.
- Adding edges to graph $\Rightarrow$ removing conditional independency statements $\Rightarrow$ enlarging the family of distributions (converse true for removing edges).

Graphs, conditional independencies, and families of distributions

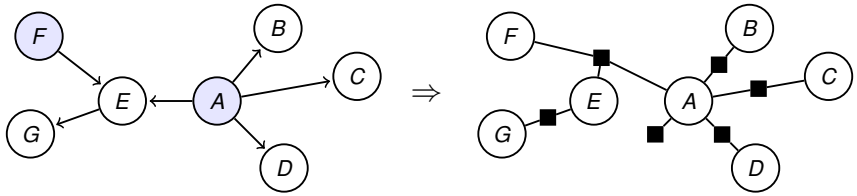


Tree-structured graphical models



These are all tree-structured or "singly-connected" graphs.

Polytrees to tree-structured factor graphs



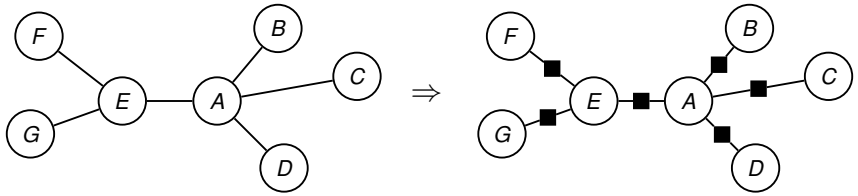
Polytrees are tree-structured DAGs that may have more than one root.

$$P(\mathcal{X}) = \prod_i P(X_i|X_{\text{pa}(i)})$$
$$= \prod_i f_i(X_{C_i})$$

where $C_i = i \cup \text{pa}(i)$ and $f_i(X_{C_i}) = P(X_i|X_{\text{pa}(i)})$.

Marginal distribution on roots $P(X_r)$ absorbed into an adjacent factor.

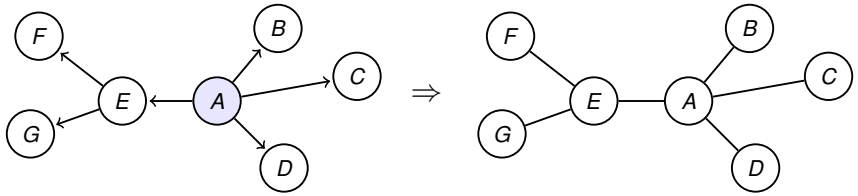
Undirected trees and factor graphs



In an undirected tree all maximal cliques are of size 2, and so the equivalent factor graph has only pairwise factors.

$$P(\mathcal{X}) = \frac{1}{Z} \prod_{\text{edges } (ij)} f_{(ij)}(X_i, X_j)$$

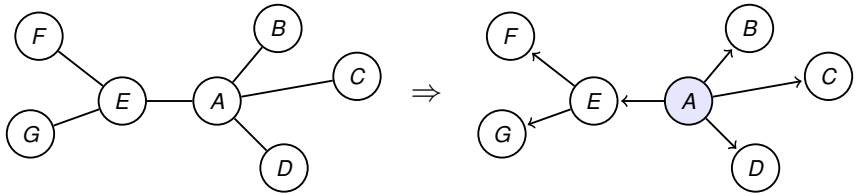
Rooted directed trees to undirected trees



The distribution for a single-rooted directed tree can be written as a product of pairwise factors $\Rightarrow$ undirected tree.

$$\begin{aligned} P(\mathcal{X}) &= P(X_r) \prod_{i \neq r} P(X_i | X_{\text{pa}(i)}) \\ &= \prod_{\text{edges } (ij)} f_{(ij)}(X_i, X_j) \end{aligned}$$

Undirected trees to rooted directed trees



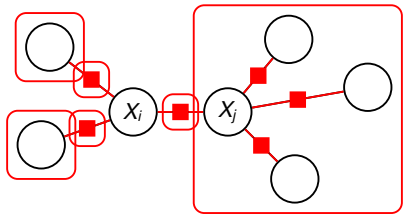
This direction is slightly trickier:

- Choose an arbitrary node X_r to be the root and point all the arrows away from it
- Compute the marginal distributions on single nodes $P(X_i)$ and on edges $P(X_i, X_j)$ implied by the undirected graph.
- Compute the conditionals in the DAG:

$$\begin{aligned} P(\mathcal{X}) &= P(X_r) \prod_{i \neq r} P(X_i | X_{\text{pa}(i)}) = P(X_r) \prod_{i \neq r} \frac{P(X_i, X_{\text{pa}(i)})}{P(X_{\text{pa}(i)})} \\ &= \frac{\prod_{\text{edges } (ij)} P(X_i, X_j)}{\prod_{\text{nodes } i} P(X_i)^{\text{deg(i)-1}}} \end{aligned}$$

How do we compute $P(X_i)$ and $P(X_i, X_j)$? $\Rightarrow$ **Belief propagation**.

Finding marginals in undirected trees

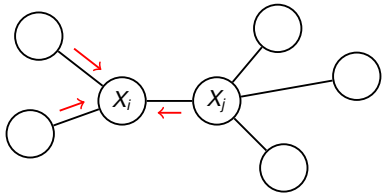


Undirected tree $\Rightarrow$ pairwise factored joint distribution: $P(\mathcal{X}) = \frac{1}{Z} \prod_{(ij) \in \mathcal{E}_T} f_{(ij)}(X_i, X_j)$

Each neigbour X_j of X_i defines a disjoint subtree $T_{j \to i}$. So we can split up the product:

$$\begin{aligned} P(X_i) = \sum_{\mathcal{X} \setminus \{X_i\}} P(\mathcal{X}) &\propto \sum_{\mathcal{X} \setminus \{X_i\}} \prod_{(ij) \in \mathcal{E}_T} f_{(ij)}(X_i, X_j) \\ &= \sum_{\mathcal{X} \setminus \{X_i\}} \prod_{X_j \in \text{ne}(X_i)} f_{(ij)}(X_i, X_j) \prod_{(i'j') \in \mathcal{E}_{T_{j \to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \end{aligned}$$

Finding marginals in undirected trees

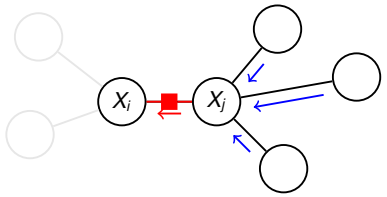


Undirected tree $\Rightarrow$ pairwise factored joint distribution: $P(\mathcal{X}) = \frac{1}{Z} \prod_{(ij)\in\mathcal{E}_T} f_{(ij)}(X_i, X_j)$

Each neigbour X_j of X_i defines a disjoint subtree $T_{j\to i}$. So we can split up the product:

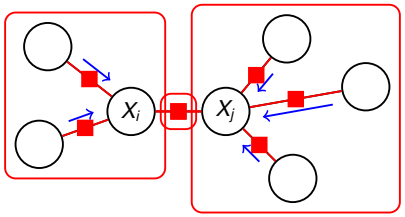
$$\begin{aligned} P(X_i) = \sum_{\mathcal{X}\setminus\{X_i\}} P(\mathcal{X}) &\propto \sum_{\mathcal{X}\setminus\{X_i\}} \prod_{(ij)\in\mathcal{E}_T} f_{(ij)}(X_i, X_j) \\ &= \sum_{\mathcal{X}\setminus\{X_i\}} \prod_{X_j\in\mathrm{ne}(X_i)} f_{(ij)}(X_i, X_j) \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \\ &= \prod_{X_j\in\mathrm{ne}(X_i)} \underbrace{\Big(\sum_{\mathcal{X}_{T_{j\to i}}} \Big(f_{(ij)}(X_i, X_j) \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \Big) \Big)}_{M_{j\to i}(X_i)} = \prod_{X_j\in\mathrm{ne}(X_i)} M_{j\to i}(X_i) \end{aligned}$$

Message recursion: Belief Propagation (BP)



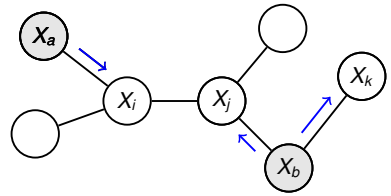
$$\begin{aligned} M_{j\to i}(X_i) &= \sum_{\mathcal{X}_{T_{j\to i}}} f_{(ij)}(X_i, X_j) \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \\ &= \sum_{X_j} f_{(ij)}(X_i, X_j) \underbrace{\sum_{\mathcal{X}_{T_{j\to i}}\setminus X_j} \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'})}_{\propto P_{T_{j\to i}}(X_j) \propto \prod_{X_k\in\mathrm{ne}(X_j)\setminus X_i} M_{k\to j}(X_j)} \\ &= \sum_{X_j} f_{(ij)}(X_i, X_j) \prod_{X_k\in\mathrm{ne}(X_j)\setminus X_i} M_{k\to j}(X_j) \end{aligned}$$

BP for pairwise marginals in undirected trees



$$\begin{aligned} P(X_i, X_j) = \sum_{\mathcal{X}\setminus\{X_i,X_j\}} P(\mathcal{X}) &\propto \sum_{\mathcal{X}\setminus\{X_i,X_j\}} \prod_{(i'j')\in\mathcal{E}_T} f_{(i'j')}(X_{i'}, X_{j'}) \\ &= \sum_{\mathcal{X}\setminus\{X_i,X_j\}} f_{(ij)}(X_i, X_j) \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \prod_{(i'j')\in\mathcal{E}_{T_{i\to j}}} f_{(i'j')}(X_{i'}, X_{j'}) \\ &= f_{(ij)}(X_i, X_j) \Big(\sum_{\mathcal{X}_{T_{j\to i}}\setminus X_j} \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \Big) \Big(\sum_{\mathcal{X}_{T_{i\to j}}\setminus X_i} \prod_{(i'j')\in\mathcal{E}_{T_{i\to j}}} f_{(i'j')}(X_{i'}, X_{j'}) \Big) \\ &= f_{(ij)}(X_i, X_j) \prod_{X_k\in\mathrm{ne}(X_j)\setminus X_i} M_{k\to j}(X_j) \prod_{X_k\in\mathrm{ne}(X_i)\setminus X_j} M_{k\to i}(X_i) \end{aligned}$$

BP for inference



Messages from observed leaf nodes are conditioned rather than marginalised:

To compute $P(X_i)$: $\quad M_{a\to i} = \sum_{X_a} f_{ai}(X_a, X_i)$

To compute $P(X_i|X_a = a)$: $\quad M_{a\to i} = f_{ai}(X_a = a, X_i)$

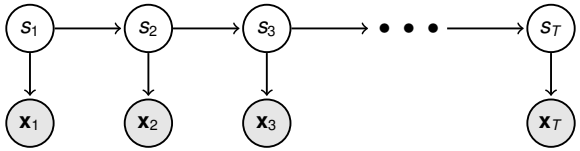
Observed internal nodes partition the graph, and so messages propagate independently.

$M_{b\to j} = f_{bj}(X_b = b, X_j) \qquad M_{b\to k} = f_{bk}(X_b = b, X_k)$

Messages $M_{i\to j}$ are proportional to the likelihood based on any observed variables ($\mathcal{O}$) within the messages subtree $T_{i\to j}$, possibly scaled by a prior factor (depending on factorisation)

$$M_{i\to j}(X_j) \propto P(\mathcal{X}_{T_{i\to j}} \cap \mathcal{O}|X_j) P(X_j)$$

BP for latent chain models



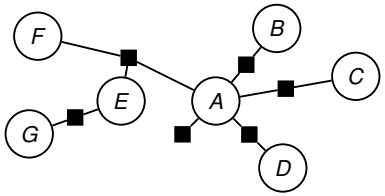
A latent chain model is a rooted directed tree $\Rightarrow$ an undirected tree.

The forward-backward algorithm is just BP on this graph.

$$\alpha_t(i) \Leftrightarrow M_{s_{t-1}\to s_t}(s_t{=}i) \propto P(\mathbf{x}_{1:t}, s_t)$$
$$\beta_t(i) \Leftrightarrow M_{s_{t+1}\to s_t}(s_t{=}i) \propto P(\mathbf{x}_{t+1:T}|s_t)$$
$$\alpha_t(i)\beta_t(i) = \prod_{j\in \mathrm{ne}(s_t)} M_{j\to s_t}(s_t{=}i) \propto P(s_t{=}i|\mathcal{O})$$

Algorithms like BP extend the power of graphical models beyond just encoding of independence and factorisation. A single derivation serves for a wide array of models.

BP for polytrees (tree-structured factor graphs)



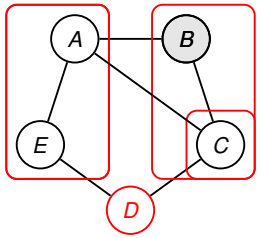
- Associated undirected tree contains loops created by non-binary cliques, but exact BP is still possible treating factors as message-passing nodes.
- Define uniform potentials on nodes, factor potentials on factors.
- Then incoming and outgoing factor messages are.
- For factor $f_a(X_i, X_j, X_k, \dots)$:

  Incoming: (node→factor) $\quad M_{i\to a}(X_i) = \prod_{x\in Ne(i)\setminus a} M_{x\to i}(X_i)$

  Outgoing: (factor→node) $\quad M_{a\to i}(X_i) = \sum_{\mathcal{X}_a\setminus X_i} f_a(X_i, X_j, X_k, \dots) \prod_{x\in Ne(a)\setminus i} M_{x\to a}(X_x)$

- Initiate at leaf nodes (or singleton factors).
- Schedule messages inward – outward as in undirected trees.

BP in non-trees?



Can we find $P(D)$ easily?

- Neighbours do not belong to disjoint subtrees, so influence of other nodes cannot be separated into messages.
- Observed nodes may break loops and make subtrees independent, but may not resolve all loops.

Possible strategies:

- Propagate local messages anyway, and hope for the best
  - loopy belief propagation — actually an approximation which we will study later.
- Group variables together into multivariate nodes until the resulting graph is a tree.
  - Junction tree

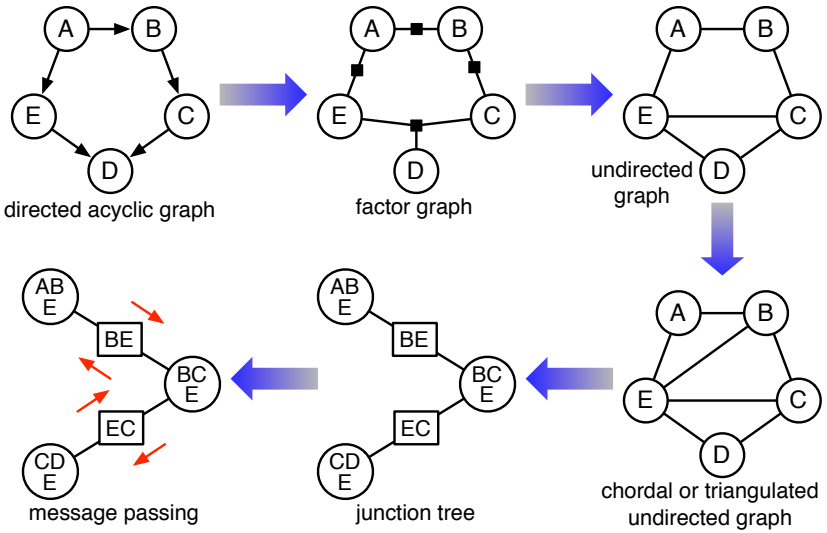
Graph transformations

For exact inference in arbitrary graphical models we need to transform the given graph into one that will be easier to handle (specifically a tree: the **junction or join tree**).

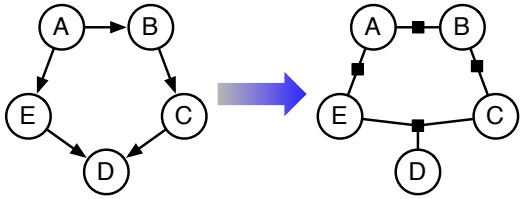
The original graph G encoded a distribution $P(\mathcal{X})$ with a certain factorisation or independence structure.

- Transformation from G to an easy-to-handle G' will only be valid if $P(\mathcal{X})$ can also be represented by G'.
- This can be ensured if every step of the graph transformation only removes conditional independencies, never adds them.
- Thus the family of possible encoded distributions grows or stays the same at each step, and $P(\mathcal{X})$ will be in the family of distributions represented by G'.
- The factor potentials on the new graph G' are built from those given on G, so as to encode the same distribution.
- Then inference on G' with the appropriate potentials acts on $P(\mathcal{X})$.

The Junction Tree algorithm



DAG to factor graph



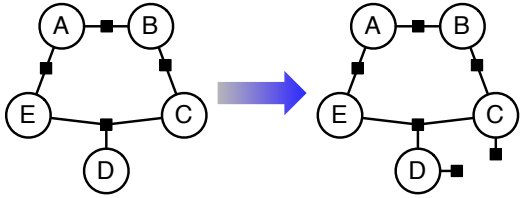
Factors are simply the conditional distributions in the DAG.

$$P(\mathcal{X}) = \prod_i P(X_i|X_{\text{pa}(i)})$$
$$= \prod_i f_i(X_{C_i})$$

where $C_i = i \cup \text{pa}(i)$ and $f_i(X_{C_i}) = P(X_i|X_{\text{pa}(i)})$.

Marginal distribution on root(s) $P(X_r)$ absorbed into an adjacent factor.

Observations in a factor graph



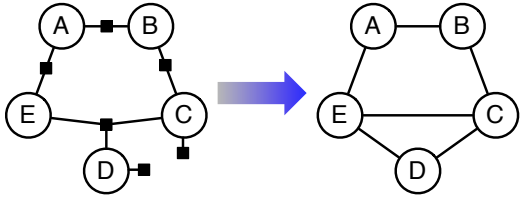
Inference usually targets a posterior marginal given a set of observed values $P(X_i|\mathcal{O})$
e.g. $P(A|D = \text{wet}, C = \text{rain})$.

Formally, we can either modify the factors linked to observed nodes, or add singleton factors adjacent to the observed nodes, e.g.

$$f_D(D) = \begin{cases} 1 & \text{if } D = \text{wet;} \\ 0 & \text{otherwise.} \end{cases}$$

$$f_C(C) = \begin{cases} 1 & \text{if } C = \text{rain;} \\ 0 & \text{otherwise.} \end{cases}$$

Factor graph to undirected graph



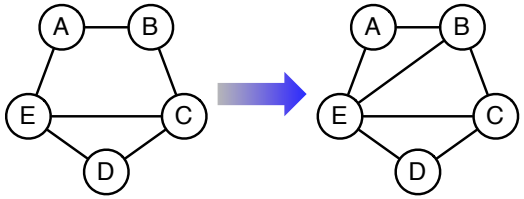
The next step (triangulation) will depend on an undirected graph. Every factor from the DAG must be contained *within* a maximal clique of the undirected graph.

- Replace each factor by an undirected clique (i.e. place edge between every pair of nodes in the factor).
- Construct the potentials on each maximal clique by multiplying together factor potentials that fall within it; ensuring each factor potential only appears once.

The transformation from DAG $\Rightarrow$ undirected graph is called **moralization**:

- "marry" all parents of each node by adding an edge to connect them
- drop arrows on all the edges

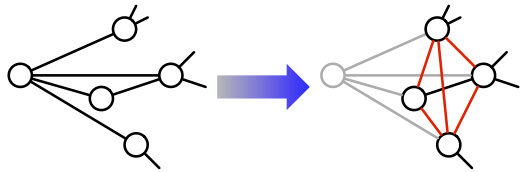
Triangulating the undirected graph



The undirected graph may have loops, which would interfere with belief propagation.

- Could join loops into cliques, but this is inefficient.
- **Triangulation**: add edges to the graph so every loop of size ≥ 4 has at least one chord.
- Recursive: new edges may create new loops; ensure new loops ≥ 4 have chords too.
- An undirected graph where every loop of size ≥ 4 has at least one chord is called **chordal** or **triangulated**.
- Adding edges always *removes* conditional independencies, enlarging the family of distributions that the graph can encode.
- Many ways to add chords; in general finding the best triangulation is NP-complete.
- One approach: **variable elimination**.

Variable elimination

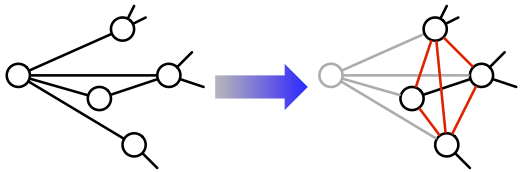


Imagine marginalising the distribution one variable at a time (eliminating each from the graph). Let the order of elimination be $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(n)}$:

$$\begin{aligned} P(X_{\sigma(n)}) &= \sum_{X_{\sigma(n-1)}} \cdots \sum_{X_{\sigma(1)}} P(\mathcal{X}) = \frac{1}{Z} \sum_{X_{\sigma(n-1)}} \cdots \sum_{X_{\sigma(2)}} \sum_{X_{\sigma(1)}} \prod_i f_i(\mathcal{X}_{C_i}) \\ &= \frac{1}{Z} \sum_{X_{\sigma(n-1)}} \cdots \sum_{X_{\sigma(2)}} \prod_{j: C_j \not\ni \sigma(1)} f_j(\mathcal{X}_{C_j}) \sum_{X_{\sigma(1)}} \prod_{i: C_i \ni \sigma(1)} f_i(\mathcal{X}_{C_i}) \\ &= \frac{1}{Z} \sum_{X_{\sigma(n-1)}} \cdots \sum_{X_{\sigma(2)}} \prod_{j: C_j \not\ni \sigma(1)} f_j(\mathcal{X}_{C_j}) f_{\text{new}}(\mathcal{X}_{C_{\text{new}}}) \end{aligned}$$

where $C_{\text{new}} = \text{ne}(X_{\sigma(1)})$, and edges are added to the graph connecting all nodes in C_{new}.

Variable elimination



Theorem: a graph including all edges that would be induced by elimination is chordal.

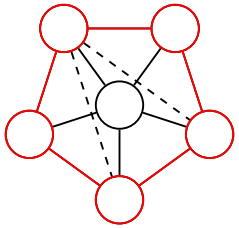
Finding a good triangulation depends on finding a good order of elimination $\sigma(1), \ldots, \sigma(n)$. This is also NP-complete.

Heuristics: pick next variable to eliminate by

- Minimum deficiency search: choose variable that induces the fewest new edges.
- Maximum cardinality search: choose variable with most previously visited neighbours.

Minimum deficiency search seems (empirically) to be better.

Triangulation may not be obvious by inpection

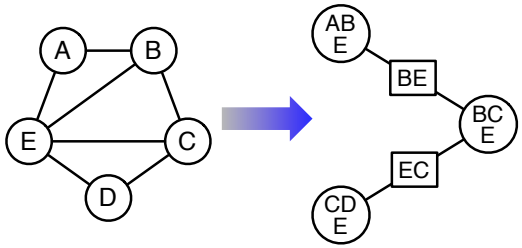


Is this graph triangulated?

No. Chords must be direct connections — they cannot step through an intermediate node.

Detecting unchorded loops by inspection rapidly becomes difficult in large graphs, necessitating automated algorithms such as variable elimination.

Chordal graph to the junction tree



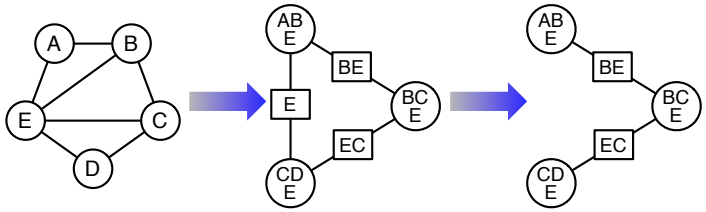
A **junction tree** (or **join tree**) is a tree whose nodes and edges are labelled with sets of variables.

Each node represents a clique; edges are labelled by the intersections of cliques, called separators.

- Cliques contain all adjacent separators.
- **Running intersection property**: if two cliques contain variable X, all cliques and separators on the path between the two cliques contain X.

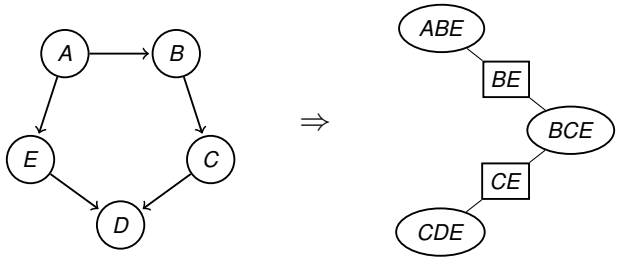
The running intersection property is required for consistency.

Chordal graph to the junction tree



- Find the maximal cliques $C_1, \ldots, C_k$ of the chordal undirected graph (each clique consists of an eliminated variable and its neighbours, so finding maximal cliques is easy).
- Construct a weighted graph, with nodes labelled by the maximal cliques and edges connecting each pair of cliques that shares variables (labelled by the variables in the intersection).
- Define the weight of an edge to be the size of the separator.
- Find the maximum-weight spanning tree of the weighted graph.

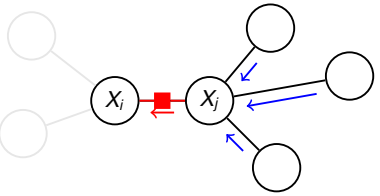
Messages on the junction tree



We've now completed the transformation from a general model to a tree-structured graph.

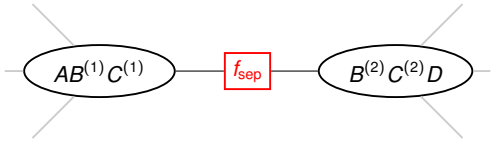
- Belief propagation on the junction tree should allow us to efficiently compute posterior marginals for inference and learning.

Recall: BP on undirected trees



$$
\begin{aligned}
M_{j\to i}(X_i) &= \sum_{\mathcal{X}_{T_{j\to i}}} f_{(ij)}(X_i, X_j) \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'}) \\
&= \sum_{X_j} f_{(ij)}(X_i, X_j) \underbrace{\sum_{\mathcal{X}_{T_{j\to i}}\setminus X_j} \prod_{(i'j')\in\mathcal{E}_{T_{j\to i}}} f_{(i'j')}(X_{i'}, X_{j'})}_{\propto P_{T_{j\to i}}(X_j) \propto \prod_{X_k\in \mathrm{ne}(X_j)\setminus X_i} M_{k\to j}(X_j)} \\
&= \sum_{X_j} f_{(ij)}(X_i, X_j) \prod_{X_k\in \mathrm{ne}(X_j)\setminus X_i} M_{k\to j}(X_j)
\end{aligned}
$$

Message passing on junction trees



Maximal cliques in the chordal graph are nodes of the junction tree. Thus, the joint distribution factors over the JT nodes:

$$P(\mathcal{X}) = \frac{1}{Z}\prod_i f_i(\mathcal{X}_{C_i}) = \ldots f_{ABC}(A,B,C) f_{BCD}(B,C,D)\ldots$$

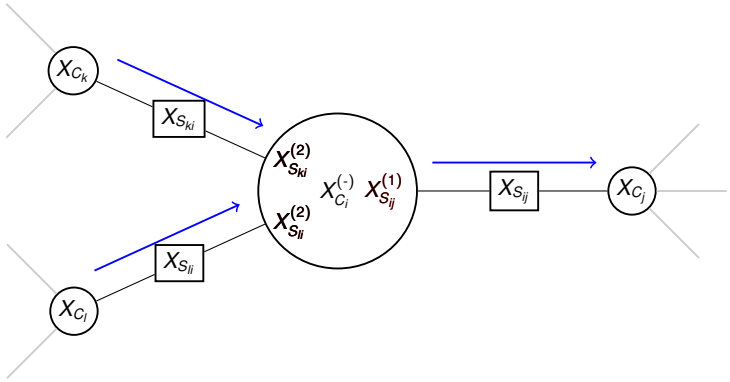
This appears to violate the usual undirected tree semantics of a factor per *edge*.

However: the appearance of the same variables in multiple nodes introduces dependencies:

- Introduce copy variables on each side of the separator.
- Factors on nodes no longer overlap.
- New delta-function factors on separators enforce consistency amongst copies:

$$P(\mathcal{X}) = \ldots f_{ABC}(A,B^{(1)},C^{(1)}) \underbrace{\delta(B^{(1)}-B^{(2)})\delta(C^{(1)}-C^{(2)})}_{f_{\text{sep}}(B^{(1)},C^{(1)},B^{(2)},C^{(2)})} f_{BCD}(B^{(2)},C^{(2)},D)\ldots$$

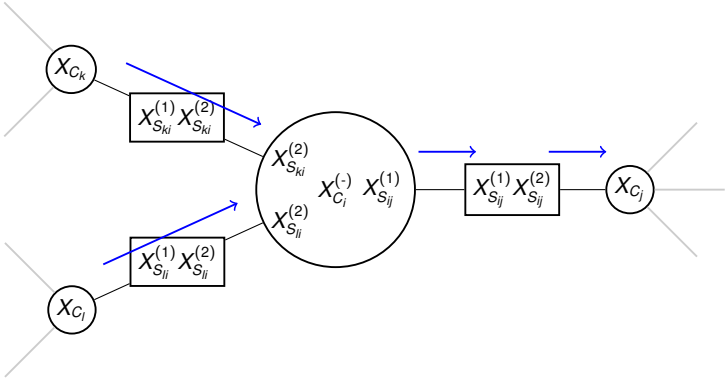
Message passing on junction trees



We can use this view to define BP messages on the junction tree:

- Copy and partition clique variables X_{C_i}:
  - Unshared variables: $X^{(-)}_{C_i} = X_{C_i} \setminus \bigcup S_{ik}$
  - Variables in incoming separators: $X^{(2)}_{S_{ki}}$ (matching variables $X^{(1)}_{S_{ki}}$ in $k \in \text{ne}(i)\setminus j$).
  - Variables in outgoing separator: $X^{(1)}_{S_{ij}}$ (matching variables $X^{(2)}_{S_{ij}}$ in clique j).
  - (Variables that appear in more than one separator will need additional copies.)

Message passing on junction trees

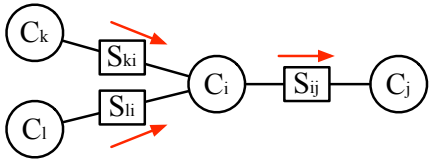


Messages become:

$$M_{S_{ij}\to C_j}(X^{(2)}_{S_{ij}}) = \sum_{X^{(1)}_{S_{ij}}, X^{(-)}_{C_i}, \{X^{(2)}_{S_{ki}}\}} \delta(X^{(2)}_{S_{ij}} - X^{(1)}_{S_{ij}}) f_i(X^{(1)}_{S_{ij}}, X^{(-)}_{C_i}, \{X^{(2)}_{S_{ki}}\}) \prod_{k\in\text{ne}(i)\setminus j} M_{k\to i}(X^{(2)}_{S_{ki}})$$

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

Shafer-Shenoy propagation



Messages computed recursively by:

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

And marginal distributions on cliques and separators are:

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

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

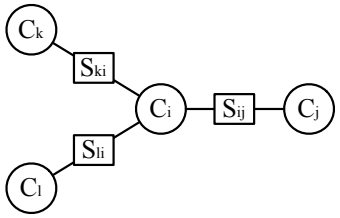
This is called **Shafer-Shenoy propagation**.

Consistency

The running intersection property and tree structure of the junction trees implies that **local consistency** between cliques and separator marginals guarantees **global consistency**.

If $q_i(X_{C_i})$, $r_{ij}(X_{S_{ij}})$ are distributions such that

$$\sum_{X_{C_i \setminus S_{ij}}} q_i(X_{C_i}) = r_{ij}(X_{S_{ij}})$$



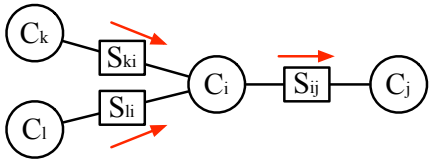
Then the following

$$P(\mathcal{X}) = \frac{\prod_{\text{cliques } i} q_i(X_{C_i})}{\prod_{\text{separators } (ij)} r_{ij}(X_{S_{ij}})}$$

is also a distribution (non-negative and sums to one) such that:

$$q_i(X_{C_i}) = \sum_{\mathcal{X} \setminus X_{C_i}} P(\mathcal{X}) \qquad r_{ij}(X_{S_{ij}}) = \sum_{\mathcal{X} \setminus X_{S_{ij}}} P(\mathcal{X})$$

Hugin propagation



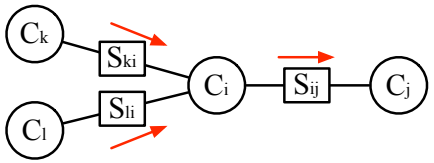
Some properties of Hugin propagation:

- The defined distribution $P(\mathcal{X})$ is unchanged by the updates.
- Each update introduces a local consistency constaint:

$$\sum_{X_{C_j \setminus S_{ij}}} q_j(X_{C_j}) = r_{ij}(X_{S_{ij}})$$

- If each update $i \to j$ is carried out only after incoming updates $k \to i$ have been carried out, then each update needs only be carried out once.
- Each Hugin update is equivalent to the corresponding Shafer-Shenoy update.

Reparameterisation for message passing



Hugin propagation is a different (but equivalent) message passing algorithm.
It is based upon the idea of **reparameterisation**. Initialize:

$$q_i(X_{C_i}) \propto f_i(X_{C_i}) \qquad r_{ij}(X_{S_{ij}}) \propto 1$$

Then our probability distribution is initially

$$P(\mathcal{X}) \propto \frac{\prod_{\text{cliques } i} q_i(X_{C_i})}{\prod_{\text{separators } (ij)} r_{ij}(X_{S_{ij}})}$$

A Hugin propagation update for $i \to j$ is:

$$r_{ij}^{\text{new}}(X_{S_{ij}}) = \sum_{X_{C_i \setminus S_{ij}}} q_i(X_{C_i}) \qquad q_j^{\text{new}}(X_{C_j}) = q_j(X_{C_j}) \frac{r_{ij}^{\text{new}}(X_{S_{ij}})}{r_{ij}(X_{S_{ij}})}$$

Computational Costs of the Junction Tree Algorithm

- Most of the computational cost of the junction tree algorithm is incurred during the message passing phase.
- The running and memory costs of the message passing phase are both $O(\sum_i |\mathcal{X}_{C_i}|)$. This can be significantly (exponentially) more efficient than brute force.
- The variable elimination ordering heuristic can have very significant impact on the message passing costs.
- For certain classes of graphical models (e.g. 2D lattice Markov random field) it is possible to hand-craft an efficient ordering.

Other Inference Algorithms

There are other approaches to inference in graphical models which may be more efficient under specific conditions:

Cutset conditioning: or "reasoning by assumptions". Find a small set of variables which, if they were given (i.e. known) would render the remaining graph "simpler". For each value of these variables run some inference algorithm on the simpler graph, and average the resulting beliefs with the appropriate weights.

Loopy belief propagation: just use belief propagation even though there are loops. No guarantee of convergence, but often works well in practice. Some (weak) guarantees about the nature of the answer if the message passing *does* converge.

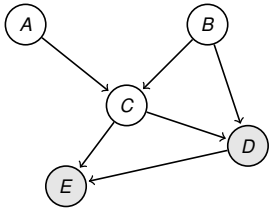
Second half of course: we will learn about a variety of approximate inference algorithms when the graphical model is so large/complex that no exact inference algorithm can work efficiently.

Learning in Graphical Models

We have discussed inference at length — what about learning? The factored structure implied by the graph also makes learning easy.

Consider data points comprising observations of a subset of variables.
ML learning $\Rightarrow$ adjust parameters to maximise:

$$\begin{aligned}\mathcal{L} &= P(X_{\text{obs}}|\theta) \\ &= \sum_{X_{\text{unobs}}} P(X_{\text{obs}}, X_{\text{unobs}}|\theta)\end{aligned}$$



by EM, need to maximise

$$\begin{aligned}\mathcal{F}(q,\theta) &= \langle \log P(X_{\text{obs}}, X_{\text{unobs}}|\theta) - \log q(X_{\text{unobs}}) \rangle_{q(X_{\text{unobs}})} \\ &= \left\langle \textstyle\sum_i \log f_i(X_{C_i}|\theta_i) - \log Z(\theta) \right\rangle_{q(X_{\text{unobs}})} + \mathbf{H}(q) \\ &= \sum_i \langle \log f_i(X_{C_i}|\theta_i) \rangle_{q(X_{C_i} \setminus X_{\text{obs}})} - \log Z(\theta) + \mathbf{H}(q)\end{aligned}$$

So learning only requires posterior marginals on cliques (obtained by messaging passing) and updates on cliques; c.f. the Baum-Welch procedure for HMMs.

Probabilistic & Unsupervised Learning
Approximate Inference

Bayesian Model selection, Hyperparameter optimisation, and Gaussian Processes

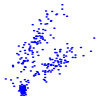
Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

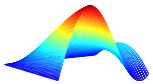
Term 1, Autumn 2022

Learning model structure

How many clusters in the data?

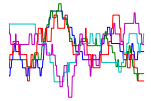


How smooth should the function be?



Is this input relevant to predicting that output?

What is the order of a dynamical system?



How many states in a hidden Markov model?

SVYDAAAQLTADVKKDLRDSWKVIGSDKKGNG

How many auditory sources in the input?

Model selection

Models (labelled by m) have parameters $\boldsymbol{\theta}_m$ that specify the probability of data:

$$P(\mathcal{D}|\boldsymbol{\theta}_m, m)\,.$$

If model is known, learning $\boldsymbol{\theta}_m$ means finding posterior or point estimate (ML, MAP, ...).

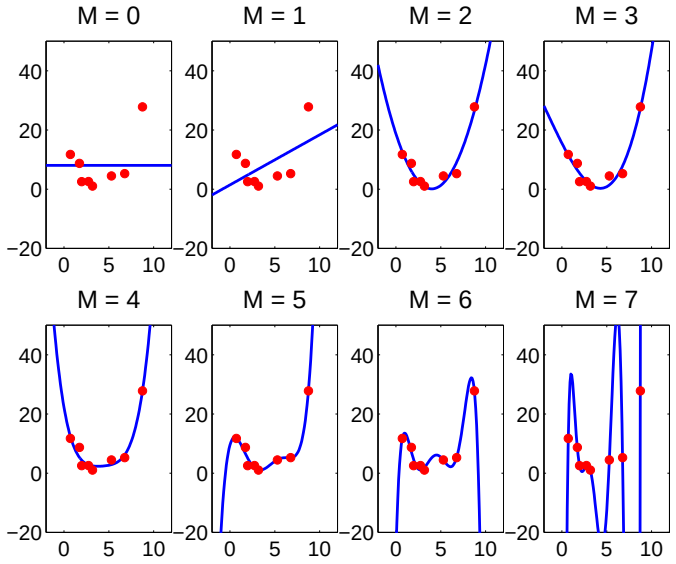
What if we need to learn the model too?

- Could combine models into a single "supermodel", with composite parameter $(m, \boldsymbol{\theta}_m)$.
  - ML learning will overfit: favours most flexible (nested) model with most parameters, even if the data actually come from a simpler one.
  - Density function on composite parameter space (union of manifolds of different dimensionalities) difficult to define $\Rightarrow$ MAP learning ill-posed.
  - Joint posterior difficult to compute — dimension of composite parameter varies [although Monte-Carlo methods may be able to sample from such a posterior.]

$\Rightarrow$ Separate model selection step:

$$P(\boldsymbol{\theta}_m, m|\mathcal{D}) = \underbrace{P(\boldsymbol{\theta}_m|m,\mathcal{D})}_{\text{model-specific posterior}} \cdot \underbrace{P(m|\mathcal{D})}_{\text{model selection}}$$

Model complexity and overfitting: a simple example



Model selection

Given models labeled by m with parameters $\boldsymbol{\theta}_m$, identify the "correct" model for data $\mathcal{D}$.

ML/MAP has no good answer: $P(\mathcal{D}|\boldsymbol{\theta}_m^{\mathrm{ML}})$ is always larger for more complex (nested) models.

Neyman-Pearson hypothesis testing

- For nested models. Starting with simplest model ($m = 1$), compare (e.g. by likelihood ratio test) null hypothesis m to alternative $m + 1$. Continue until $m + 1$ is rejected.
- Tests often only exact asympotically in data number.
- Conservative (N-P hypothesis tests are asymmetric by design).

Likelihood validation

- Partition data into disjoint *training* and *validation* data sets $\mathcal{D} = \mathcal{D}_{\mathrm{tr}} \cup \mathcal{D}_{\mathrm{vld}}$. Choose model with greatest $P(\mathcal{D}_{\mathrm{vld}}|\boldsymbol{\theta}_m^{\mathrm{ML}})$, with $\boldsymbol{\theta}_m^{\mathrm{ML}} = \operatorname{argmax} P(\mathcal{D}_{\mathrm{tr}}|\boldsymbol{\theta})$. [Or, better, greatest $P(\mathcal{D}_{\mathrm{vld}}|\mathcal{D}_{\mathrm{tr}}, m)$.]
- Consistent; and selects most useful model, even if all are incorrect.
- May be biased towards simpler models; often high-variance.
- Cross-validation uses multiple partitions and averages likelihoods.

Bayesian model selection

- Choose most likely model: $\operatorname{argmax} P(m|\mathcal{D})$.
- Consistent; probabilistically principled if true model is in set being considered, but sensitive to assumed priors etc.
- Posterior probabilities can weight models for combined predictions (avoiding selection).

Bayesian model selection: some terminology

A model class m is a set of distributions parameterised by $\boldsymbol{\theta}_m$, e.g. the set of all possible mixtures of m Gaussians.

The model implies both a prior over the parameters $P(\boldsymbol{\theta}_m|m)$, and a likelihood of data given parameters (which might require integrating out latent variables) $P(\mathcal{D}|\boldsymbol{\theta}_m, m)$.

The posterior distribution over parameters is

$$P(\boldsymbol{\theta}_m|\mathcal{D}, m) = \frac{P(\mathcal{D}|\boldsymbol{\theta}_m, m)P(\boldsymbol{\theta}_m|m)}{P(\mathcal{D}|m)}.$$

The marginal probability of the data under model class m is:

$$P(\mathcal{D}|m) = \int_{\Theta_m} P(\mathcal{D}|\boldsymbol{\theta}_m, m)P(\boldsymbol{\theta}_m|m)\, d\boldsymbol{\theta}_m.$$

(also called the Bayesian evidence for model m).

The ratio of two marginal probabilities (or sometimes its log) is known as the Bayes factor:

$$\frac{P(\mathcal{D}|m)}{P(\mathcal{D}|m')} = \frac{P(m|\mathcal{D})}{P(m'|\mathcal{D})}\frac{p(m')}{p(m)}$$

The Bayesian Occam's razor

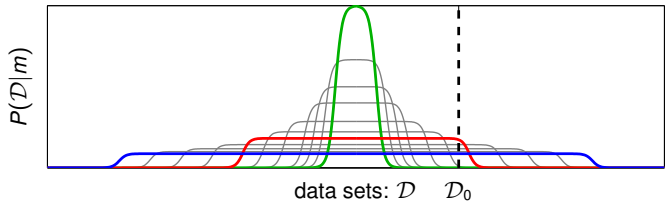
Occam's Razor is a principle of scientific philosophy: of two explanations adequate to explain the same set of observations, the simpler should always be preferred.
Bayesian inference formalises and *automatically* implements a form of Occam's Razor.

Compare model classes m using their posterior probability given the data:

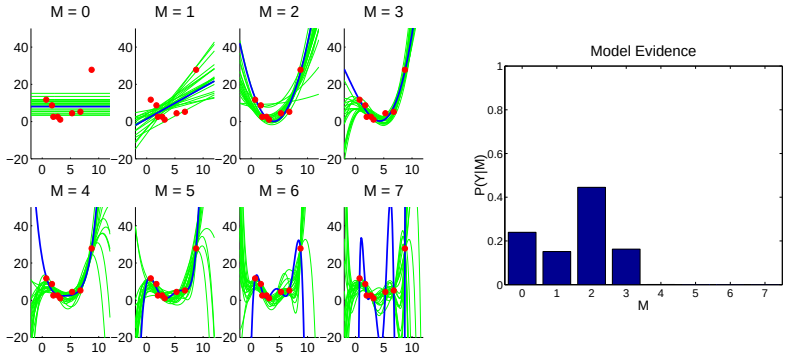
$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}, \qquad P(\mathcal{D}|m) = \int_{\Theta_m} P(\mathcal{D}|\boldsymbol{\theta}_m, m)P(\boldsymbol{\theta}_m|m)\, d\boldsymbol{\theta}_m$$

$P(\mathcal{D}|m)$: The probability that *randomly selected* parameter values from the model class would generate data set $\mathcal{D}$.

Model classes that are too simple are unlikely to generate the observed data set.
Model classes that are too complex can generate many possible data sets, so again, they are unlikely to generate that particular data set at random.
Like Goldilocks, we favour a model that is just right.



Bayesian model comparison: Occam's razor at work



Conjugate-exponential families (recap)

Can we compute $P(\mathcal{D}|m)$? ..... Sometimes.

Suppose $P(\mathcal{D}|\boldsymbol{\theta}_m, m)$ is a member of the exponential family:

$$P(\mathcal{D}|\boldsymbol{\theta}_m, m) = \prod_{i=1}^{N} P(\mathbf{x}_i|\boldsymbol{\theta}_m, m) = \prod_{i=1}^{N} e^{\mathbf{s}(\mathbf{x}_i)^\mathsf{T}\boldsymbol{\theta}_m - A(\boldsymbol{\theta}_m)}.$$

If our prior on $\boldsymbol{\theta}_m$ is conjugate:

$$P(\boldsymbol{\theta}_m|m) = e^{\mathbf{s}_p^\mathsf{T}\boldsymbol{\theta}_m - n_p A(\boldsymbol{\theta}_m)} / Z(\mathbf{s}_p, n_p)$$

then the joint is in the same family:

$$P(\mathcal{D}, \boldsymbol{\theta}_m|m) = e^{\left(\sum_i \mathbf{s}(\mathbf{x}_i) + \mathbf{s}_p\right)^\mathsf{T}\boldsymbol{\theta}_m - (N+n_p)A(\boldsymbol{\theta}_m)} / Z(\mathbf{s}_p, p)$$

and so:

$$P(\mathcal{D}|m) = \int d\boldsymbol{\theta}_m \; P(\mathcal{D}, \boldsymbol{\theta}_m|m) = \frac{Z\left(\sum_i \mathbf{s}(\mathbf{x}_i) + \mathbf{s}_p, N + n_p\right)}{Z(\mathbf{s}_p, p)} \quad \begin{matrix} \leftarrow \text{ posterior volume} \\ \leftarrow \text{ prior volume} \end{matrix}$$

While the intuition is general, tractability is a special case. Thus, we must approximate ...

Practical Bayesian approaches

- Laplace approximation:
  - Approximate posterior by a Gaussian centred at the maximum *a posteriori* parameter estimate.
- Bayesian Information Criterion (BIC)
  - an asymptotic ($N \to \infty$) approximation.
- Variational Bayes
  - Lower bound on the marginal probability.
  - Biased estimate.
  - Easy and fast, and often better than Laplace or BIC.
- Monte Carlo methods:
  - (Annealed) Importance sampling: estimate evidence using samples $\boldsymbol{\theta}^{(i)}$ from arbitrary $f(\boldsymbol{\theta})$:
    $$\sum_i \frac{P(\mathcal{D}|\boldsymbol{\theta}^{(i)}, m) P(\boldsymbol{\theta}^{(i)}|m)}{f(\boldsymbol{\theta}^{(i)})} \to \int d\boldsymbol{\theta}\, f(\boldsymbol{\theta}) \frac{P(\mathcal{D}, \boldsymbol{\theta}|m)}{f(\boldsymbol{\theta})} = P(\mathcal{D}|m)$$
  - "Reversible jump" Markov Chain Monte Carlo: sample from posterior on composite $(m, \boldsymbol{\theta}_m)$. # samples for each $m \propto p(m|\mathcal{D})$.
  - Both exact in the limit of infinite samples, but may have high variance with finite samples.
- Bethe approximations, Expectation propagation ...

We will encounter many of these approaches later in the course.

Laplace approximation

We want to find $P(\mathcal{D}|m) = \int P(\mathcal{D}, \boldsymbol{\theta}_m|m)\; d\boldsymbol{\theta}_m$.

As data size N grows (relative to parameter count d), $\boldsymbol{\theta}_m$ becomes more constrained $\Rightarrow P(\mathcal{D}, \boldsymbol{\theta}_m|m) \propto P(\boldsymbol{\theta}_m|\mathcal{D}, m)$ becomes concentrated on posterior mode $\boldsymbol{\theta}_m^*$.

Idea: approximate $\log P(\mathcal{D}, \boldsymbol{\theta}_m|m)$ to second-order around $\boldsymbol{\theta}^*$.

$$\begin{aligned}
\int P(\mathcal{D}, \boldsymbol{\theta}_m|m) d\boldsymbol{\theta}_m &= \int e^{\log P(\mathcal{D}, \boldsymbol{\theta}_m|m)}\; d\boldsymbol{\theta}_m \\
&\approx \int e^{\log P(\mathcal{D}, \boldsymbol{\theta}_m^*|m) + \underbrace{\nabla \log P(\mathcal{D}, \boldsymbol{\theta}_m^*|m)}_{=0} \cdot (\boldsymbol{\theta}_m - \boldsymbol{\theta}_m^*) + \frac{1}{2}(\boldsymbol{\theta}_m - \boldsymbol{\theta}_m^*)^\mathsf{T} \underbrace{\nabla^2 \log P(\mathcal{D}, \boldsymbol{\theta}^*|m)}_{=-A} (\boldsymbol{\theta}_m - \boldsymbol{\theta}_m^*)}\; d\boldsymbol{\theta}_m \\
&= \int P(\mathcal{D}, \boldsymbol{\theta}_m^*|m) e^{-\frac{1}{2}(\boldsymbol{\theta}_m - \boldsymbol{\theta}_m^*)^\mathsf{T} A (\boldsymbol{\theta}_m - \boldsymbol{\theta}_m^*)}\; d\boldsymbol{\theta}_m \\
&= P(\mathcal{D}|\boldsymbol{\theta}_m^*, m) P(\boldsymbol{\theta}_m^*|m) (2\pi)^{\frac{d}{2}} |A|^{-\frac{1}{2}}
\end{aligned}$$

$A = -\nabla^2 \log P(\mathcal{D}, \boldsymbol{\theta}_m^*|m)$ is the negative Hessian of $\log P(\mathcal{D}, \boldsymbol{\theta}|m)$ evaluated at $\boldsymbol{\theta}_m^*$.

This is equivalent to approximating the posterior by a Gaussian: an approximation which is asymptotically correct.

Bayesian Information Criterion (BIC)

BIC can be obtained from the Laplace approximation:

$$\log P(\mathcal{D}|m) \approx \log P(\boldsymbol{\theta}_m^*|m) + \log P(\mathcal{D}|\boldsymbol{\theta}_m^*, m) + \frac{d}{2}\log 2\pi - \frac{1}{2}\log |A|$$

We have

$$A = -\nabla^2 \log P(\mathcal{D}, \boldsymbol{\theta}^*|m) = -\nabla^2 \log P(\mathcal{D}|\boldsymbol{\theta}^*, m) - \nabla^2 \log P(\boldsymbol{\theta}^*|m)$$

So as $N = |\mathcal{D}| \to \infty$, $A \to NA_0 + \text{constant}$, for fixed PD matrix $A_0 = \left\langle -\nabla^2 \log P(\mathbf{x}|\theta^*, m) \right\rangle$.
$\Rightarrow \log |A| \to \log |NA_0| = \log(N^d |A_0|) = d \log N + \log |A_0|$.

Retaining only terms that grow with N we get:

$$\log P(\mathcal{D}|m) \approx \log P(\mathcal{D}|\boldsymbol{\theta}_m^*, m) - \frac{d}{2}\log N$$

Properties:

- Quick and easy to compute.
- Does not depend on prior.
- We can use the ML estimate of θ instead of the MAP estimate (= as $N \to \infty$).
- Related to the "Minimum Description Length" (MDL) criterion (Asst 2).
- Assumes that in the large sample limit, all the parameters are well-determined (i.e. the model is identifiable; otherwise, d should be the number of well-determined parameters).
- Neglects multiple modes (e.g. permutations in a MoG).

Hyperparameters and Evidence optimisation

In some cases, we need to choose between a family of continuously parameterised models.

$$P(\mathcal{D}|\eta) = \int P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\eta)\, d\boldsymbol{\theta}$$

η: hyperparameters

This choice can be made by ascending the gradient in:

- the exact evidence (if tractable).
- the approximated evidence (Laplace, EP, Bethe, ...)
- a free-energy bound on the evidence (Variational Bayes)

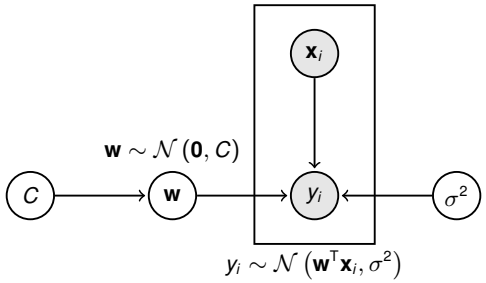
or by placing a hyperprior on the hyperparameters η, and sampling from the posterior

$$P(\eta|\mathcal{D}) = \frac{P(\mathcal{D}|\eta)P(\eta)}{P(\mathcal{D})}$$

using Markov chain Monte Carlo sampling.

Evidence optimisation in linear regression

Consider simple linear regression:



- Maximize

$$P(y_1 \dots y_N|\mathbf{x}_1 \dots \mathbf{x}_N, C, \sigma^2) = \int d\mathbf{w}\, P(y_1 \dots y_N|\mathbf{x}_1 \dots \mathbf{x}_N, \mathbf{w}, \sigma^2)P(\mathbf{w}|C)$$
$$= \int d\mathbf{w} \prod_i P(y_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)P(\mathbf{w}|C)$$

  to find optimal values of C, σ.
- Compute the posterior $P(\mathbf{w}|y_1 \dots y_N, \mathbf{x}_1 \dots \mathbf{x}_N, C, \sigma^2)$ given these optimal values.

The evidence for linear regression

- The posterior on $\mathbf{w}$ is normal: $\Sigma_\mathbf{w} = (\frac{XX^\mathsf{T}}{\sigma^2} + C^{-1})^{-1}$; $\bar{\mathbf{w}} = \Sigma_\mathbf{w}\frac{XY^\mathsf{T}}{\sigma^2}$.
  Note: X is a matrix where columns are input vectors, and Y is a row vector of corresponding predicted outputs.
- The evidence, $\mathcal{E}(C, \sigma^2) = \int P(Y|X, \mathbf{w}, \sigma^2)P(\mathbf{w}|C)\, d\mathbf{w}$, is given by:

$$\mathcal{E}(C, \sigma^2) = \sqrt{\frac{|2\pi\Sigma_\mathbf{w}|}{|2\pi\sigma^2 I|\,|2\pi C|}} \exp\left(-\frac{1}{2}Y\left(\frac{I}{\sigma^2} - \frac{X^\mathsf{T}\Sigma_\mathbf{w}X}{\sigma^4}\right)Y^\mathsf{T}\right)$$

- For optimization, general forms for the gradients are available. If θ is a parameter in C:

$$\frac{\partial}{\partial\theta}\log\mathcal{E}(C, \sigma^2) = \frac{1}{2}\mathrm{Tr}\left[(C - \Sigma_\mathbf{w} - \bar{\mathbf{w}}\bar{\mathbf{w}}^\mathsf{T})\frac{\partial}{\partial\theta}C^{-1}\right]$$
$$\frac{\partial}{\partial\sigma^2}\log\mathcal{E}(C, \sigma^2) = \frac{1}{\sigma^2}\left(-N + \mathrm{Tr}\left[I - \Sigma_\mathbf{w}C^{-1}\right] + \frac{1}{\sigma^2}(Y - \bar{\mathbf{w}}^\mathsf{T}X)(Y - \bar{\mathbf{w}}^\mathsf{T}X)^\mathsf{T}\right)$$

Automatic Relevance Determination

The most common form of evidence optimization for regression (due to MacKay and Neal) takes $C^{-1} = \mathrm{diag}(\boldsymbol{\alpha})$ (i.e. $w_i \sim \mathcal{N}(0, \alpha_i^{-1})$) and then optimizes the precisions $\{\alpha_i\}$.

Setting the gradients to 0 and solving gives

$$\alpha_i^{\mathrm{new}} = \frac{1 - \alpha_i[\Sigma_\mathbf{w}]_{ii}}{\bar{\mathbf{w}}_i^2}$$
$$(\sigma^2)^{\mathrm{new}} = \frac{(Y - \bar{\mathbf{w}}^\mathsf{T}X)(Y - \bar{\mathbf{w}}^\mathsf{T}X)^\mathsf{T}}{N - \sum_i(1 - [\Sigma_\mathbf{w}]_{ii}\alpha_i)}$$

During optimization the α_is meet one of two fates

$\alpha_i \to \infty$	$\Rightarrow$	$w_i = 0$	irrelevant input x_i
α_i finite	$\Rightarrow w_i = \mathrm{argmax}\, \mathrm{P}(w_i \mid X, Y, \alpha_i)$		relevant input x_i

This procedure, Automatic Relevance Determination (ARD), yields sparse solutions that improve on ML regression. (cf. L_1-regression or LASSO).

Evidence optimisation is also called maximum marginal likelihood or ML-2 (Type 2 maximum likelihood).

Prediction averaging

Sometimes, our goal is not to learn the model structure or parameters (or their posteriors); but rather to predict the (conditional) density at a new data point.

The Bayesian approach in this case should integrate out the parameters:

- Density (unsupervised learning): $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$

$$p(\mathbf{x}|\mathcal{D}, m) = \int d\theta\, p(\mathbf{x}|\theta, m) p(\theta|\mathcal{D}, m)$$

- Predictions (supervised learning): $\mathcal{D} = \{(\mathbf{x}_1, y_1)(\mathbf{x}_2, y_2) \ldots (\mathbf{x}_n, y_n)\}$

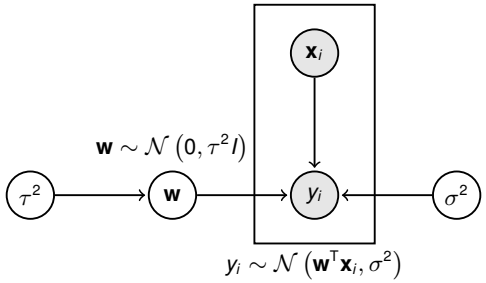
$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}, m) = \int d\theta\, p(\mathbf{y}|\mathbf{x}, \theta, m) p(\theta|\mathcal{D}, m)$$

The integral naturally favours predictions associated with large volumes in the posterior, and so incorporates an Occam-like factor.

In principle, predictions may resist overfitting even with an infinitely complex model [or, put another way, the marginalised model has no parameters] $\Rightarrow$ Bayesian nonparametrics.

Taking this approach to (non)linear regression leads to a powerful supervised learning method called Gaussian process regression.

Prediction averaging in linear regression



Let $X = [\mathbf{x}_1 \ldots \mathbf{x}_N]$, $Y = [y_1 \ldots y_N]$. Then (as we've seen)

$$\mathbf{w}|\mathcal{D} \sim \mathcal{N}(\bar{\mathbf{w}}, \Sigma_\mathbf{w})$$

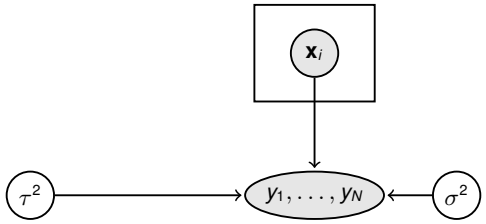
where $\Sigma_\mathbf{w} = (\frac{1}{\sigma^2} XX^\mathsf{T} + \frac{1}{\tau^2} I)^{-1}$ and $\bar{\mathbf{w}} = \frac{1}{\sigma^2} \Sigma_\mathbf{w} XY^\mathsf{T}$

Thus, given a new input vector $\mathbf{x}$, the predicted output y (integrating out $\mathbf{w}$) is:

$$y|\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{w}}^\mathsf{T}\mathbf{x}, \mathbf{x}^\mathsf{T}\Sigma_\mathbf{w}\mathbf{x} + \sigma^2)\,.$$

Added variance $\mathbf{x}^\mathsf{T}\Sigma_\mathbf{w}\mathbf{x}$ comes from posterior uncertainty in $\mathbf{w}$ (cf Factor Analysis).

Marginalised linear regression



$$Y^\mathsf{T} \sim \mathcal{N}(\mathbf{0}, \tau^2 X^\mathsf{T} X + \sigma^2 I)$$

Integrate out $\mathbf{w}$ in the model: the joint distribution of $y_1, \ldots, y_N$ given $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is Gaussian. The means and covariances are:

$$E[y_i] = E[\mathbf{w}^\mathsf{T}\mathbf{x}_i] = 0^\mathsf{T}\mathbf{x}_i = 0$$
$$E[(y_i - \overline{y}_i)^2] = E[(\mathbf{x}_i^\mathsf{T}\mathbf{w})(\mathbf{w}^\mathsf{T}\mathbf{x}_i)] + \sigma^2 = \tau^2 \mathbf{x}_i^\mathsf{T}\mathbf{x}_i + \sigma^2$$
$$E[(y_i - \overline{y}_i)(y_j - \overline{y}_j)] = E[(\mathbf{x}_i^\mathsf{T}\mathbf{w})(\mathbf{w}^\mathsf{T}\mathbf{x}_j)] = \tau^2 \mathbf{x}_i^\mathsf{T}\mathbf{x}_j$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \Bigg| \mathbf{x}_1, \ldots, \mathbf{x}_N \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \tau^2\mathbf{x}_1^\mathsf{T}\mathbf{x}_1 + \sigma^2 & \tau^2\mathbf{x}_1^\mathsf{T}\mathbf{x}_2 & \cdots & \tau^2\mathbf{x}_1^\mathsf{T}\mathbf{x}_N \\ \tau^2\mathbf{x}_2^\mathsf{T}\mathbf{x}_1 & \tau^2\mathbf{x}_2^\mathsf{T}\mathbf{x}_2 + \sigma^2 & & \tau^2\mathbf{x}_2^\mathsf{T}\mathbf{x}_N \\ \vdots & & \ddots & \vdots \\ \tau^2\mathbf{x}_N^\mathsf{T}\mathbf{x}_1 & \tau^2\mathbf{x}_N^\mathsf{T}\mathbf{x}_2 & \cdots & \tau^2\mathbf{x}_N^\mathsf{T}\mathbf{x}_N + \sigma^2 \end{bmatrix} \right)$$

Predictions with marginalised regression

Now, include the test input vector $\mathbf{x}$ and test output y:

$$\begin{bmatrix} Y^\mathsf{T} \\ y \end{bmatrix} \Bigg| X, \mathbf{x} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \tau^2 X^\mathsf{T}X + \sigma^2 I & \tau^2 X^\mathsf{T}\mathbf{x} \\ \tau^2 \mathbf{x}^\mathsf{T}X & \tau^2 \mathbf{x}^\mathsf{T}\mathbf{x} + \sigma^2 \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \widetilde{K}_{XX} & \widetilde{K}_{X\mathbf{x}} \\ \widetilde{K}_{\mathbf{x}X} & \widetilde{K}_{\mathbf{xx}} \end{bmatrix} \right)$$

We can find $y|Y$ by the standard multivariate Gaussian result:

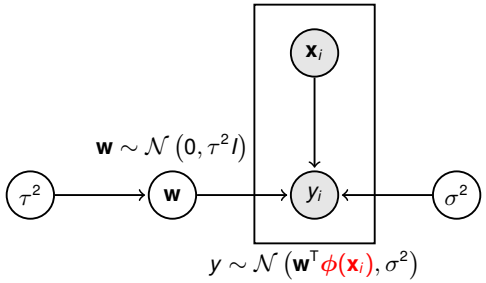
$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{bmatrix} \right) \quad \Rightarrow \quad \mathbf{b}|\mathbf{a} \sim \mathcal{N}\left(K_{BA}K_{AA}^{-1}\mathbf{a}, K_{BB} - K_{BA}K_{AA}{}^{-1}K_{AB} \right)$$

So

$$\begin{aligned} y|Y, X, \mathbf{x} &\sim \mathcal{N}\left(\widetilde{K}_{\mathbf{x}X}\widetilde{K}_{XX}^{-1}Y^\mathsf{T}, \widetilde{K}_{\mathbf{xx}} - \widetilde{K}_{\mathbf{x}X}\widetilde{K}_{XX}^{-1}\widetilde{K}_{X\mathbf{x}} \right) \\ &\sim \mathcal{N}\left(\tau^2\mathbf{x}^\mathsf{T}X(\tau^2 X^\mathsf{T}X + \sigma^2 I)^{-1}Y^\mathsf{T}, \tau^2\mathbf{x}^\mathsf{T}\mathbf{x} + \sigma^2 - \tau^2\mathbf{x}^\mathsf{T}X(\tau^2 X^\mathsf{T}X + \sigma^2 I)^{-1}\tau^2 X^\mathsf{T}\mathbf{x} \right) \\ &\sim \mathcal{N}\Big(\mathbf{x}^\mathsf{T} \underbrace{\tfrac{1}{\sigma^2}\Sigma XY^\mathsf{T}}_{\bar{\mathbf{w}}}, \mathbf{x}^\mathsf{T}\Sigma\mathbf{x} + \sigma^2 \Big) \qquad \Sigma = \underbrace{(\tfrac{1}{\sigma^2}XX^\mathsf{T} + \tfrac{1}{\tau^2}I)^{-1}}_{\Sigma_\mathbf{w}} \qquad \text{[Matrix Inv. Lem.]} \end{aligned}$$

- Same answer as obtained by integrating wrt posterior over $\mathbf{w}$.
- Evidence $P(Y|X)$ is just joint Gaussian probability; reduces to previous expression.
- Thus, Bayesian linear regression can be derived from a joint, parameter-free distribution on all the outputs conditioned on all the inputs.

Nonlinear regression



Introduce nonlinear mapping $\mathbf{x} \mapsto \phi(\mathbf{x})$. Each element of $\phi(\mathbf{x})$ is a (nonlinear) feature extracted from $\mathbf{x}$. May be many more features than elements in $\mathbf{x}$.

The regression function $f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x})$ is nonlinear, but outputs Y are still jointly Gaussian:

$$Y^\mathsf{T}|X \sim \mathcal{N}(0_N, \tau^2\Phi^\mathsf{T}\Phi + \sigma^2 I_N)$$

where the i^{th} column of matrix Φ is $\phi(\mathbf{x}_i)$.
Proceeding as before, the predictive distribution over y for a test input $\mathbf{x}$ is:

$$y|\mathbf{x}, Y, X \sim \mathcal{N}\left(\widetilde{K}_{\mathbf{x}X}\widetilde{K}_{XX}^{-1}Y^\mathsf{T}, \widetilde{K}_{\mathbf{xx}} - \widetilde{K}_{\mathbf{x}X}\widetilde{K}_{XX}^{-1}\widetilde{K}_{X\mathbf{x}}\right)$$

where, now $\widetilde{K}_{XX} = \tau^2\Phi^\mathsf{T}\Phi + \sigma^2 I$; $\widetilde{K}_{X\mathbf{x}} = \tau^2\Phi^\mathsf{T}\phi(\mathbf{x})$ and $\widetilde{K}_{\mathbf{xx}} = \tau^2\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}) + \sigma^2$.

Regression using the covariance kernel

For Bayesian regression, prediction depends on $\widetilde{K}(\mathbf{x}, \mathbf{x})$ rather than explicitly on $\phi(\mathbf{x})$.

So we can define the joint in terms of $\widetilde{K}$ *implicitly* using a (potentially infinite-dimensional) feature map $\phi(\mathbf{x})$.

$$Y|X, \widetilde{K} \sim \mathcal{N}(0_N, \widetilde{K}_{XX}))$$

where the i, j entry in the covariance matrix $\widetilde{K}_{XX}$ is $\widetilde{K}(\mathbf{x}_i, \mathbf{x}_j)$.

This is the **kernel trick**.

Prediction: compute the predictive distribution of y conditioned on Y as before:

$$y|\mathbf{x}, X, Y, \widetilde{K} \sim \mathcal{N}\left(\widetilde{K}_{X\mathbf{x}}^\mathsf{T}\widetilde{K}_{XX}^{-1}Y^\mathsf{T}, \widetilde{K}_{\mathbf{xx}} - \widetilde{K}_{X\mathbf{x}}^\mathsf{T}\widetilde{K}_{XX}^{-1}\widetilde{K}_{X\mathbf{x}}\right)$$

where now $[\widetilde{K}_{XX}]_{ij} = \widetilde{K}(\mathbf{x}_i, \mathbf{x}_j)$; $[\widetilde{K}_{X\mathbf{x}}]_i = \widetilde{K}(\mathbf{x}_i, \mathbf{x})$ and $\widetilde{K}_{\mathbf{xx}} = \widetilde{K}(\mathbf{x}, \mathbf{x})$.

Evidence: given by the Gaussian likelihood:

$$P(Y|X, \widetilde{K}) = |2\pi\widetilde{K}_{XX}|^{-\frac{1}{2}} e^{-\frac{1}{2}Y\widetilde{K}_{XX}^{-1}Y^\mathsf{T}}$$

Evidence optimisation: the covariance kernel $\widetilde{K}$ often has (hyper)parameters, and these can be optimized by gradient ascent in $\log P(Y|X, \widetilde{K})$.

The covariance kernel

$$Y^\mathsf{T}|X \sim \mathcal{N}\left(\mathbf{0}_N, \tau^2\Phi^\mathsf{T}\Phi + \sigma^2 I_N\right)$$

The covariance of the output vector Y plays a central role in the development of the theory of Gaussian processes.

Define a covariance kernel function $\widetilde{K} : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{R}$ such that if $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$ are two input vectors with corresponding outputs y, y', then

$$\widetilde{K}(\mathbf{x}, \mathbf{x}') = \text{Cov}[y, y'] = E[yy'] - E[y]E[y']$$

In the nonlinear regression example we have $\widetilde{K}(\mathbf{x}, \mathbf{x}') = \tau^2\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}') + \sigma^2\delta_{\mathbf{x}=\mathbf{x}'}$.
Any covariance kernel K has two properties:

- Symmetric: $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$ for all $\mathbf{x}, \mathbf{x}'$.
- Positive semidefinite: the matrix $[K(\mathbf{x}_i, \mathbf{x}_j)]$ formed by any finite set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is positive semidefinite.

Theorem: A covariance kernel $K : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{R}$ is symmetric and positive semidefinite if and only if there is a feature map $\phi : \mathbb{X} \mapsto \mathbb{H}$ such that

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}')$$

The feature space $\mathbb{H}$ might be an infinite-dimensional Hilbert space.

The Gaussian process

A covariance kernel $K(\mathbf{x}, \mathbf{x}')$ (and mean function $m(\mathbf{x})$) defined on a domain $\mathbb{X}$ defines a **Gaussian process** (GP): a stochastic process (ie collection of random variables) on $\mathbb{R}$ indexed by $\mathbf{x} \in \mathbb{X}$, any finite subset of which have (consistent) Gaussian distributions.

Let $f(\mathbf{x})$ be the random variable indexed by $\mathbf{x}$. Then a draw from the whole GP (ie for all $\mathbf{x}$) is a random **function** $f : \mathbb{X} \mapsto \mathbb{R}$. We write

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), K(\cdot, \cdot))$$

where the $(\cdot)$s emphasise that these are all functions.

The GP is defined such that, given a finite list of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the joint distribution of the function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\mathsf{T}$ is:

$$\mathbf{f}|X, K \sim \mathcal{N}(\mathbf{m}, K_{XX})$$

where, as usual, $[K_{XX}]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $[\mathbf{m}]_i = m(\mathbf{x}_i)$. If we enlarge or reduce the set of $\mathbf{x}_i$s then the means and covariance matrices produced by fixed functions marginalise correctly.

For nonlinear regression, $f(\cdot)$ could instead be defined by drawing the weight vector $\mathbf{w} \in \mathbb{H}$ from the prior. But $\mathbb{H}$ may be infinite dimensional $\Rightarrow$ need an infinite-size object to make even a single prediction. In the GP view, each $f(\mathbf{x})$ can be drawn separately.

Regression with Gaussian processes

We seek to learn a function that maps inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$ to outputs $y_1, \ldots, y_N$.

Instead of assuming a specific form, consider a random function drawn from a GP prior:

$$f(\cdot) \sim \mathcal{GP}(0, K(\cdot, \cdot)) .$$

Any function is possible (no restriction on support) but some are (much) more likely *a priori*.

Observations y_i are taken to be noisy versions of the (almost surely continuous) latent $f(\mathbf{x}_i)$:

$$y_i|\mathbf{x}_i, f(\cdot) \sim \mathcal{N}\left(f(\mathbf{x}_i), \sigma^2\right) \qquad [\text{so } Y \sim \mathcal{N}\left(0, \widetilde{K}_{XX}\right) \text{ with } \widetilde{K}_{XX} = K_{XX} + \sigma^2 I]$$

Evidence: given by the multivariate Gaussian likelihood:

$$P(Y|X) = |2\pi(K_{XX} + \sigma^2 I)|^{-\frac{1}{2}} e^{-\frac{1}{2} Y (K_{XX} + \sigma^2 I)^{-1} Y^{\mathsf{T}}}$$

Posterior: on latent f is also a GP:

$$f(\cdot)|X, Y \sim \mathcal{GP}(K_{\cdot X}(K_{XX} + \sigma^2 I)^{-1} Y^{\mathsf{T}}, K(\cdot, \cdot) - K_{\cdot X}(K_{XX} + \sigma^2 I)^{-1} K_{X \cdot}))$$

Predictions: posterior on f, plus observation noise:

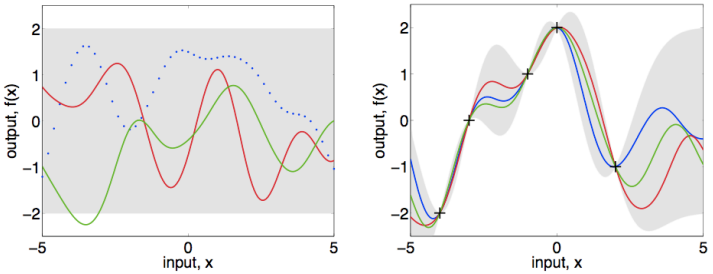
$$y|X, Y, \mathbf{x} \sim \mathcal{N}\left(E[f(\mathbf{x})|X, Y], \mathrm{Var}[f(\mathbf{x})|X, Y] + \sigma^2\right) = \mathcal{N}\left(K_{\mathbf{x}X}\widetilde{K}_{XX}^{-1} Y, K_{\mathbf{x}X}\widetilde{K}_{XX}^{-1} K_{X\mathbf{x}} + \sigma^2\right)$$

Evidence Optimisation: gradient ascent in $\log P(Y|X)$.

Samples from a Gaussian process

We can draw sample functions from a GP by fixing a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, and drawing a sample $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)$ from the corresponding multivariate Gaussian.

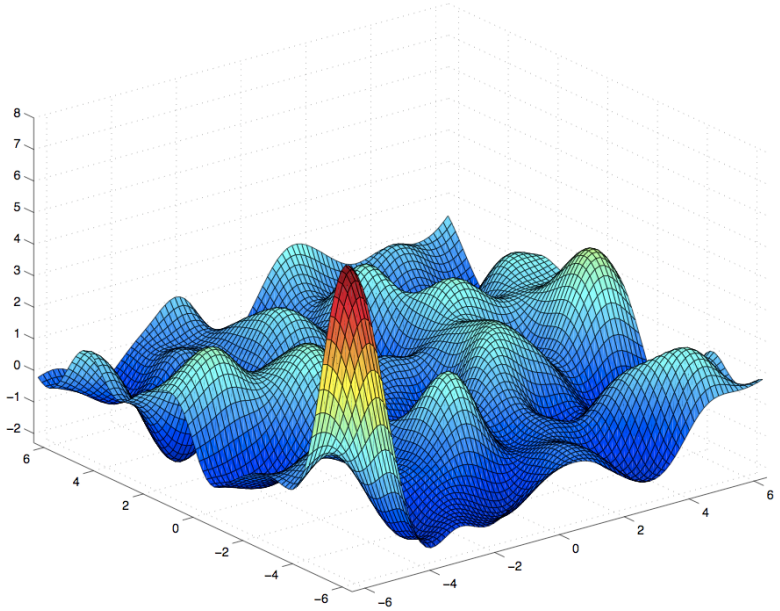
Example prior and posterior GPs:



Another approach is to

- sample $f(\mathbf{x}_1)$ first,
- then $f(\mathbf{x}_2)|f(\mathbf{x}_1)$,
- and generally $f(\mathbf{x}_n)|f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{n-1})$ for $n = 1, 2, \ldots$.

Sample from a 2D Gaussian process



Examples of covariance kernels

• Polynomial:

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^{\mathsf{T}}\mathbf{x}')^m \qquad m = 1, 2, \ldots$$

f is inhomogeneous polynomial of degree m

• Squared-exponential (or exponentiated-quadratic):

$$K(\mathbf{x}, \mathbf{x}') = \theta^2 e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\eta^2}}$$

f is smooth (C^∞ almost surely) on length scale η

• Periodic (exp-sine):

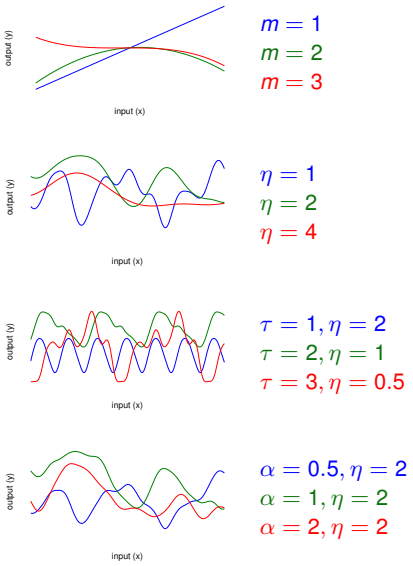
$$K(x, x') = \theta^2 e^{-\frac{2\sin^2(\pi(x - x')/\tau)}{\eta^2}}$$

f is smooth and periodic

• Rational Quadratic:

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\eta^2}\right)^{-\alpha} \qquad \alpha > 0$$

f is smooth over multiple scales



Forms of kernels

If K_1 and K_2 are covariance kernels, then so are:

- Rescaling: αK_1 for $\alpha > 0$.
- Addition: $K_1 + K_2$
- Elementwise product: $K_1 K_2$
- Mapping: $K_1(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ for some function ϕ.

A covariance kernel is translation-invariant if

$$K(\mathbf{x}, \mathbf{x}') = h(\mathbf{x} - \mathbf{x}')$$

A GP with a translation-invariant covariance kernel is stationary: if $f(\cdot) \sim \mathcal{GP}(0, K)$, then so is $f(\cdot - \mathbf{x}) \sim \mathcal{GP}(0, K)$ for each $\mathbf{x}$.

A covariance kernel is radial or radially symmetric if

$$K(\mathbf{x}, \mathbf{x}') = h(\|\mathbf{x} - \mathbf{x}'\|)$$

A GP with a radial covariance kernel is stationary with respect to translations, rotations, and reflections of the input space.

Nonparametric Bayesian Models and Occam's Razor

Overparameterised models can overfit. In the GP, the "parameter" is the function $f(\mathbf{x})$ (or "weights" in non-linear feature space) which can be infinite-dimensional.

However, the Bayesian treatment integrates over these parameters: we never identify a single "best fit" f, just a posterior (and posterior mean). So f cannot be adjusted to overfit the data.

The GP is an example of the larger class of **nonparametric Bayesian models**.

- Infinite number of parameters.
- Often constructed as the infinite limit of a nested family of finite models (sometimes equivalent to infinite model averaging).
- Parameters integrated out, so effective number of parameters to overfit is zero or small (hyperparameters).
- No need for model selection. Bayesian posterior on parameters will concentrate on "submodel" with largest integral automatically.
- No explicit need for Occam's razor, validation or added regularisation penalty.
- Examples include the Dirichlet process (infinite mixtures), Infinite Binary Prior (infinite binary factor models), Infinite HMM ...

GP methods

- With suitable kernels, combinations of kernels, and hyperparameter learning, GPs can identify a wide range of functional dependence. (The "automated statistician" project starts with GPs).
- With approximation, the mapping from f to y may be taken to be non-Gaussian, allowing GP classication, ordinal regression, domain-specific noise and more.
- Functions in more complex hierarchical models may be drawn from GP priors:
  - GP latent variable model (GPLVM)
  - Stacked GPs
  - Deep GP networks
- Inference and learning require inversion of K_{XX}: scales as N^3. Sparse approximate methods reduce this to order N.
- State-of-the-art approach, particularly when data are limited.

Probabilistic & Unsupervised Learning
Approximate Inference

Beyond linear-Gaussian models and Mixtures

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2022

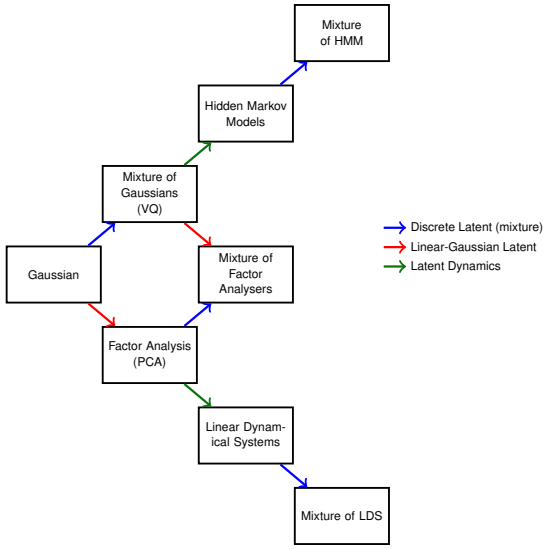
Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.
- Linear regression, Gaussian processes.
- Mixture of Gaussians, mixture of experts.
- Hidden Markov models, linear-Gaussian state space models.

Models consisting of various combinations of:

- Linear Gaussian,
- Discrete variables,
- Chains and trees (or junction trees),

A Generative Model for Generative Models



Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

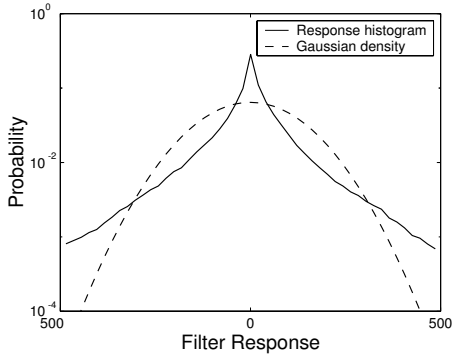
- hierarchical models,
- distributed models,
- nonlinear models,
- non-Gaussian models.

and various combinations of these.

Whilst sometimes tractable (particularly in corner cases), these models will most often require approximate inference.

Why We Need . . . Nonlinear/Non-Gaussian Models

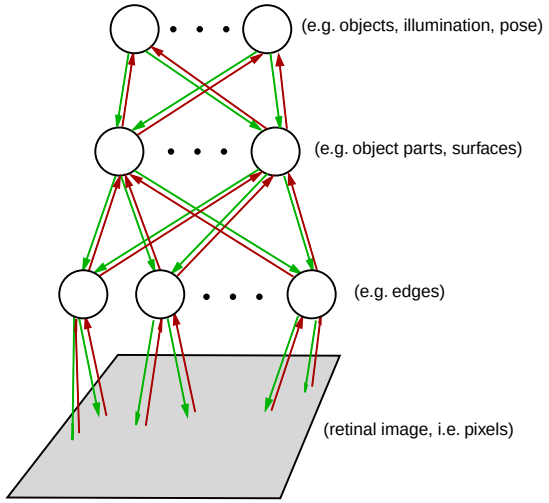
Much of the world is neither linear nor Gaussian



. . . and most interesting structure we would like to learn about is not either.

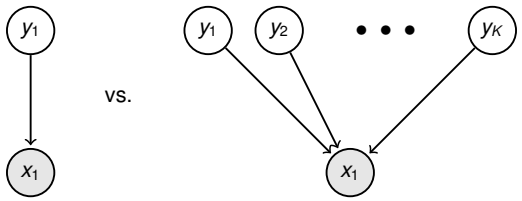
Why We Need . . . Hierarchical (Deep) Models

Many generative processes can be naturally described at different levels of detail.



Biology seems to have developed hierarchical representations.

Why We Need . . . Distributed Models

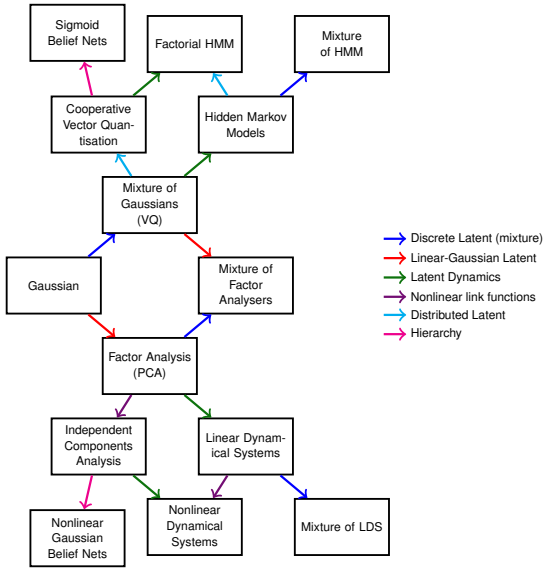


In a distributed representation each observation is characterised by a vector of (discrete or continous) attibutes. Some of these attributes might be latent.

- Unitary representation: categorise voters into small groups who (may) vote similarly e.g.: `London-based university professors of Asian descent`.
- Distributed respresentation: consider separate contributions from a group of attributes, e.g.:
  (`Single`, `Black`, `Female`, `34 yrs`, `Urban`, `Liberal`, `£35k p.a.`).
- Attributes resemble factors, but may be discrete or non-Gaussian, and may outnumber observations.

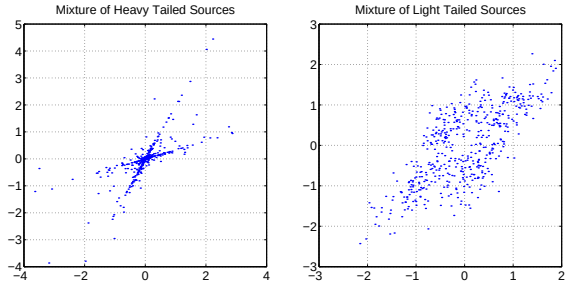
Distributed representations can be exponentially efficient: K binary factors $\Rightarrow$ K bits of info. (K parallel binary state variables in an HMM can replace one variable with 2^K states.)

A Generative Model for Generative Models



Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

Independent Components Analysis

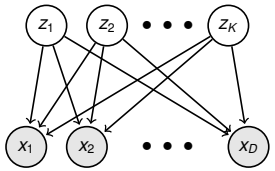


These distributions are generated by linearly combining (or mixing) two *non-Gaussian* sources.

- The ICA graphical model is identical to factor analysis:

$$x_d = \sum_{k=1}^{K} \Lambda_{dk}\, z_k + \epsilon_d$$

but with $z_k \overset{\text{iid}}{\sim} P_z$ non-Gaussian.



- Well-posed even with $K \geq D$ (e.g. $K = D = 2$ above).
- Tractable for 0 noise ("PCA-like" case).
- Intractable in general: posterior non-Gaussian, MAP inference non-linear.
- Exact inference and learning difficult $\Rightarrow$ "noise" components or variational approx.

Square, Noiseless ICA

- The special case of $K = D$, and zero observation noise has been studied extensively (also called infomax ICA, c.f. information view of PCA):

$$\mathbf{x} = \Lambda \mathbf{z} \quad \Rightarrow \quad \mathbf{z} = W\mathbf{x} \quad \text{with} \quad W = \Lambda^{-1}$$

$\mathbf{z}$ are called independent components; W is the unmixing matrix.

- The likelihood can be obtained by transforming the density of $\mathbf{z}$ to that of $\mathbf{x}$. If $F : \mathbf{z} \mapsto \mathbf{x}$ is a differentiable bijection, and if $d\mathbf{z}$ is a small neighbourhood around $\mathbf{z}$, then

$$P_x(\mathbf{x})d\mathbf{x} = P_z(\mathbf{z})d\mathbf{z} = P_z(F^{-1}(\mathbf{x}))\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| d\mathbf{x} = P_z(F^{-1}(\mathbf{x}))\left|\nabla F^{-1}\right| d\mathbf{x}$$

- This gives (for parameter W):

$$P(\mathbf{x}|W) = |W| \prod_k P_z(\underbrace{[W\mathbf{x}]_k}_{z_k})$$

- (A similar idea underlies the more general method of normalising flows, discussed later)

Learning in ICA

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log|W| + \sum_i \log P_z(W_i\mathbf{x})$$

- Learning by gradient ascent:

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) = W^{-T} + g(\mathbf{z})\mathbf{x}^{\mathsf{T}} \qquad g(z) = \frac{\partial \log P_z(z)}{\partial z}$$

- Better approach: "natural" or covariant gradient

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) \cdot \underbrace{(W^{\mathsf{T}}W)} = W + g(\mathbf{z})\mathbf{z}^{\mathsf{T}}W$$
$$\approx \langle -\nabla\nabla \log P \rangle^{-1}$$

(see MacKay 1996).

- Note: we can't use EM in the square noiseless causal ICA model. Why?

Infomax ICA

- Consider a feedforward model:

$$z_i = W_i\mathbf{x}; \qquad \xi_i = f_i(z_i)$$

with a monotonic squashing function $f_i(-\infty) = 0$, $f_i(+\infty) = 1$.

- Infomax finds filtering weights W maximizing the information carried by $\boldsymbol{\xi}$ about $\mathbf{x}$:

$$\underset{W}{\text{argmax}}\, I(\mathbf{x};\boldsymbol{\xi}) = \underset{W}{\text{argmax}}\, H(\boldsymbol{\xi}) - H(\boldsymbol{\xi}|\mathbf{x}) = \underset{W}{\text{argmax}}\, H(\boldsymbol{\xi})$$

Thus we just have to maximize entropy of $\boldsymbol{\xi}$: make it as uniform as possible on $[0, 1]$ (note squashing function).

- But if data were generated from a square noiseless causal ICA then best we can do is if

$$\xi_i = f_i(z_i) = \text{cdf}_i(z_i) \quad \text{and} \quad W = \Lambda^{-1}$$

Infomax ICA $\Leftrightarrow$ square noiseless causal ICA.

- Another view: redundancy reduction in the representation $\boldsymbol{\xi}$ of the data $\mathbf{x}$.

$$\underset{W}{\text{argmax}}\, H(\boldsymbol{\xi}) = \underset{W}{\text{argmax}} \sum_i H(\xi_i) - I(\xi_1, \ldots, \xi_D)$$

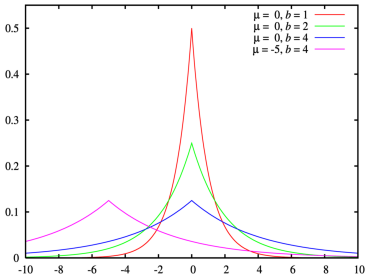
See: MacKay (1996), Pearlmutter and Parra (1996), Cardoso (1997) for equivalence, Teh et al (2003) for an energy-based view.

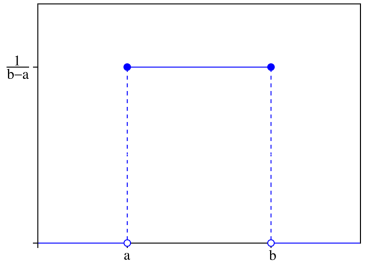
Kurtosis

The kurtosis (or excess kurtosis) measures how “peaky” or “heavy-tailed” a distribution is:

$K = \frac{E((x-\mu)^4)}{E((x-\mu)^2)^2} - 3$, where $\mu = E(x)$ is the mean of x.

Gaussian distributions have zero kurtosis.





Heavy tailed: positive kurtosis (leptokurtic). Light tailed: negative kurtosis (platykurtic).

Linear mixtures of independent non-Gaussian sources tend to be “more” Gaussian $\Rightarrow K \to 0$.

Some ICA algorithms are essentially kurtosis pursuit approaches. Possibly fewer assumptions about generating distributions.

ICA and BSS

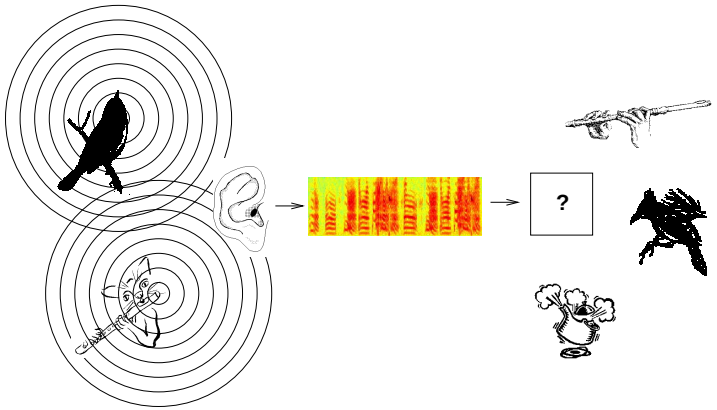
Applications:

- Separating auditory sources
- Analysis of EEG data
- Analysis of functional MRI data
- Natural scene analysis
- . . .

Extensions:

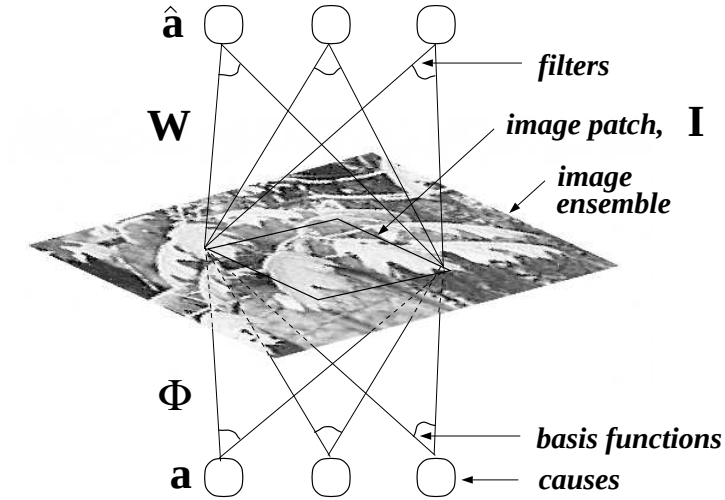
- Non-zero output noise – approximate posteriors and learning.
- Undercomplete ($K < D$) or overcomplete ($K > D$).
- Learning prior distributions (on $\mathbf{z}$).
- Dynamical hidden models (on $\mathbf{z}$).
- Learning number of sources.
- Time-varying mixing matrix.
- Nonparametric, kernel ICA.
- . . .

Blind Source Separation

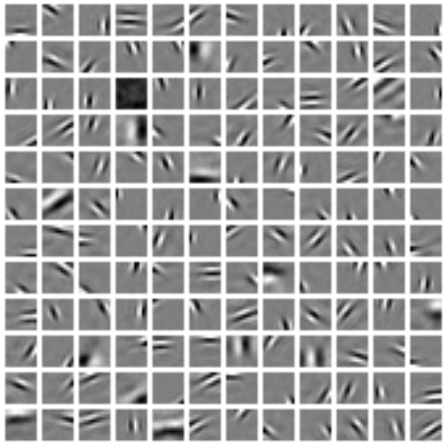


- ICA solution to blind source separation assumes no dependence across time; still works fine much of the time.
- Many other algorithms: DCA, SOBI, JADE, . . .

Images

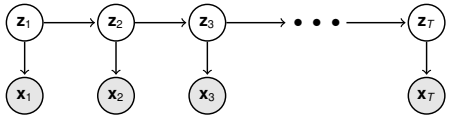


Natural Scenes



Olshausen & Field (1996)

The LGSSM: Kalman Filtering



$$\mathbf{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{z}_t|\mathbf{z}_{t-1} \sim \mathcal{N}(A\mathbf{z}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{z}_t \sim \mathcal{N}(C\mathbf{z}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{z}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{z}_1|\mathbf{x}_1) = \mathcal{N}(\underbrace{\hat{\mathbf{z}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{z}}_i^0)}_{\hat{\mathbf{z}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C \hat{V}_1^0}_{\hat{V}_1^1}) \qquad K_1 = \hat{V}_1^0 C^\mathsf{T}(C\hat{V}_1^0 C^\mathsf{T} + R)^{-1}$$

In general, we define $\hat{\mathbf{z}}_t^\tau \equiv E[\mathbf{z}_t|\mathbf{x}_1, \ldots, \mathbf{x}_\tau]$ and $\hat{V}_t^\tau \equiv V[\mathbf{z}_t|\mathbf{x}_1, \ldots, \mathbf{x}_\tau]$. Then,

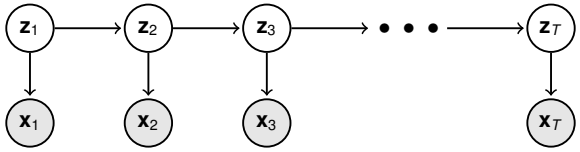
$$P(\mathbf{z}_t|\mathbf{x}_{1:t-1}) = \int d\mathbf{z}_{t-1} P(\mathbf{z}_t|\mathbf{z}_{t-1}) P(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1}) = \mathcal{N}(\underbrace{A\hat{\mathbf{z}}_{t-1}^{t-1}}_{\hat{\mathbf{z}}_t^{t-1}}, \underbrace{A\hat{V}_{t-1}^{t-1}A^\mathsf{T} + Q}_{\hat{V}_t^{t-1}})$$

$$P(\mathbf{z}_t|\mathbf{x}_{1:t}) = \mathcal{N}(\underbrace{\hat{\mathbf{z}}_t^{t-1} + K_t(\mathbf{x}_t - C\hat{\mathbf{z}}_t^{t-1})}_{\hat{\mathbf{z}}_t^t}, \underbrace{\hat{V}_t^{t-1} - K_t C \hat{V}_t^{t-1}}_{\hat{V}_t^t})$$

$$\underbrace{K_t = \overbrace{\hat{V}_t^{t-1}C^\mathsf{T}}^{\langle \mathbf{z}\mathbf{x}^\mathsf{T}\rangle}(\overbrace{C\hat{V}_t^{t-1}C^\mathsf{T} + R}^{\langle \mathbf{x}\mathbf{x}^\mathsf{T}\rangle})^{-1}}_{\text{Kalman gain}}$$

FA: $\beta = (I + \Lambda^\mathsf{T}\Psi^{-1}\Lambda)^{-1}\Lambda^\mathsf{T}\Psi^{-1} \overset{\text{mat. inv. lem.}}{=} \Lambda^\mathsf{T}(\Lambda\Lambda^\mathsf{T} + \Psi)^{-1}$; $\mu = \beta\mathbf{x}_n$; $\Sigma = I - \beta\Lambda$.

The LGSSM: Kalman smoothing



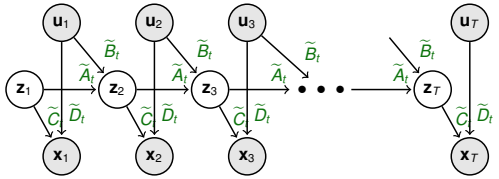
We use a slightly different decomposition:

$$\begin{aligned} P(\mathbf{z}_t|\mathbf{x}_{1:T}) &= \int P(\mathbf{z}_t, \mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1} \\ &= \int P(\mathbf{z}_t|\mathbf{z}_{t+1}, \mathbf{x}_{1:T}) P(\mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1} \\ &\overset{\text{Markov property}}{=} \int P(\mathbf{z}_t|\mathbf{z}_{t+1}, \mathbf{x}_{1:t}) P(\mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1} \end{aligned}$$

This gives the additional **backward recursion**:

$$J_t = \hat{V}_t^t A^\mathsf{T}(\hat{V}_{t+1}^t)^{-1}$$
$$\hat{\mathbf{z}}_t^T = \hat{\mathbf{z}}_t^t + J_t(\hat{\mathbf{z}}_{t+1}^T - A\hat{\mathbf{z}}_t^t)$$
$$\hat{V}_t^T = \hat{V}_t^t + J_t(\hat{V}_{t+1}^T - \hat{V}_{t+1}^t)J_t^\mathsf{T}$$

Nonlinear state-space model (NLSSM)



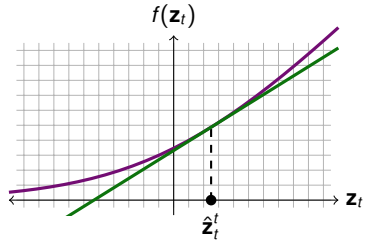
$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{z}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

Extended Kalman Filter (EKF): linearise nonlinear functions about current estimate, $\hat{\mathbf{z}}_t^t$:

$$\mathbf{z}_{t+1} \approx \underbrace{f(\hat{\mathbf{z}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t\mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{z}_t}\right|_{\hat{\mathbf{z}}_t^t}}_{\widetilde{A}_t}(\mathbf{z}_t - \hat{\mathbf{z}}_t^t) + \mathbf{w}_t$$

$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{z}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t\mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{z}_t}\right|_{\hat{\mathbf{z}}_t^{t-1}}}_{\widetilde{C}_t}(\mathbf{z}_t - \hat{\mathbf{z}}_t^{t-1}) + \mathbf{v}_t$$



Run the Kalman filter (smoother) on non-stationary linearised system ($\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t$):

- Adaptively approximates non-Gaussian messages by Gaussians.
- Local linearisation depends on central point of distribution ⇒ approximation degrades with increased state uncertainty. May work acceptably for close-to-linear systems.

Can base EM-like algorithm on EKF/EKS (or alternatives).

Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters :$\bar{\bar{\mathbf{z}}}_t = \begin{bmatrix} \mathbf{z}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{z}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{z}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{z}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{z}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w}_t \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{z}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{z}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{z}_t$$

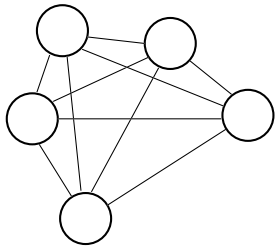
(where A and C need to be vectorised and de-vectorised as appropriate).

Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{z}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{z}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.
- Can model nonstationarity by assuming non-zero innovations noise in A, C.
- Not simple to implement for Q and R (e.g. covariance constraints?).
- May be faster than EM/gradient approaches.

Sometimes called the joint-EKF approach.

Boltzmann Machines



Undirected graphical model (i.e. a Markov network) over a vector of binary variables $s_i \in \{0, 1\}$. Some variables may be hidden, some may be visible (observed).

$$P(\mathbf{s}|W, \mathbf{b}) = \frac{1}{Z} \exp\left\{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i\right\}$$

where Z is the normalization constant (partition function).

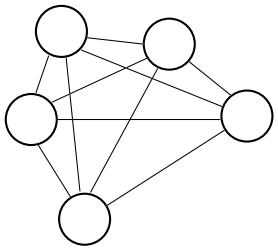
A jointly exponential-family model, with intractable normaliser.

- Inference requires expectations of hidden nodes $\mathbf{s}^H$:

$$\left\langle \mathbf{s}^H \right\rangle_{P(\mathbf{s}^H|\mathbf{s}^V, W, \mathbf{b})} \qquad \left\langle \mathbf{s}^H \mathbf{s}^{H^\mathsf{T}} \right\rangle_{P(\mathbf{s}^H|\mathbf{s}^V, W, \mathbf{b})}$$

- Usually requires approximate methods: sampling or loopy BP.
- Intractable normaliser also complicates M-step $\Rightarrow$ doubly intractable.

Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H|W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

Generalised (gradient M-step) EM requires parameter step

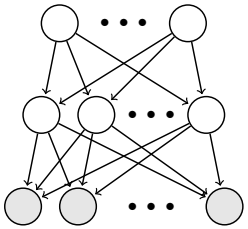
$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \left\langle \log P(\mathbf{s}^V \mathbf{s}^H|W, \mathbf{b}) \right\rangle_{P(\mathbf{s}^H|\mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s_i^v, s^v_{i,obs}}$). Then

$$\begin{aligned}
[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} &= \frac{\partial}{\partial W_{ij}} \left[\textstyle\sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z\right] = \langle s_i s_j \rangle_c - \frac{\partial}{\partial W_{ij}} \log Z \\
&= \langle s_i s_j \rangle_c - \frac{1}{Z} \frac{\partial}{\partial W_{ij}} \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} \frac{1}{Z} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} s_i s_j \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} P(\mathbf{s}|W, \mathbf{b}) s_i s_j \quad = \langle s_i s_j \rangle_c - \langle s_i s_j \rangle_u
\end{aligned}$$

with $\langle\rangle_u$ (unclamped) expectation under the current joint. $\Rightarrow$ ExpFam moment matching, but requires simulation and gradient ascent.

Sigmoid Belief Networks



Directed graphical model (i.e. Bayesian network) over a vector of binary variables $s_i \in \{0, 1\}$.

$$P(\mathbf{s}|W, \mathbf{b}) = \prod_i P(s_i|\{s_j\}_{j<i}, W, \mathbf{b})$$

$$s_i|\{s_j\}_{j<i}, W, \mathbf{b} \sim \text{Bernoulli}(\sigma(\sum_{j<i} W_{ij} s_j - b_i))$$

$$P(s_i = 1|\{s_j\}_{j<i}, W, \mathbf{b}) = \frac{1}{1 + \exp\{-\sum_{j<i} W_{ij} s_j - b_i\}}$$

- parents most often grouped into layers
- logistic function σ of linear combination of parents
- "generative multilayer perceptron" ("neural network")

Learning algorithm: a gradient version of EM

- E step involves computing averages w.r.t. $P(\mathbf{s}^H|\mathbf{s}^V, W, \mathbf{b})$. This could be done either exactly or approximately using Gibbs sampling or mean field approximations. Or using a parallel 'recognition network' (the Helmholtz machine).
- Unlike Boltzmann machines, there is no separate partition function, so no need for an unclamped phase in the M step.

Restricted Boltzmann Machines

Special case Boltzmann Machine: $W_{ij} = 0$ for any two visible or any two hidden nodes (bipartite graph).

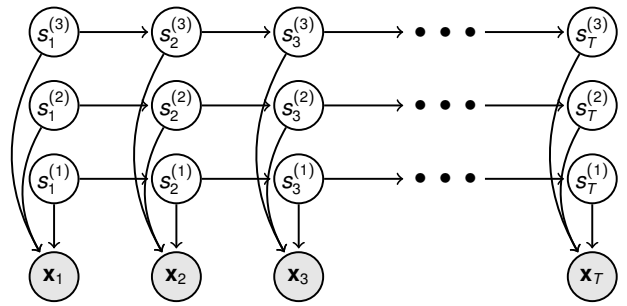
$$
\begin{aligned}
P(\mathbf{s}^V|\mathbf{s}^H) &= \frac{1}{Z} e^{\sum_{i\in V}\sum_{j\in H} W_{ij}s_i s_j - \sum_{i\in V} b_i s_i - \sum_{j\in H} b_j s_j} \\
&= \frac{1}{Z'} \prod_i e^{s_i \sum_{j\in H} W_{ij}s_j - b_i s_i} \\
&= \prod_i \text{Bernoulli}(\sigma(\sum_{j\in H} W_{ij}s_j - b_i))
\end{aligned}
$$

similarly

$$
P(\mathbf{s}^H|\mathbf{s}^V) = \prod_j \text{Bernoulli}(\sigma(\sum_{i\in V} W_{ij}s_i - b_j))
$$

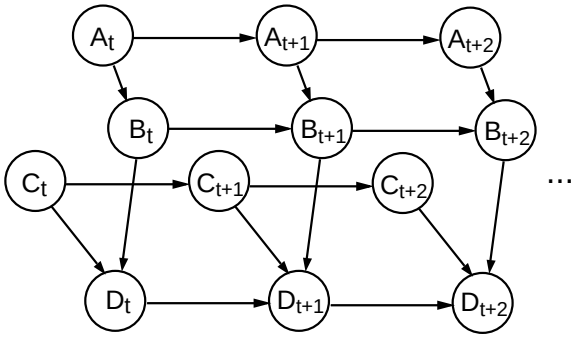
- So inference is tractable ...
- ... but learning still intractable because of normaliser.
- Unclamped samples can be generated efficiently by block Gibbs sampling.
- Often combined with a futher approximation called contrastive divergence learning.

Factorial Hidden Markov Models



- Hidden Markov models with many state variables (i.e. distributed state representation).
- Each state variable evolves independently.
- The state can capture many bits of information about the sequence (linear in the number of state variables).
- E step is typically intractable (due to explaining away in latent states).
- Example case for variational approximation

Dynamic Bayesian Networks



- Distributed HMM with structured dependencies amongst latent states.

Topic Modelling

Topic modelling: given a corpus of documents, find the "topics" they discuss.

Example: consider abstracts of papers PNAS.

Global climate change and mammalian species diversity in U.S. national parks

National parks and bioreserves are key conservation tools used to protect species and their habitats within the confines of fixed political boundaries. This inflexibility may be their "Achilles' heel" as conservation tools in the face of emerging global-scale environmental problems such as climate change. Global climate change, brought about by rising levels of greenhouse gases, threatens to alter the geographic distribution of many habitats and their component species....

The influence of large-scale wind power on global climate

Large-scale use of wind power can alter local and global climate by extracting kinetic energy and altering turbulent transport in the atmospheric boundary layer. We report climate-model simulations that address the possible climatic impacts of wind power at regional to global scales by using two general circulation models and several parameterizations of the interaction of wind turbines with the boundary layer....

Twentieth century climate change: Evidence from small glaciers

The relation between changes in modern glaciers, not including the ice sheets of Greenland and Antarctica, and their climatic environment is investigated to shed light on paleoglacier evidence of past climate change and for projecting the effects of future climate warming on cold regions of the world. Loss of glacier volume has been more or less continuous since the 19th century, but it is not a simple adjustment to the end of an "anomalous" Little Ice Age....

Topic Modelling

Example topics discovered from PNAS abstracts (each topic represented in terms of the top 5 most common words in that topic).

217	274	126	63	200	209
INSECT	SPECIES	GENE	STRUCTURE	FOLDING	NUCLEAR
MYB	PHYLOGENETIC	VECTOR	ANGSTROM	NATIVE	NUCLEUS
PHEROMONE	EVOLUTION	VECTORS	CRYSTAL	PROTEIN	LOCALIZATION
LENS	EVOLUTIONARY	EXPRESSION	RESIDUES	STATE	CYTOPLASM
LARVAE	SEQUENCES	TRANSFER	STRUCTURES	ENERGY	EXPORT
42	**2**	**280**	**15**	**64**	**102**
NEURAL	SPECIES	SPECIES	CHROMOSOME	CELLS	TUMOR
DEVELOPMENT	GLOBAL	SELECTION	REGION	CELL	CANCER
DORSAL	CLIMATE	EVOLUTION	CHROMOSOMES	ANTIGEN	TUMORS
EMBRYOS	CO2	GENETIC	KB	LYMPHOCYTES	HUMAN
VENTRAL	WATER	POPULATIONS	MAP	CD4	CELLS
112	**210**	**201**	**165**	**142**	**222**
HOST	SYNAPTIC	RESISTANCE	CHANNEL	PLANTS	CORTEX
BACTERIAL	NEURONS	RESISTANT	CHANNELS	PLANT	BRAIN
BACTERIA	POSTSYNAPTIC	DRUG	VOLTAGE	ARABIDOPSIS	SUBJECTS
STRAINS	HIPPOCAMPAL	DRUGS	CURRENT	TOBACCO	TASK
SALMONELLA	SYNAPSES	SENSITIVE	CURRENTS	LEAVES	AREAS
39	**105**	**221**	**270**	**55**	**114**
THEORY	HAIR	LARGE	TIME	FORCE	POPULATION
TIME	MECHANICAL	SCALE	SPECTROSCOPY	SURFACE	POPULATIONS
SPACE	MB	DENSITY	NMR	MOLECULES	GENETIC
GIVEN	SENSORY	OBSERVED	SPECTRA	SOLUTION	DIVERSITY
PROBLEM	EAR	OBSERVATIONS	TRANSFER	SURFACES	ISOLATES
		109	**120**		
		RESEARCH	AGE		
		NEW	OLD		
		INFORMATION	AGING		
		UNDERSTANDING	LIFE		
		PAPER	YOUNG		

Recap: Beta Distributions

Recall the Bayesian coin toss example.

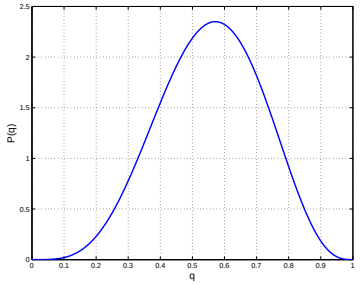
$$P(H|q) = q \qquad P(T|q) = 1 - q$$

The probability of a sequence of coin tosses is:

$$P(HHTT\cdots HT|q) = q^{\#\text{heads}}(1-q)^{\#\text{tails}}$$

A conjugate prior for q is the Beta distribution:

$$P(q) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} q^{a-1}(1-q)^{b-1} \qquad a, b \geq 0$$



Dirichlet Distributions

Imagine a Bayesian dice throwing example.

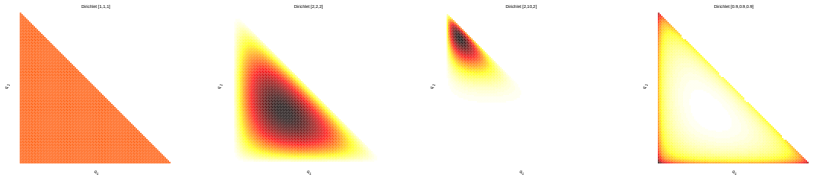
$$P(1|\mathbf{q}) = q_1 \quad P(2|\mathbf{q}) = q_2 \quad P(3|\mathbf{q}) = q_3 \quad P(4|\mathbf{q}) = q_4 \quad P(5|\mathbf{q}) = q_5 \quad P(6|\mathbf{q}) = q_6$$

with $q_i \geq 0, \sum_i q_i = 1$. The probability of a sequence of dice throws is:

$$P(34156\cdots 12|\boldsymbol{q}) = \prod_{i=1}^{6} q_i^{\#\text{ face } i}$$

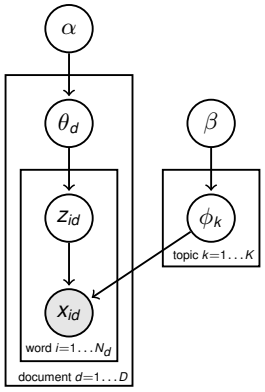
A conjugate prior for $\boldsymbol{q}$ is the Dirichlet distribution:

$$P(\boldsymbol{q}) = \frac{\Gamma(\sum_i a_i)}{\prod_i \Gamma(a_i)} \prod_i q_i^{a_i - 1} \qquad q_i \geq 0, \sum_i q_i = 1 \qquad a_i \geq 0$$



Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.



- Draw topic distributions from a prior

  $$\boldsymbol{\phi}_k \sim \text{Dir}(\beta, \ldots, \beta)$$

- For each document:
  - draw a distribution over topics

    $$\boldsymbol{\theta}_d \sim \text{Dir}(\alpha, \ldots, \alpha)$$

  - generate words iid:
    - draw topic from a document-specific dist:

      $$z_{id} \sim \text{Discrete}(\boldsymbol{\theta}_d)$$

    - draw word from a topic-specific dist:

      $$x_{id} \sim \text{Discrete}(\boldsymbol{\phi}_{z_{id}})$$

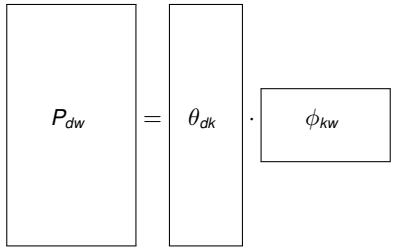
Multiple mixtures of discrete distributions, sharing the same set of components (topics).

Latent Dirichlet Allocation as Matrix Decomposition

Let N_{dw} be the number of times word w appears in document d, and P_{dw} is the probability of word w appearing in document d.

$$p(N|P) = \prod_{dw} P_{dw}^{N_{dw}} \quad \text{likelihood term}$$

$$P_{dw} = \sum_{k} p(\text{pick topic } k)p(\text{pick word } w|k) = \sum_{k=1}^{K} \theta_{dk}\phi_{kw}$$



This decomposition is similar to PCA and factor analysis, but not Gaussian. Related to non-negative matrix factorisation (NMF).

Latent Dirichlet Allocation

- Exact inference in latent Dirichlet allocation is intractable, and typically either variational or Markov chain Monte Carlo approximations are deployed.
- Latent Dirichlet allocation is an example of a mixed membership model from statistics.
- Latent Dirichlet allocation has also been applied to computer vision, social network modelling, natural language processing...
- Generalizations:
  - Relax the bag-of-words assumption (e.g. a Markov model).
  - Model changes in topics through time.
  - Model correlations among occurrences of topics.
  - Model authors, recipients, multiple corpora.
  - Cross modal interactions (images and tags).
  - Nonparametric generalisations.

Nonlinear Dimensionality Reduction

We can see matrix factorisation methods as performing linear dimensionality reduction.

There are many ways to generalise PCA and FA to deal with data which lie on a nonlinear manifold:

- Nonlinear autoencoders
- Generative topographic mappings (GTM) and Kohonen self-organising maps (SOM)
- Multi-dimensional scaling (MDS)
- Kernel PCA (based on MDS representation)
- Isomap
- Locally linear embedding (LLE)
- Stochastic Neighbour Embedding
- Gaussian Process Latent Variable Models (GPLVM)

Another view of PCA: matching inner products

We have viewed PCA as providing a decomposition of the covariance or scatter matrix S. We obtain similar results if we approximate the Gram matrix:

$$\text{minimise} \quad \mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{z}_i \cdot \mathbf{z}_j)^2$$

for $\mathbf{z} \in \mathbb{R}^k$.

That is, look for a k-dimensional embedding in which dot products (which depend on lengths, and angles) are preserved as well as possible.

We will see that this is also equivalent to preserving distances between points.

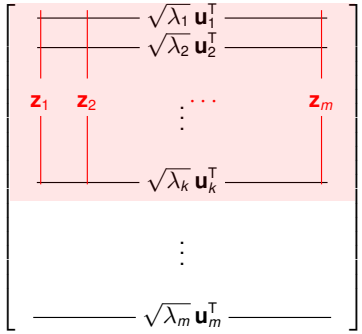
Another view of PCA: matching inner products

Consider the eigendecomposition of G:

$$G = U\Lambda U^\mathsf{T} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-k approximation $G \approx Z^\mathsf{T} Z$ is given by:

$$Z^\mathsf{T} = [U]_{1:m,1:k}[\Lambda^{1/2}]_{1:k,1:k};$$
$$= [U\Lambda^{1/2}]_{1:m,1:k}$$
$$Z = [\Lambda^{1/2}U^\mathsf{T}]_{1:k,1:m}$$



The same operations can be performed on the kernel Gram matrix $\Rightarrow$ Kernel PCA.

Multidimensional Scaling

Suppose all we were given were distances or symmetric "dissimilarities" Δ_{ij}.

$$\Delta = \begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{13} & \Delta_{23} & 0 & \Delta_{34} \\ \Delta_{14} & \Delta_{24} & \Delta_{34} & 0 \end{bmatrix}$$

Goal: Find vectors $\mathbf{z}_i$ such that $\|\mathbf{z}_i - \mathbf{z}_j\| \approx \Delta_{ij}$.

This is called **Multidimensional Scaling (MDS)**.

Metric MDS

Assume the dissimilarities represent Euclidean distances between points in some high-D space.

$$\Delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \text{ with } \sum_i \mathbf{x}_i = \mathbf{0}.$$

We have:

$$\Delta_{ij}^2 = \|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j$$
$$\sum_k \Delta_{ik}^2 = m\|\mathbf{x}_i\|^2 + \sum_k \|\mathbf{x}_k\|^2 - \mathbf{0}$$
$$\sum_k \Delta_{kj}^2 = \sum_k \|\mathbf{x}_k\|^2 + m\|\mathbf{x}_j\|^2 - \mathbf{0}$$
$$\sum_{kl} \Delta_{kl}^2 = 2m \sum_k \|\mathbf{x}_k\|^2$$

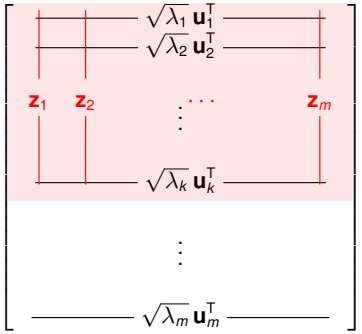
$$\Rightarrow G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j = \frac{1}{2}\left(\frac{1}{m}\sum_k (\Delta_{ik}^2 + \Delta_{kj}^2) - \frac{1}{m^2}\sum_{kl}\Delta_{kl}^2 - \Delta_{ij}^2\right)$$

Metric MDS and eigenvalues

We will actually minimize the error in the dot products:

$$\mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{z}_i \cdot \mathbf{z}_j)^2$$

As in PCA, this is given by the top slice of the eigenvector matrix.



Interpreting MDS

$$G = \frac{1}{2}\left(\frac{1}{m}(\Delta^2 1 + 1\Delta^2) - \Delta^2 - \frac{1}{m^2} 1^\top \Delta^2 1\right)$$

$$G = U\Lambda U^\top; \qquad Y = [\Lambda^{1/2} U^\top]_{1:k,1:m}$$

(1 is a matrix of ones.)

- **Eigenvectors.** Ordered, scaled and truncated to yield low-dimensional embedded points $\mathbf{z}_i$.
- **Eigenvalues.** Measure how much each dimension contributes to dot products.
- **Estimated dimensionality.** Number of significant (nonnegative – negative possible if Δ_{ij} are not metric) eigenvalues.

MDS and PCA

Dual matrices:

$S = \frac{1}{m} XX^\top$	scatter matrix	$(n \times n)$
$G = X^\top X$	Gram matrix	$(m \times m)$

- **Same eigenvalues** up to a constant factor.
- **Equivalent on metric data**, but MDS can run on non-metric dissimilarities.
- **Computational cost** is different.
  - PCA: $O((m+k)n^2)$
  - MDS: $O((n+k)m^2)$

Non-metric MDS

MDS can be generalised to permit a monotonic mapping:

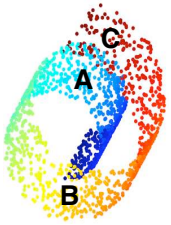
$$\Delta_{ij} \to g(\Delta_{ij}),$$

even if this violates metric rules (like the triangle inequality).

This can introduce a non-linear warping of the manifold.

But

Rank ordering of Euclidean distances is NOT preserved in "manifold learning".

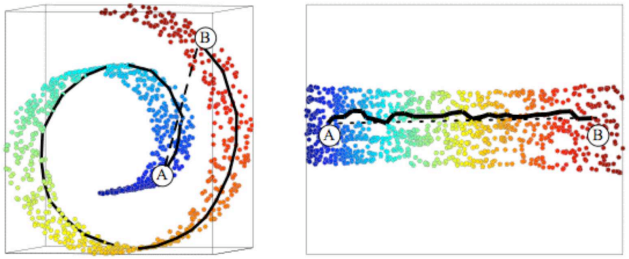


$d(A,C) < d(A,B)$

$d(A,C) > d(A,B)$

Isomap

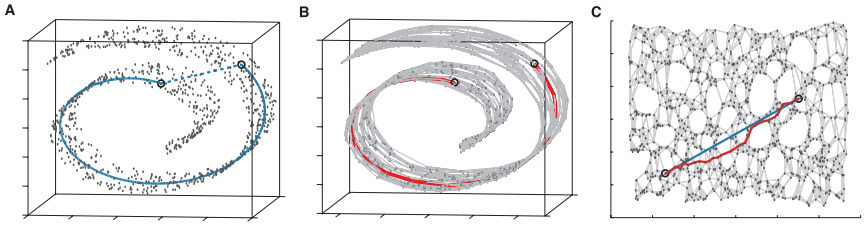
Idea: try to trace distance along the manifold. Use geodesic instead of (transformed) Euclidean distances in MDS.



- preserves local structure
- estimates “global” structure
- preserves information (MDS)

Stages of Isomap

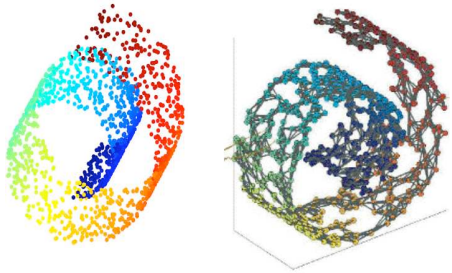
1. Identify neighbourhoods around each point (local points, assumed to be local on the manifold). Euclidean distances are preserved within a neighbourhood.
2. For points outside the neighbourhood, estimate distances by hopping between points within neighbourhoods.
3. Embed using MDS.



Step 1: Adjacency graph

First we construct a graph linking each point to its neighbours.

- vertices represent input points
- undirected edges connect neighbours (weight = Euclidean distance)



Forms a discretised approximation to the submanifold, assuming:

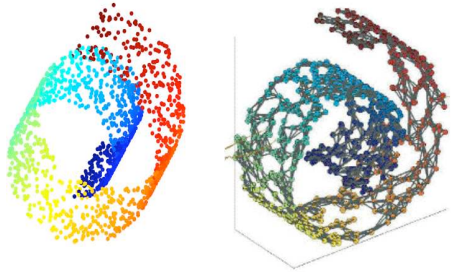
- Graph is singly-connected.
- Graph neighborhoods reflect manifold neighborhoods. No “short cuts”.

Defining the neighbourhood is critical: k-nearest neighbours, inputs within a ball of radius r, prior knowledge.

Step 2: Geodesics

Estimate distances by shortest path in graph.

$$\Delta_{ij} = \min_{\text{path}(\mathbf{x}_i,\mathbf{x}_j)} \left\{ \sum_{e_i \in \text{path}(\mathbf{x}_i,\mathbf{x}_j)} \delta_i \right\}$$

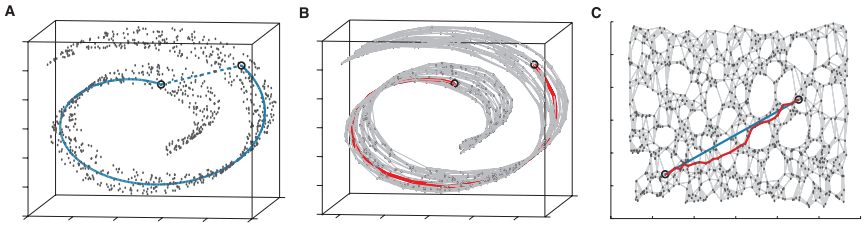


- Standard graph problem. Solved by Dijkstra’s algorithm (and others).
- Better estimates for denser sampling.
- Short cuts very dangerous (“average” path distance?) .

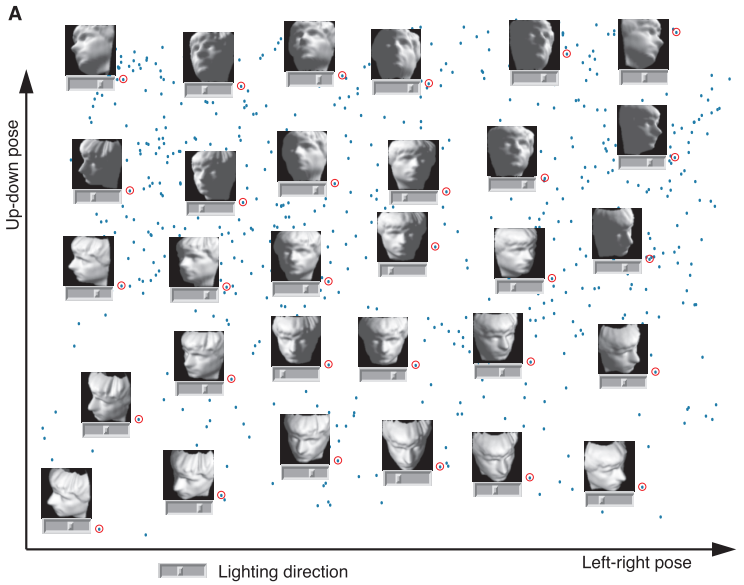
Step 3: Embed

Embed using metric MDS (path distances obey the triangle inequality)

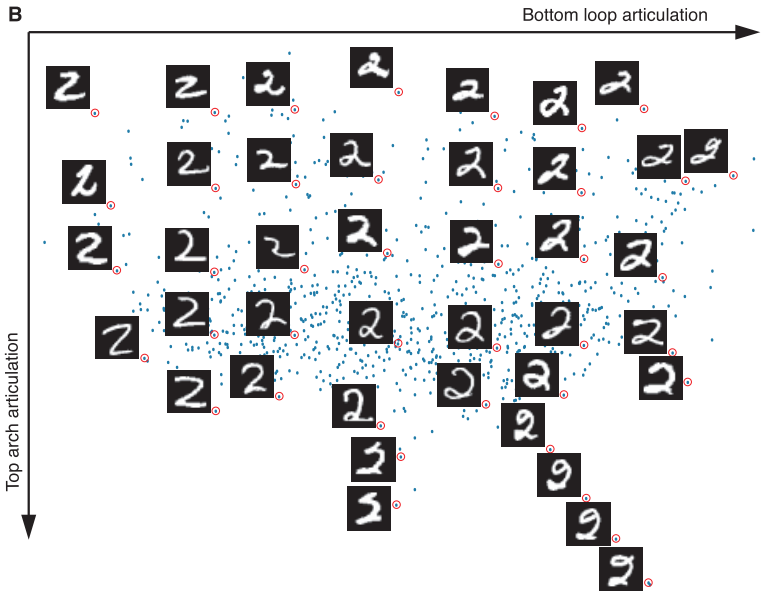
- Eigenvectors of Gram matrix yield low-dimensional embedding.
- Number of significant eigenvalues estimates dimensionality.



Isomap example 1

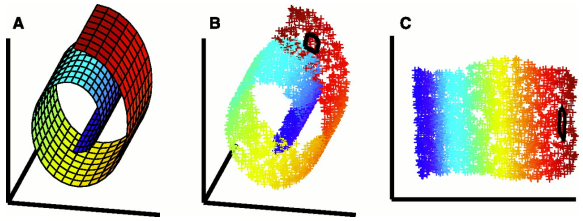


Isomap example 2



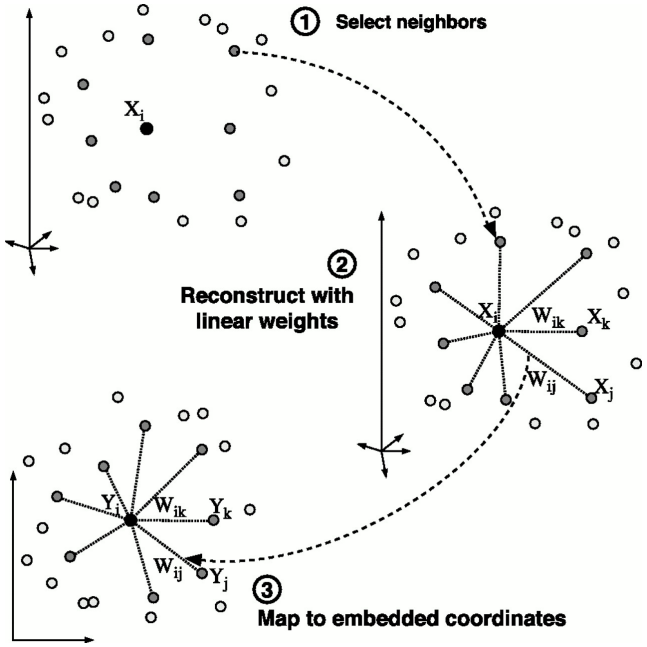
Locally Linear Embedding (LLE)

MDS and isomap preserve local and global (estimated, for isomap) distances. PCA preserves local and global structure.
Idea: estimate local (linear) structure of manifold. Preserve this as well as possible.



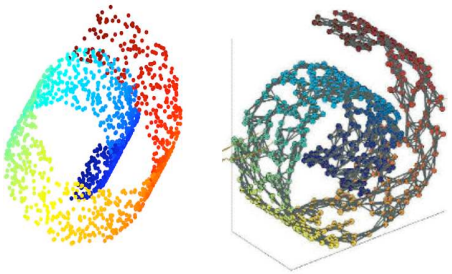
- preserves local structure (not just distance)
- not explicitly global
- preserves only local information

Stages of LLE



Step 1: Neighbourhoods

Just as in isomap, we first define neighbouring points for each input. Equivalent to the isomap graph, but we won't need the graph structure.



Forms a discretised approximation to the submanifold, assuming:

- Graph is singly-connected — although will "work" if not.
- Neighborhoods reflect manifold neighborhoods. No "short cuts".

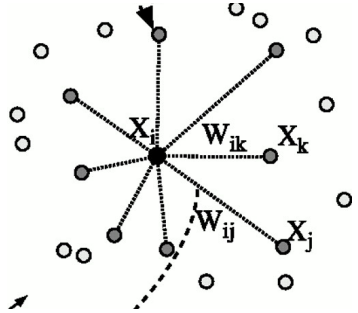
Defining the neighbourhood is critical: k-nearest neighbours, inputs within a ball of radius r, prior knowledge.

Step 2: Local weights

Estimate local weights to minimize error

$$\Phi(W) = \sum_i \left\| \mathbf{x}_i - \sum_{j \in \mathrm{Ne}(i)} W_{ij} \mathbf{x}_j \right\|^2$$

$$\sum_{j \in \mathrm{Ne}(i)} W_{ij} = 1$$



- Linear regression – under- or over-constrained depending on $|\mathrm{Ne}(i)|$.
- Local structure – optimal weights are invariant to rotation, translation and scaling.
- Short cuts less dangerous (one in many).

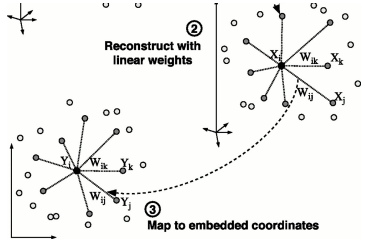
Step 3: Embed

Minimise reconstruction errors in $\mathbf{z}$-space under the same weights:

$$\psi(Z) = \sum_i \left\| \mathbf{z}_i - \sum_{j \in \mathrm{Ne}(i)} W_{ij} \mathbf{z}_j \right\|^2$$

subject to:

$$\sum_i \mathbf{z}_i = \mathbf{0}; \qquad \sum_i \mathbf{z}_i \mathbf{z}_i^\mathsf{T} = mI$$



We can re-write the cost function in quadratic form:

$$\psi(Z) = \sum_{ij} \Psi_{ij} [Z^\mathsf{T} Z]_{ij} \text{ with } \Psi = (I - W)^\mathsf{T} (I - W)$$

Minimise by setting Z to equal the bottom $2 \ldots k+1$ eigenvectors of Ψ. (Bottom eigenvector always $\mathbf{1}$ – discard due to centering constraint)

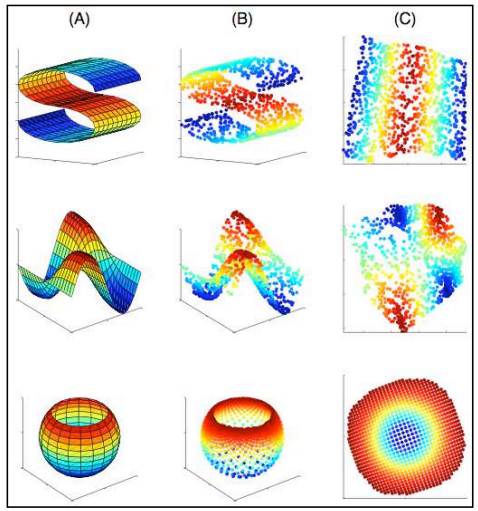
LLE example 1

Surfaces

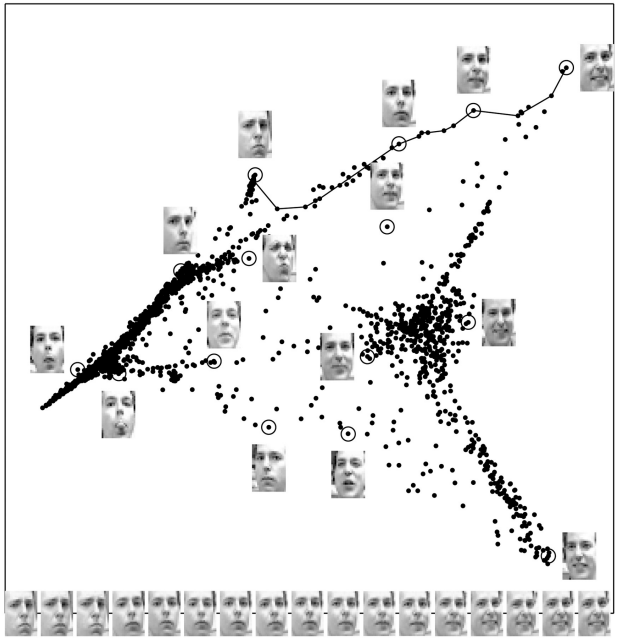
N=1000
inputs

k=8
nearest
neighbors

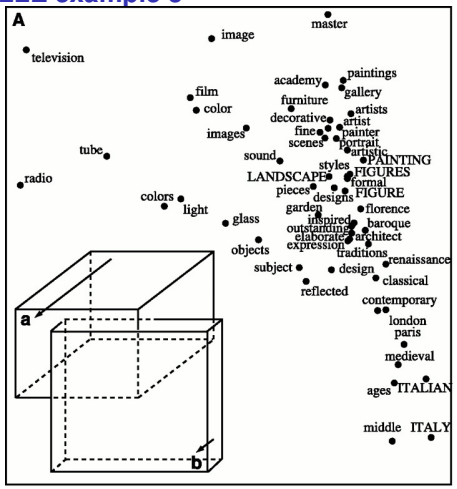
D=3
d=2
dimensions



LLE example 2



LLE example 3



LLE and Isomap

Many similarities

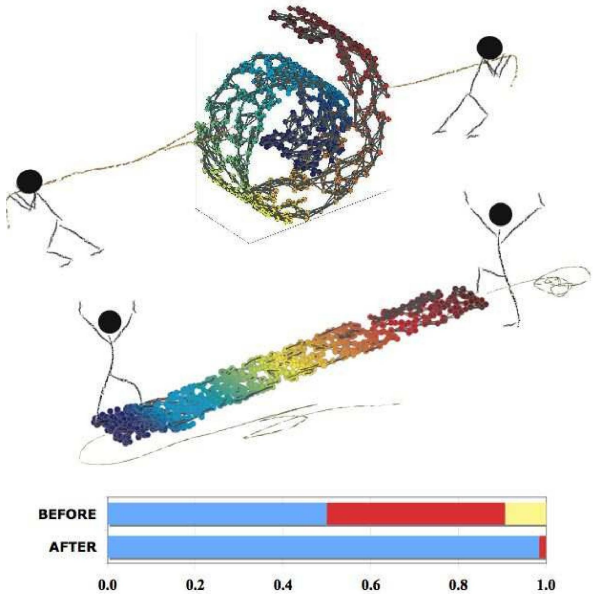
- Graph-based, spectral methods
- No local optima

Essential differences

- LLE does not estimate dimensionality
- Isomap can be shown to be consistent; no theoretical guarantees for LLE.
- LLE diagonalises a sparse matrix – more efficient than isomap.
- Local weights vs. local & global distances.

Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.



Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.

1. Build the neighbourhood graph.
2. Find $\{\mathbf{z}_i\} \subset \mathbb{R}^n$ (points in **high-D** space) with maximum variance, preserving local distances. Let $K_{ij} = \mathbf{z}_i^\top \mathbf{z}_j$. Then:

   Maximise $\mathrm{Tr}\,[K]$ subject to:

   $$\sum_{ij} K_{ij} = 0 \qquad \text{(centered)}$$
   $$K \succeq 0 \qquad \text{(positive definite)}$$
   $$\underbrace{K_{ii} - 2K_{ij} + K_{jj}}_{\|\mathbf{z}_i - \mathbf{z}_j\|^2} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \text{ for } j \in \mathrm{Ne}(i) \qquad \text{(locally metric)}$$

   This is a **semi-definite program**: convex optimisation with unique solution.
3. Embed $\mathbf{z}_i$ in $\mathbb{R}^k$ using linear methods (PCA/MDS).

Stochastic Neighbour Embedding

Softer "probabilistic" notions of neighbourhood and consistency.

High-D "transition" probabilities:

$$p_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k \neq i} e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i, \qquad p_{i|i} = 0$$

Find $\{\mathbf{z}_i\} \subset \mathbb{R}^k$ to:

$$\text{minimise} \sum_{ij} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \qquad \text{with } q_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{z}_i - \mathbf{z}_j\|^2}}{\sum_{k \neq i} e^{-\frac{1}{2}\|\mathbf{z}_i - \mathbf{z}_k\|^2}}.$$

Nonconvex optimisation is initialisation dependent.

Scale σ plays a similar role to neighbourhood definition:

- Fixed σ: resembles a fixed-radius ball.
- Choose σ_i to maintain consistent entropy in $p_{j|i}$ of $\log_2 k$: similar to k-nearest neighbours.

SNE variants

- Symmetrise probabilities ($p_{ij} = p_{ji}$)

  $$p_{ij} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k \neq l} e^{-\frac{1}{2}\|\mathbf{x}_l - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i$$

- Gaussian Process Latent Variable Models. Lawrence. Advances in Neural Information Processing Systems, 2004.
  Define q_{ij} analagously, optimise joint KL.

- Heavy-tailed embedding distributions allow embedding to lower dimensions than true manifold:

  $$q_{ij} = \frac{(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{z}_k - \mathbf{z}_l\|^2)^{-1}}$$

  Student-t distribution defines "t-SNE".

Focus is on visualisation, rather than manifold discovery.

Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{x}_i|\mathbf{z}_i, \Lambda \sim \mathcal{N}(\Lambda\mathbf{z}_i, \beta^{-1}I)$$
$$\mathbf{z}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$, maximizing likelihood over Λ.

Suppose we know the values of the latent Z, then we can integrate out Λ (c.f. linear regression), giving a conditional probability of $X = [\mathbf{x}_1 \ldots \mathbf{x}_N]^\top$:

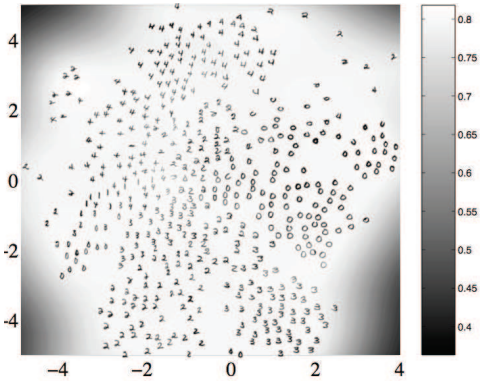
$$\Lambda \sim \mathcal{N}(0, \alpha^{-1}I)$$
$$p(X|Z) \sim |2\pi K|^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}[K^{-1}XX^\top]\right) \qquad K = \alpha ZZ^\top + \beta I$$

This is just D independent Gaussian processes, one for each dimension of X! Each Gaussian process describes a mapping from latent space $\mathbf{z}$ to one dimension of $\mathbf{x}$.

Replacing the linear kernel with nonlinear kernels gives nonlinear mappings—nonlinear dimensionality reduction.

But now dependence on Z is complicated—instead of computing a posterior over Z we must find point values that maximise the likelihood (jointly with the hyperparameters), or use a variational approximation (cf also the Locally-Linear Latent Variable Model).

Gaussian Process Latent Variable Models

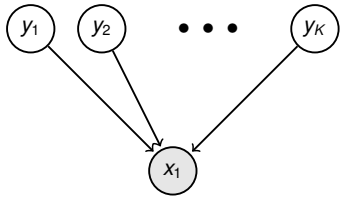


Intractability

For many probabilistic models of interest, exact inference is not computationally feasible.

There are three (main) reasons:

- Distributions may have complicated forms (e.g. non-linearities in generative model).
- "Explaining away": observing the value of a child induces dependencies amongst its parents.



- Even with simple models, Bayesian computation of the full posterior over both latent variables and parameters is made complicated by the strong coupling between latent variables and parameters.

We can still work with such models by using *approximate inference* techniques to estimate the latent variables.

Approximate Inference

- Linearisation: Approximate nonlinearities by Taylor series expansion about a point (e.g. the approximate mean or mode of the hidden variable distribution). Linear approximations are particularly useful since Gaussian distributions are closed under linear transformations (e.g., EKF). Also Laplace's approximation.
- Monte Carlo Sampling: Approximate posterior distribution over unobserved variables by a set of random samples. We often need Markov chain Monte carlo or sequential Monte Carlo methods to sample from difficult distributions.
- Variational Methods: Approximate the hidden variable posterior $p(H)$ with a tractable form $q(H)$, such that $\mathbf{KL}[q||p]$ is minimised. This gives a lower bound on the likelihood that can be maximised with respect to the parameters of $q(H)$.
- Local Message Passing Methods: Approximate the hidden variable posterior $p(H)$ with a tractable form $q(H)$ or with a set of locally consistent tractable forms by other means (loopy belief propagation, expectation propagation).
- Recognition Models and Autoencoders: Approximate the hidden variable posterior distribution using an explicit *bottom-up* recognition model/network.

References

- Pattern Classification. Duda, Hart and Stork. Wiley, 2000.
- A Unifying Review of Linear Gaussian Models. Roweis and Ghaharamani. Neural Computation, 1999.
- Independent Component Analysis. Hyvarinen, Karhunan and Oja. John Wiley and Sons, 2001.
- Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. Olshausen & Field Nature, 1996.
- A Learning Algorithm for Boltzmann Machines. Ackley, Hinton and Sejnowski. Cognitive Science, 1985.
- Connectionist Learning of Belief Networks. Neal. Artificial Intelligence, 1992.
- Latent Dirichlet Allocation. Blei, Ng and Jordan. Journal of Machine Learning Research, 2003.
- Factorial Hidden Markov Models. Ghahramani and Jordan. Machine Learning, 1997.
- Dynamic Bayesian Networks: Representation, Inference and Learning. Kevin Murphy. PhD Thesis, 2002.

References

- Isomap. Tenenbaum, de Silva & Langford, Science, **290**(5500):2319–23 (2000).
- LLE. Roweis & Saul, Science, **290**(5500):2323–6 (2000).
- Laplacian Eigenmaps. Belkin & Niyogi, Neural Comput **23**(6):1373–96 (2003).
- Hessian LLE. Donoho & Grimes, PNAS **100**(10): 5591–6 (2003).
- Maximum variance unfolding. Weinberger & Saul, Int J Comput Vis **70**(1):77–90 (2006).
- Conformal eigenmaps. Sha & Saul ICML **22**:785–92 (2005).
- SNE Hinton & Roweis, NIPS, 2002; t-SNE van der Maaten & Hinton, JMLR, 9:2579–2605, 2008.
- Gaussian Process Latent Variable Models Lawrence. Advances in Neural Information Processing Systems, 2004.
- Locally-Linear Latent Variable Models Park et al. Advances in Neural Information Processing Systems, 2015.

More at: `http://www.gatsby.ucl.ac.uk/~maneesh/dimred/`

Probabilistic & Unsupervised Learning
Approximate Inference

Factored Variational Approximations and Variational Bayes

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2023

Intractabilities and approximations

- Inference – computational intractability
  - Factored variational approx
  - Loopy BP/EP/Power EP
  - LP relaxations/ convexified BP
  - Gibbs sampling, other MCMC
- Inference – analytic intractability
  - Laplace approximation (global)
  - Parametric variational approx
  - Message approximations (linearised, sigma-point, Laplace)
  - Assumed-density methods and Expectation-Propagation
  - (Sequential) Monte-Carlo methods
- Learning – intractable partition function
  - Sampling parameters
  - Constrastive divergence
  - Score-matching
- Model selection
  - Laplace approximation / BIC
  - Variational Bayes
  - (Annealed) importance sampling
  - Reversible jump MCMC

Not a complete list!

Intractabilities

- **Analytic intractability**: non-conjugacy prevents closed-form evaluation of likelihood or free energy

$$\ell(\theta) = \int d\mathcal{Z}\; P(\mathcal{Z}|\theta)P(\mathcal{X}|\mathcal{Z},\theta) \quad \mathcal{F}(\theta, Q) = \int d\mathcal{Z}\; Q(\mathcal{Z}) \log P(\mathcal{X}, \mathcal{Z}|\theta) + \mathbf{H}[Q]$$

- **Computational intractability**: graphical structure prevents simplification of high-dimensional sums or integrals

$$Q(\mathbf{z}_i) \propto \int d\mathbf{z}_1 \ldots d\mathbf{z}_{i-1}\, d\mathbf{z}_{i+1} \ldots d\mathbf{z}_k\; \log P(\mathcal{X}, \mathcal{Z}|\theta)$$

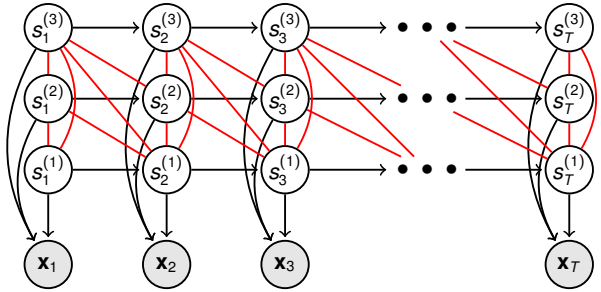
- **Intractable normalisers**: learning requires normaliser gradient

$$\nabla_\theta \mathcal{F}(\theta, Q) = \nabla_\theta \langle E(\mathcal{X}, \mathcal{Z}|\theta)\rangle_Q - \nabla_\theta \log Z(\theta)$$

- **Model selection or weighting** (by marginal likelihood)

$$p(\mathcal{D}|m) = \int d\theta\; p(\theta|m)p(\mathcal{D}|\theta, m)$$

Computational intractability – distributed models



Consider an FHMM with M state variables taking on K values each.

- Moralisation puts simultaneous states $(s_t^{(1)}, s_t^{(2)}, \ldots, s_t^{(M)})$ into a single clique
- Triangulation extends cliques to size $M + 1$
- Each state takes K values $\Rightarrow$ sums over K^{M+1} terms.
- Factorial *prior* $\not\Rightarrow$ Factorial *posterior* (explaining away).

Variational methods approximate the posterior, often in a factored form.
To see how they work, we need to review the free-energy interpretation of EM.

The Free Energy for a Latent Variable Model

Observed data $\mathcal{X} = \{\mathbf{x}_i\}$; Latent variables $\mathcal{Z} = \{\mathbf{z}_i\}$; Parameters θ.

Goal: Maximize the log likelihood wrt θ (i.e. ML learning):

$$\ell(\theta) = \log P(\mathcal{X}|\theta) = \log \int P(\mathcal{Z}, \mathcal{X}|\theta) d\mathcal{Z}$$

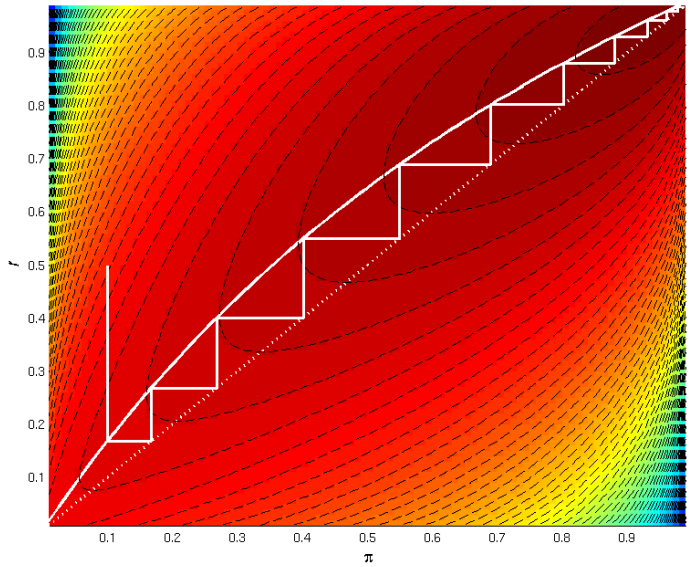
Any distribution, $q(\mathcal{Z})$, over the hidden variables can be used to obtain a lower bound on the log likelihood using Jensen's inequality:

$$\ell(\theta) = \log \int q(\mathcal{Z}) \frac{P(\mathcal{Z}, \mathcal{X}|\theta)}{q(\mathcal{Z})} \, d\mathcal{Z} \geq \int q(\mathcal{Z}) \log \frac{P(\mathcal{Z}, \mathcal{X}|\theta)}{q(\mathcal{Z})} \, d\mathcal{Z} \stackrel{\text{def}}{=} \mathcal{F}(q, \theta)$$

$$\begin{aligned} \int q(\mathcal{Z}) \log \frac{P(\mathcal{Z}, \mathcal{X}|\theta)}{q(\mathcal{Z})} \, d\mathcal{Z} &= \int q(\mathcal{Z}) \log P(\mathcal{Z}, \mathcal{X}|\theta) \, d\mathcal{Z} - \int q(\mathcal{Z}) \log q(\mathcal{Z}) \, d\mathcal{Z} \\ &= \int q(\mathcal{Z}) \log P(\mathcal{Z}, \mathcal{X}|\theta) \, d\mathcal{Z} + \mathbf{H}[q], \end{aligned}$$

where $\mathbf{H}[q]$ is the entropy of $q(\mathcal{Z})$.
So: $\mathcal{F}(q, \theta) = \langle \log P(\mathcal{Z}, \mathcal{X}|\theta) \rangle_{q(\mathcal{Z})} + \mathbf{H}[q]$

EM as Coordinate Ascent in $\mathcal{F}$



The E and M steps of EM

The log likelihood is bounded below by:

$$\mathcal{F}(q, \theta) = \langle \log P(\mathcal{Z}, \mathcal{X}|\theta) \rangle_{q(\mathcal{Z})} + \mathbf{H}[q] = \ell(\theta) - \mathbf{KL}[q(\mathcal{Z}) \| P(\mathcal{Z}|\mathcal{X}, \theta)]$$

EM alternates between:

E step: optimise $\mathcal{F}(q, \theta)$ wrt distribution over hidden variables holding parameters fixed:

$$q^{(k)}(\mathcal{Z}) := \underset{q(\mathcal{Z})}{\operatorname{argmax}} \; \mathcal{F}(q(\mathcal{Z}), \theta^{(k-1)}) = P(\mathcal{Z}|\mathcal{X}, \theta^{(k-1)})$$

M step: maximise $\mathcal{F}(q, \theta)$ wrt parameters holding hidden distribution fixed:

$$\theta^{(k)} := \underset{\theta}{\operatorname{argmax}} \; \mathcal{F}(q^{(k)}(\mathcal{Z}), \theta) = \underset{\theta}{\operatorname{argmax}} \; \langle \log P(\mathcal{Z}, \mathcal{X}|\theta) \rangle_{q^{(k)}(\mathcal{Z})}$$

EM Never Decreases the Likelihood

The E and M steps together never decrease the log likelihood:

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

- The E step brings the free energy to the likelihood.
- The M-step maximises the free energy wrt θ.
- $\mathcal{F} \leq \ell$ by Jensen – or, equivalently, from the non-negativity of KL

If the M-step is executed so that $\theta^{(k)} \neq \theta^{(k-1)}$ iff $\mathcal{F}$ increases, then the overall EM iteration will step to a new value of θ iff the likelihood increases.

Free-energy-based variational approximation

What if finding expected sufficient stats under $P(\mathcal{Z}|\mathcal{X},\theta)$ is computationally intractable?

For the **generalised EM** algorithm, we argued that intractable maximisations could be replaced by gradient M-steps.

- Each step increases the likelihood.
- A fixed point of the gradient M-step must be at a mode of the expected log-joint.

For the E-step we could:

- Parameterise $q = q_\rho(\mathcal{Z})$ and take a gradient step in ρ.
- Assume some simplified form for q, usually factored: $q = \prod_i q_i(\mathcal{Z}_i)$ where $\mathcal{Z}_i$ partition $\mathcal{Z}$, and maximise within this form.

In either case, we choose q from within a limited set $\mathcal{Q}$:

VE step: maximise $\mathcal{F}(q,\theta)$ wrt constrained latent distribution given parameters:

$$q^{(k)}(\mathcal{Z}) := \underset{q(\mathcal{Z})\in\mathcal{Q}\leftarrow\textbf{Constraint}}{\operatorname{argmax}}\ \mathcal{F}(q(\mathcal{Z}),\theta^{(k-1)}).$$

M step: unchanged

$$\theta^{(k)} := \underset{\theta}{\operatorname{argmax}}\ \mathcal{F}(q^{(k)}(\mathcal{Z}),\theta) = \underset{\theta}{\operatorname{argmax}} \int q^{(k)}(\mathcal{Z}) \log p(\mathcal{Z},\mathcal{X}|\theta) d\mathcal{Z},$$

Unlike in GEM, the fixed point may not be at an unconstrained optimum of $\mathcal{F}$.

KL divergence

Recall that

$$\begin{aligned}\mathcal{F}(q,\theta) &= \langle \log P(\mathcal{X},\mathcal{Z}|\theta)\rangle_{q(\mathcal{Z})} + \mathbf{H}[q] \\ &= \langle \log P(\mathcal{X}|\theta) + \log P(\mathcal{Z}|\mathcal{X},\theta)\rangle_{q(\mathcal{Z})} - \langle \log q(\mathcal{Z})\rangle_{q(\mathcal{Z})} \\ &= \langle \log P(\mathcal{X}|\theta)\rangle_{q(\mathcal{Z})} - \mathbf{KL}[q\|P(\mathcal{Z}|\mathcal{X},\theta)].\end{aligned}$$

Thus,

E step maximise $\mathcal{F}(q,\theta)$ wrt the distribution over latents, given parameters:

$$q^{(k)}(\mathcal{Z}) := \underset{q(\mathcal{Z})\in\mathcal{Q}}{\operatorname{argmax}}\ \mathcal{F}(q(\mathcal{Z}),\theta^{(k-1)}).$$

is equivalent to:

E step minimise $\mathbf{KL}[q\|p(\mathcal{Z}|\mathcal{X},\theta)]$ wrt distribution over latents, given parameters:

$$q^{(k)}(\mathcal{Z}) := \underset{q(\mathcal{Z})\in\mathcal{Q}}{\operatorname{argmin}} \int q(\mathcal{Z}) \log \frac{q(\mathcal{Z})}{p(\mathcal{Z}|\mathcal{X},\theta^{(k-1)})} d\mathcal{Z}$$

So, in each E step, the algorithm is trying to find the best approximation to $P(\mathcal{Z}|\mathcal{X})$ in $\mathcal{Q}$ in a KL sense. This is related to ideas in *information geometry*. It also suggests generalisations to other distance measures.

What do we lose?

What does restricting q to $\mathcal{Q}$ cost us?

- Recall that the free-energy is bounded above by Jensen:

  $$\mathcal{F}(q,\theta) \le \ell(\theta^{\mathrm{ML}})$$

  Thus, as long as every step increases $\mathcal{F}$, convergence is still guaranteed.

- But, since $P(\mathcal{Z}|\mathcal{X},\theta^{(k)})$ may not lie in $\mathcal{Q}$, we no longer saturate the bound after the E-step. Thus, the likelihood may not increase on each full EM step.

  $$\ell(\theta^{(k-1)}) \underset{\text{E step}}{\not\le} \mathcal{F}(q^{(k)},\theta^{(k-1)}) \underset{\text{M step}}{\le} \mathcal{F}(q^{(k)},\theta^{(k)}) \underset{\text{Jensen}}{\le} \ell(\theta^{(k)}),$$

- This means we may not (and usually won't) converge to a maximum of ℓ.

The hope is that by *increasing a lower bound* on ℓ we will find a decent solution.
[Note that if $P(\mathcal{Z}|\mathcal{X},\theta^{\mathrm{ML}}) \in \mathcal{Q}$, then θ^{ML} is a fixed point of the variational algorithm.]

Factored Variational E-step

The most common form of variational approximation partitions $\mathcal{Z}$ into disjoint sets $\mathcal{Z}_i$ with

$$\mathcal{Q} = \{q \mid q(\mathcal{Z}) = \prod_i q_i(\mathcal{Z}_i)\}.$$

In this case the E-step is itself iterative:

(Factored VE step)$_i$: maximise $\mathcal{F}(q,\theta)$ wrt $q_i(\mathcal{Z}_i)$ given other q_j and parameters:

$$q_i^{(k)}(\mathcal{Z}_i) := \underset{q_i(\mathcal{Z}_i)}{\operatorname{argmax}}\ \mathcal{F}(q_i(\mathcal{Z}_i)\prod_{j\neq i} q_j(\mathcal{Z}_j),\theta^{(k-1)}).$$

- q_i updates iterated to convergence to "complete" VE-step.
- In fact, every (VE)$_i$-step separately increases $\mathcal{F}$, so any schedule of (VE)$_i$- and M-steps will converge. Choice can be dictated by practical issues (rarely efficient to fully converge E-step before updating parameters).

Factored Variational E-step

The Factored Variational E-step has a general form.

The free energy is:

$$\mathcal{F}\Big(\prod_j q_j(\mathcal{Z}_j), \theta^{(k-1)}\Big) = \Big\langle \log P(\mathcal{X}, \mathcal{Z}|\theta^{(k-1)})\Big\rangle_{\prod_j q_j(\mathcal{Z}_j)} + \mathbf{H}\Big[\prod_j q_j(\mathcal{Z}_j)\Big]$$

$$= \int d\mathcal{Z}_i \; q_i(\mathcal{Z}_i)\Big\langle \log P(\mathcal{X}, \mathcal{Z}|\theta^{(k-1)})\Big\rangle_{\prod_{j\neq i} q_j(\mathcal{Z}_j)} + \mathbf{H}[q_i] + \sum_{j\neq i}\mathbf{H}[q_j]$$

Now, taking the variational derivative of the Lagrangian (enforcing normalisation of q_i):

$$\frac{\delta}{\delta q_i}\left(\mathcal{F} + \lambda\left(\int q_i - 1\right)\right) = \Big\langle \log P(\mathcal{X}, \mathcal{Z}|\theta^{(k-1)})\Big\rangle_{\prod_{j\neq i} q_j(\mathcal{Z}_j)} - \log q_i(\mathcal{Z}_i) - \frac{q_i(\mathcal{Z}_i)}{q_i(\mathcal{Z}_i)} + \lambda$$

$$(=0) \quad \Rightarrow \quad q_i(\mathcal{Z}_i) \propto \exp\Big\langle \log P(\mathcal{X}, \mathcal{Z}|\theta^{(k-1)})\Big\rangle_{\prod_{j\neq i} q_j(\mathcal{Z}_j)}$$

In general, this depends only on the expected sufficient statistics under q_j. Thus, again, we don't actually need the *entire* distributions, just the relevant expectations (now for approximate inference as well as learning).

Mean-field approximations

If $\mathcal{Z}_i = z_i$ (*i.e.*, q is factored over all variables) then the variational technique is often called a "mean field" approximation.

- Suppose $P(\mathcal{X}, \mathcal{Z})$ has sufficient statistics that are separable in the latent variables: *e.g.* the Boltzmann machine

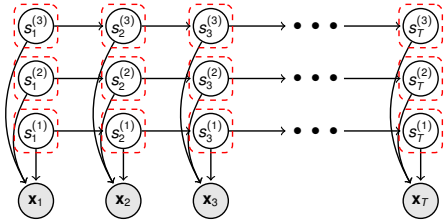
$$P(\mathcal{X}, \mathcal{Z}) = \frac{1}{Z}\exp\Big(\sum_{ij} W_{ij} s_i s_j + \sum_i b_i s_i\Big)$$

  with some $s_i \in \mathcal{Z}$ and others observed.
- Expectations wrt a fully-factored q distribute over all $s_i \in \mathcal{Z}$

$$\langle \log P(\mathcal{X}, \mathcal{Z})\rangle_{\prod q_i} = \sum_{ij} W_{ij}\langle s_i\rangle_{q_i}\langle s_j\rangle_{q_j} + \sum_i b_i \langle s_i\rangle_{q_i}$$

  (where q_i for $s_i \in \mathcal{X}$ is a delta function on the observed value).
- Thus, we can update each q_i in turn given the means (or, in general, mean sufficient statistics) of the others.
- Each variable sees the mean field imposed by its neighbours, and we update these fields until they all agree.

Mean-field FHMM



$$q(s^{1:M}_{1:T}) = \prod_{m,t} q^m_t(s^m_t)$$

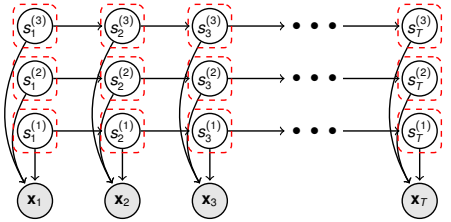
$$q^m_t(s^m_t) \propto \exp\Big\langle \log P(\mathbf{s}^{1:M}_{1:T}, \mathbf{x}_{1:T})\Big\rangle_{\prod_{\neg(m,t)} q^{m'}_{t'}(s^{m'}_{t'})}$$

$$= \exp\left\langle \sum_\mu \sum_\tau \log P(s^\mu_\tau | s^\mu_{\tau-1}) + \sum_\tau \log P(\mathbf{x}_\tau | s^{1:M}_\tau)\right\rangle_{\prod_{\neg(m,t)} q^{m'}_{t'}}$$

$$\propto \exp\Big[\underbrace{\big\langle \log P(s^m_t|s^m_{t-1})\big\rangle_{q^m_{t-1}} + \big\langle \log P(\mathbf{x}_t|s^{1:M}_t)\big\rangle_{\prod_{\neg m} q^{m'}_t}}_{\alpha^m_t(i) \propto e^{\sum_j \log \Phi^m_{ji} q^m_{t-1}(j)} \cdot e^{\langle \log A_i(\mathbf{x}_t)\rangle_{q^{\neg m}_t}}} + \underbrace{\big\langle \log P(s^m_{t+1}|s^m_t)\big\rangle_{q^m_{t+1}}}_{\beta^m_t(i) \propto e^{\sum_j \log \Phi^m_{ij} q^m_{t+1}(j)}}\Big]$$

Cf. forward-backward: $\alpha_t(i) \propto \sum_j \alpha_{t-1}(j)\Phi_{ji}\cdot A_i(\mathbf{x}_t)$ $\quad$ $\beta_t(i) \propto \sum_j \Phi_{ij} A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)$

Mean-field FHMM



$$q(s^{1:M}_{1:T}) = \prod_{m,t} q^m_t(s^m_t)$$

$$q^m_t(s^m_t) \propto \exp\Big[\underbrace{\big\langle \log P(s^m_t|s^m_{t-1})\big\rangle_{q^m_{t-1}} + \big\langle \log P(\mathbf{x}_t|s^{1:M}_t)\big\rangle_{\prod_{\neg m} q^{m'}_t}}_{\alpha^m_t(i) \propto e^{\sum_j \log \Phi^m_{ji} q^m_{t-1}(j)} \cdot e^{\langle \log A_i(\mathbf{x}_t)\rangle_{q^{\neg m}_t}}} + \underbrace{\big\langle \log P(s^m_{t+1}|s^m_t)\big\rangle_{q^m_{t+1}}}_{\beta^m_t(i) \propto e^{\sum_j \log \Phi^m_{ij} q^m_{t+1}(j)}}\Big]$$

Cf. forward-backward: $\alpha_t(i) \propto \sum_j \alpha_{t-1}(j)\Phi_{ji}\cdot A_i(\mathbf{x}_t)$ $\quad$ $\beta_t(i) \propto \sum_j \Phi_{ij} A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)$

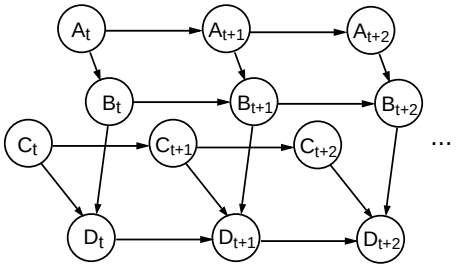
- Yields a message-passing algorithm like forward-backward
- Updates depend only on immediate neighbours in chain
- Chains couple only through joint output
- Multiple passes; messages depend on (approximate) marginals
- Evidence does not appear explicitly in backward message (cf Kalman smoothing)

Structured variational approximation

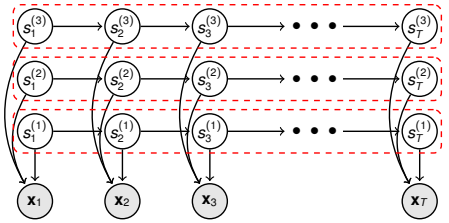
- $q(\mathcal{Z})$ need not be completely factorized.
- For example, suppose $\mathcal{Z}$ can be partitioned into sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ such that computing the expected sufficient statistics under $P(\mathcal{Z}_1|\mathcal{Z}_2,\mathcal{X})$ and $P(\mathcal{Z}_2|\mathcal{Z}_1,\mathcal{X})$ would be tractable.

$\Rightarrow$ Then the factored approximation $q(\mathcal{Z}) = q(\mathcal{Z}_1)q(\mathcal{Z}_2)$ is tractable.

- In particular, any factorisation of $q(\mathcal{Z})$ into a product of distributions on trees, yields a tractable approximation.



Stuctured FHMM



For the FHMM we can factor the chains:

$$q(s_{1:T}^{1:M}) = \prod_m q^m(s_{1:T}^m)$$

$$q^m(s_{1:T}^m) \propto \exp\left\langle \log P(\mathbf{s}_{1:T}^{1:M}, \mathbf{x}_{1:T}) \right\rangle_{\prod_{\neg m} q^{m'}(s_{1:T}^{m'})}$$

$$= \exp\left\langle \sum_\mu \sum_t \log P(s_t^\mu | s_{t-1}^\mu) + \sum_t \log P(\mathbf{x}_t | s_t^{1:M}) \right\rangle_{\prod_{\neg m} q^{m'}}$$

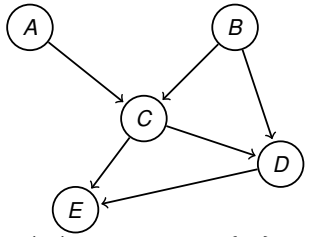
$$\propto \exp\left[\sum_t \log P(s_t^m | s_{t-1}^m) + \sum_t \left\langle \log P(\mathbf{x}_t | s_t^{1:M}) \right\rangle_{\prod_{\neg m} q^{m'}(s_t^{m'})} \right]$$

$$= \prod_t P(s_t^m | s_{t-1}^m) \prod_t e^{\left\langle \log P(\mathbf{x}_t | s_t^{1:M}) \right\rangle_{\prod_{\neg m} q^{m'}(s_t^{m'})}}$$

This looks like a standard HMM joint, with a modified likelihood term $\Rightarrow$ cycle through multiple forward-backward passes, updating likelihood terms each time.

Messages on an arbitrary graph

Consider a DAG:



$$P(\mathcal{X},\mathcal{Z}) = \prod_k P(V_k | \text{pa}(V_k))$$

and let $q(\mathcal{Z}) = \prod_i q_i(\mathcal{Z}_i)$ for disjoint sets $\{\mathcal{Z}_i\}$.

We have that the VE update for q_i is given by $q_i^*(\mathcal{Z}_i) \propto \exp\langle \log p(\mathcal{Z},\mathcal{X})\rangle_{q_{\neg i}(\mathcal{Z})}$ where $\langle\cdot\rangle_{q_{\neg i}(\mathcal{Z})}$ denotes averaging with respect to $q_j(\mathcal{Z}_j)$ for all $j \neq i$

Then:

$$\log q_i^*(\mathcal{Z}_i) = \left\langle \sum_k \log P(V_k | \text{pa}(V_k)) \right\rangle_{q_{\neg i}(\mathcal{Z})} + \text{const}$$

$$= \sum_{j \in \mathcal{Z}_i} \langle \log P(Z_j | \text{pa}(Z_j)) \rangle_{q_{\neg i}(\mathcal{Z})} + \sum_{j \in \text{ch}(\mathcal{Z}_i)} \langle \log P(V_j | \text{pa}(V_j)) \rangle_{q_{\neg i}(\mathcal{Z})} + \text{const}$$

This defines messages that are passed between nodes in the graph. Each node receives messages from its Markov boundary: parents, children and parents of children (all neighbours in the corresponding factor graph).

Non-factored variational methods

The term variational approximation is used whenever a bound on the likelihood (or on another estimation cost function) is optimised, but does not necessarily become tight.

Many further variational approximations have been developed, including:

- parametric forms (e.g. Gaussian) for non-linear models (later lecture)
  - closed form updates in special cases
  - numerical or sampling-based computation of expectations
  - 'recognition networks' or amortisation to estimate variational parameters
- non-free-energy-based bounds (both upper and lower) on the likelihood.

We can also see MAP- or zero-temperature EM and recognition models as parametric forms of variational inference.

Variational Bayes

So far, we have applied Jensen's bound and factorisations to help with integrals over latent variables.

We can do the same for integrals over parameters in order to bound the log marginal likelihood or evidence.

$$\begin{aligned}\log P(\mathcal{X}|\mathcal{M}) &= \log \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, P(\mathcal{X},\mathcal{Z}|\boldsymbol{\theta},\mathcal{M})P(\boldsymbol{\theta}|\mathcal{M}) \\ &= \max_{Q} \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, Q(\mathcal{Z},\boldsymbol{\theta}) \log \frac{P(\mathcal{X},\mathcal{Z},\boldsymbol{\theta}|\mathcal{M})}{Q(\mathcal{Z},\boldsymbol{\theta})} \\ &\geq \max_{Q_{\mathcal{Z}},Q_{\boldsymbol{\theta}}} \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, Q_{\mathcal{Z}}(\mathcal{Z})Q_{\theta}(\boldsymbol{\theta}) \log \frac{P(\mathcal{X},\mathcal{Z},\boldsymbol{\theta}|\mathcal{M})}{Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})}\end{aligned}$$

The constraint that the distribution Q must factor into the product $Q_y(\mathcal{Z})Q_\theta(\boldsymbol{\theta})$ leads to the **variational Bayesian EM algorithm** or just **"Variational Bayes"**.

Some call this the "Evidence Lower Bound" (ELBO). I'm not fond of that term.

Variational Bayesian EM ...

Coordinate maximization of the VB free-energy lower bound

$$\mathcal{F}(Q_{\mathcal{Z}},Q_{\boldsymbol{\theta}}) = \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, Q_{\mathcal{Z}}(\mathcal{Z})Q_{\theta}(\boldsymbol{\theta}) \log \frac{p(\mathcal{X},\mathcal{Z},\boldsymbol{\theta}|\mathcal{M})}{Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})}$$

leads to **EM-like** updates:

$$\begin{aligned} Q^*_{\mathcal{Z}}(\mathcal{Z}) &\propto \exp\langle \log P(\mathcal{Z},\mathcal{X}|\boldsymbol{\theta})\rangle_{Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})} && \textit{E-like step} \\ Q^*_{\boldsymbol{\theta}}(\boldsymbol{\theta}) &\propto P(\boldsymbol{\theta}) \exp\langle \log P(\mathcal{Z},\mathcal{X}|\boldsymbol{\theta})\rangle_{Q_{\mathcal{Z}}(\mathcal{Z})} && \textit{M-like step}\end{aligned}$$

Maximizing $\mathcal{F}$ is equivalent to minimizing KL-divergence between the *approximate posterior*, $Q(\boldsymbol{\theta})Q(\mathcal{Z})$ and the *true posterior*, $P(\boldsymbol{\theta},\mathcal{Z}|\mathcal{X})$.

$$\begin{aligned}\log P(\mathcal{X}) - \mathcal{F}(Q_{\mathcal{Z}},Q_{\boldsymbol{\theta}}) &= \log P(\mathcal{X}) - \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \log \frac{P(\mathcal{X},\mathcal{Z},\boldsymbol{\theta})}{Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})} \\ &= \iint d\mathcal{Z}\, d\boldsymbol{\theta}\, Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \log \frac{Q_{\mathcal{Z}}(\mathcal{Z})Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})}{P(\mathcal{Z},\boldsymbol{\theta}|\mathcal{X})} = KL(Q||P)\end{aligned}$$

Conjugate-Exponential models

Let's focus on *conjugate-exponential* (**CE**) latent-variable models:

- **Condition (1)**. The joint probability over *variables* is in the exponential family:

  $$P(\mathcal{Z},\mathcal{X}|\boldsymbol{\theta}) = f(\mathcal{Z},\mathcal{X})\, g(\boldsymbol{\theta}) \exp\left\{\phi(\boldsymbol{\theta})^{\mathsf{T}} \mathsf{T}(\mathcal{Z},\mathcal{X})\right\}$$

  where $\phi(\boldsymbol{\theta})$ is the vector of *natural parameters*, T are *sufficient statistics*

- **Condition (2)**. The prior over *parameters* is conjugate to this joint probability:

  $$P(\boldsymbol{\theta}|\nu,\tau) = h(\nu,\tau)\, g(\boldsymbol{\theta})^{\nu} \exp\left\{\phi(\boldsymbol{\theta})^{\mathsf{T}}\tau\right\}$$

  where ν and τ are hyperparameters of the prior.

Conjugate priors are computationally convenient and have an intuitive interpretation:

- ν: number of pseudo-observations
- τ: values of pseudo-observations

Conjugate-Exponential examples

In the **CE** family:

- Gaussian mixtures
- factor analysis, probabilistic PCA
- hidden Markov models and factorial HMMs
- linear dynamical systems and switching models
- discrete-variable belief networks

Other as yet undreamt-of models combinations of Gaussian, Gamma, Poisson, Dirichlet, Wishart, Multinomial and others.

Not in the **CE** family:

- Boltzmann machines, MRFs (no simple conjugacy)
- logistic regression (no simple conjugacy)
- sigmoid belief networks (not exponential)
- independent components analysis (not exponential)

Note: one can often approximate such models with a suitable choice from the **CE** family.

Conjugate-exponential VB

Given an iid data set $\mathcal{D} = (\mathbf{x}_1, \ldots \mathbf{x}_n)$, if the model is **CE** then:

- $Q_\theta(\theta)$ is also conjugate, *i.e.*

$$\begin{aligned} Q_\theta(\theta) &\propto P(\theta) \quad \exp\left\langle \sum_i \log P(\mathbf{z}_i, \mathbf{x}_i|\theta) \right\rangle_{Q_\mathcal{Z}} \\ &= h(\nu, \tau) g(\theta)^\nu e^{\phi(\theta)^\mathsf{T} \tau} \quad g(\theta)^n e^{\left\langle \log f(\mathcal{Z}, \mathcal{X}) \right\rangle_{Q_\mathcal{Z}}} e^{\phi(\theta)^\mathsf{T} \left\langle \sum_i \mathsf{T}(\mathbf{z}_i, \mathbf{x}_i) \right\rangle_{Q_\mathcal{Z}}} \\ &\propto h(\tilde{\nu}, \tilde{\tau}) g(\theta)^{\tilde{\nu}} e^{\phi(\theta)^\mathsf{T} \tilde{\tau}} \end{aligned}$$

  with $\tilde{\nu} = \nu + n$ and $\tilde{\tau} = \tau + \sum_i \langle \mathsf{T}(\mathbf{z}_i, \mathbf{x}_i) \rangle_{Q_\mathcal{Z}}$ $\Rightarrow$ only need to track $\tilde{\nu}, \tilde{\tau}$.

- $Q_\mathcal{Z}(\mathcal{Z}) = \prod_{i=1}^n Q_{\mathbf{z}_i}(\mathbf{z}_i)$ takes the same form as in the E-step of regular EM

$$\begin{aligned} Q_{\mathbf{z}_i}(\mathbf{z}_i) &\propto \exp \langle \log P(\mathbf{z}_i, \mathbf{x}_i|\theta) \rangle_{Q_\theta} \\ &\propto f(\mathbf{z}_i, \mathbf{x}_i) e^{\langle \phi(\theta) \rangle_{Q_\theta}^\mathsf{T} \mathsf{T}(\mathbf{z}_i, \mathbf{x}_i)} = P(\mathbf{z}_i|\mathbf{x}_i, \bar{\phi}(\theta)) \end{aligned}$$

  with natural parameters $\bar{\phi}(\theta) = \langle \phi(\theta) \rangle_{Q_\theta}$ $\Rightarrow$ inference unchanged from regular EM.

The Variational Bayesian EM algorithm

EM for MAP estimation

Goal: maximize $P(\theta|\mathcal{X}, m)$ wrt θ

E Step: compute

$Q_\mathcal{Z}(\mathcal{Z}) \leftarrow p(\mathcal{Z}|\mathcal{X}, \theta)$

M Step:

$\theta \leftarrow \underset{\theta}{\text{argmax}} \int d\mathcal{Z}\, Q_\mathcal{Z}(\mathcal{Z}) \log P(\mathcal{Z}, \mathcal{X}, \theta)$

Variational Bayesian EM

Goal: maximise bound on $P(\mathcal{X}|m)$ wrt Q_θ

VB-E Step: compute

$Q_\mathcal{Z}(\mathcal{Z}) \leftarrow p(\mathcal{Z}|\mathcal{X}, \bar{\phi})$

VB-M Step:

$Q_\theta(\theta) \leftarrow \exp \int d\mathcal{Z}\, Q_\mathcal{Z}(\mathcal{Z}) \log P(\mathcal{Z}, \mathcal{X}, \theta)$

Properties:

- Reduces to the EM algorithm if $Q_\theta(\theta) = \delta(\theta - \theta^*)$.
- $\mathcal{F}_m$ increases monotonically, and incorporates the model complexity penalty.
- Analytical parameter distributions (but not constrained to be Gaussian).
- VB-E step has same complexity as corresponding E step.
- We can use the junction tree, belief propagation, Kalman filter, etc, algorithms in the VB-E step of VB-EM, but **using expected natural parameters**, $\bar{\phi}$.

VB and model selection

- Variational Bayesian EM yields an approximate posterior Q_θ over model parameters.
- It also yields an optimised lower bound on the model evidence

$$\max \mathcal{F}_\mathcal{M}(Q_\mathcal{Z}, Q_\theta) \leq P(\mathcal{D}|\mathcal{M})$$

- These lower bounds can be compared amongst models to learn the right (structure, connectivity ... of the) model
- If a continuous domain of models is specified by a hyperparameter η, then the VB free energy depends on that parameter:

$$\mathcal{F}(Q_\mathcal{Z}, Q_\theta, \eta) = \iint d\mathcal{Z}\, d\theta\, Q_\mathcal{Z}(\mathcal{Z}) Q_\theta(\theta) \log \frac{P(\mathcal{X}, \mathcal{Z}, \theta|\eta)}{Q_\mathcal{Z}(\mathcal{Z}) Q_\theta(\theta)} \leq P(\mathcal{X}|\eta)$$

  A hyper-M step maximises the current bound wrt η:

$$\eta \leftarrow \underset{\eta}{\text{argmax}} \iint d\mathcal{Z}\, d\theta\, Q_\mathcal{Z}(\mathcal{Z}) Q_\theta(\theta) \log P(\mathcal{X}, \mathcal{Z}, \theta|\eta)$$

ARD for unsupervised learning

Recall that ARD (automatic relevance determination) was a hyperparameter method to select relevant or useful inputs in regression.

- A similar idea used with variational Bayesian methods can learn a latent dimensionality.
- Consider factor analysis:

$$\mathbf{x} \sim \mathcal{N}(\Lambda \mathbf{z}, \Psi) \qquad \mathbf{z} \sim \mathcal{N}(0, I) \qquad \text{with a column-wise prior} \quad \Lambda_{:i} \sim \mathcal{N}\left(0, \alpha_i^{-1} I\right)$$

- The VB free energy is

$$\mathcal{F}(Q_\mathcal{Z}(\mathcal{Z}), Q_\Lambda(\Lambda), \Psi, \alpha) = \left\langle \log P(\mathcal{X}, \mathcal{Z}|\Lambda, \Psi) + \log P(\Lambda|\alpha) + \log P(\Psi) \right\rangle_{Q_\mathcal{Z} Q_\Lambda} + \ldots$$

  and so hyperparameter optimisation requires

$$\alpha \leftarrow \text{argmax} \langle \log P(\Lambda|\alpha) \rangle_{Q_\Lambda}$$

- Now Q_Λ is Gaussian, with the same form as in linear regression, but with expected moments of $\mathbf{z}$ appearing in place of the inputs.
- Optimisation wrt the distributions, Ψ and α in turn causes some α_i to diverge as in regression ARD.
- In this case, these parameters select "relevant" latent dimensions, effectively learning the dimensionality of $\mathbf{z}$.

Augmented Variational Methods

In our examples so far, the approximate variational distribution has been over the "natural" latent variables (and parameters) of the generative model.

Sometimes it may be useful to introduce additional latent variables, solely to achieve computational tractability.

Two examples are GP regression and the GPLVM.

Sparse GP approximations

GP predictions:

$$y'|X, Y, \mathbf{x}' \sim \mathcal{N}\left(K_{\mathbf{x}'X}(K_{XX}+\sigma^2 I)^{-1}Y, K_{\mathbf{x}'\mathbf{x}'} - K_{\mathbf{x}'X}(K_{XX}+\sigma^2 I)^{-1}K_{X\mathbf{x}'} + \sigma^2\right)$$

Evidence (for learning kernel hyperparameters):

$$\log P(Y|X) = -\frac{1}{2}\log|2\pi(K_{XX}+\sigma^2 I)| - \frac{1}{2}Y(K_{XX}+\sigma^2 I)^{-1}Y^{\mathsf{T}}$$

Computing either form requires inverting the $N \times N$ matrix K_{XX}, in $\mathcal{O}(N^3)$ time.

One proposal to make this more efficient is to find (or select) a smaller set of possibly fictitious measurements U at inputs Z such that

$$P(y'|Z, U, \mathbf{x}') \approx P(y'|X, Y, \mathbf{x}')\,.$$

What values should U and Z take?

Variational Sparse GP approximations

Write F for the (smooth) GP function values that underlie Y (so $Y \sim \mathcal{N}(F, \sigma^2 I)$).

Introduce latent measurements U at inputs Z (and integrate over U).

The likelihood can be written

$$P(Y|X) = \iint dF\, dU\, P(Y, F, U|X, Z) = \iint dF\, dU\, P(Y|F)P(F|U, X, Z)P(U|Z)$$

Now, both U and F are latent, so we introduce a variational distribution $q(F, U)$ to form a free-energy.

$$\mathcal{F}(q(F,U),\theta) = \left\langle \log \frac{P(Y|F)P(F|U,X,Z)P(U|Z)}{q(F,U)} \right\rangle_{q(F,U)}$$

Now, choose the variational form $q(F, U) = P(F|U, X, Z)q(U)$. That is, fix $F|U$ without reference to Y – so information about Y will need to be "compressed" into $q(U)$.
Then

$$\begin{aligned}\mathcal{F}(q(F,U),\theta,Z) &= \left\langle \log \frac{P(Y|F)\ \cancel{P(F|U,X,Z)}\ P(U|Z)}{\cancel{P(F|U,X,Z)}\ q(U)} \right\rangle_{P(F|U)q(U)} \\ &= \left\langle \langle \log P(Y|F)\rangle_{P(F|U)} + \log P(U|Z) - \log q(U) \right\rangle_{q(U)}\end{aligned}$$

Variational Sparse GP approximations

$$\mathcal{F}(q(U),\theta,Z) = \left\langle \langle \log P(Y|F)\rangle_{P(F|U)} + \log P(U|Z) - \log q(U) \right\rangle_{q(U)}$$

Now $P(F|U)$ is fixed by the generative model (rather than being subject to free optimisation). So we can evaluate that expectation:

$$\begin{aligned}&\langle \log P(Y|F)\rangle_{P(F|U)} \\ &= \left\langle -\frac{1}{2}\log\left|2\pi\sigma^2 I\right| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[(Y-F)(Y-F)^{\mathsf{T}}\right] \right\rangle_{P(F|U)} \\ &= -\frac{1}{2}\log\left|2\pi\sigma^2 I\right| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[(Y-\langle F\rangle_{P(F|U)})(Y-\langle F\rangle_{P(F|U)})^{\mathsf{T}}\right] - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\Sigma_{F|U}\right] \\ &= \log \mathcal{N}\left(Y|K_{XZ}K_{ZZ}^{-1}U, \sigma^2 I\right) - \frac{1}{2\sigma^2}\mathrm{Tr}\left[K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{ZX}\right]\end{aligned}$$

So,

$$\begin{aligned}\mathcal{F}(q(U),\theta,Z) = \big\langle \log \mathcal{N}\left(Y|K_{XZ}K_{ZZ}^{-1}U, \sigma^2 I\right) + \log P(U|Z) - \log q(U)\big\rangle_{q(U)} \\ - \frac{1}{2\sigma^2}\mathrm{Tr}\left[K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{ZX}\right]\,.\end{aligned}$$

Variational Sparse GP approximations

$$\mathcal{F}(q(U),\theta,Z) = \left\langle \log \frac{\mathcal{N}\left(Y|K_{XZ}K_{ZZ}^{-1}U,\sigma^2 I\right)P(U|Z)}{q(U)} \right\rangle_{q(U)} - \frac{1}{2\sigma^2}\mathrm{Tr}\left[K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{ZX}\right] .$$

The expectation is the free energy of a PPCA-like model with normal prior $U \sim \mathcal{N}(0, K_{ZZ})$ and loading matrix $K_{XZ}K_{ZZ}^{-1}$. The maximum of this free energy is the log-likelihood (achieved with q equal to the posterior under the PPCA-like model).
This gives

$$\mathcal{F}(q^*(U),\theta,Z) = \log \mathcal{N}\left(Y|0, K_{XZ}\cancel{K_{ZZ}^{-1}K_{ZZ}}K_{ZZ}^{-1}K_{ZX} + \sigma^2 I\right) - \frac{1}{2\sigma^2}\mathrm{Tr}\left[K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{ZX}\right] .$$

Note that we have eliminated all terms in K_{XX}^{-1}.

We can optimise the free energy numerically with respect to Z and θ to adjust the GP prior and quality of variational approximation.

A similar approach can be used to learn X if they are unobserved (*i.e.* in the GPLVM). Assume $q(X,F,U) = q(X)P(F|X,U)q(U)$. Then $\mathcal{F} = \langle \log P(Y,F,U|X) \log P(X)\rangle_{q(U)q(X)}$ which simplifies into tractable components in much the same way as above.

A few references

- Jordan, Ghahramani, Jaakkola, Saul, 1999. An introduction to variational methods for graphical models. *Machine Learning* **37**:183–233.
- Attias, 2000. A variational Bayesian framework for graphical models. *NIPS 12*. `http://www.gatsby.ucl.ac.uk/publications/papers/03-2000.ps`
- Beal, 2003. Variational algorithms for approximate Bayesian inference. *PhD thesis*, Gatsby Unit, UCL. `http://www.cse.buffalo.edu/faculty/mbeal/thesis/`
- Winn, 2003. Variational message passing and its applications. *PhD thesis*, Cambridge. `http://johnwinn.org/Publications/Thesis.html`; also VIBES software for conjugate-exponential graphs.

Some complexities:

- MacKay, 2001. A problem with variational free energy minimization. `http://www.inference.phy.cam.ac.uk/mackay/minima.pdf`
- Turner, MS, 2011. Two problems with variational expectation maximisation for time-series models. In Barber, Cemgil, Chiappa, eds., *Bayesian Time Series Models*. `http://www.gatsby.ucl.ac.uk/~maneesh/papers/turner-sahani-2010-ildn.pdf`
- Berkes, Turner, MS, 2008. On sparsity and overcompleteness in image models. *NIPS 20*. `http://www.gatsby.ucl.ac.uk/~maneesh/papers/berkes-etal-2008-nips.pdf`
- Giordano, Broderick, Jordan, 2015. Linear response methods for accurate covariance estimates from mean field variational Bayes. *NIPS*

Probabilistic & Unsupervised Learning
Approximate Inference

Expectation Propagation

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

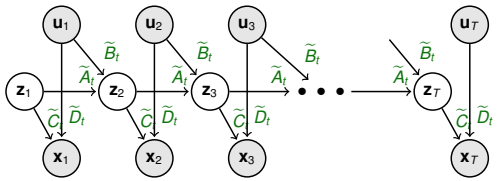
Term 1, Autumn 2023

Intractabilities and approximations

- Inference – computational intractability
  - Gibbs sampling, other MCMC
  - Factored variational approx
  - Loopy BP/EP/Power EP
  - Recognition models
- Inference – analytic intractability
  - Laplace approximation (global)
  - (Sequential) Monte-Carlo
  - Message approximations (linearised, sigma-point, Laplace)
  - Assumed-density methods and Expectation-Propagation
  - Parametric variational approx
  - Recognition models
- Learning – intractable partition function
  - Sampling parameters
  - Constrastive divergence
  - Score-matching
- Posterior estimation and model selection
  - Laplace approximation / BIC
  - Monte-Carlo
  - (Annealed) importance sampling
  - Reversible jump MCMC
  - Variational Bayes

Not a complete list!

Nonlinear state-space model (NLSSM)



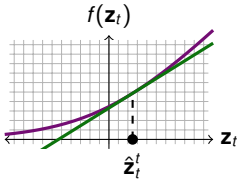
$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{z}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

Extended Kalman Filter (EKF): linearise nonlinear functions about current estimate, $\hat{\mathbf{z}}_t^t$:

$$\mathbf{z}_{t+1} \approx \underbrace{f(\hat{\mathbf{z}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{z}_t}\right|_{\hat{\mathbf{z}}_t^t}}_{\widetilde{A}_t} (\mathbf{z}_t - \hat{\mathbf{z}}_t^t) + \mathbf{w}_t$$

$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{z}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{z}_t}\right|_{\hat{\mathbf{z}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{z}_t - \hat{\mathbf{z}}_t^{t-1}) + \mathbf{v}_t$$



Run the Kalman filter (smoother) on non-stationary linearised system ($\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t$):

- Adaptively approximates non-Gaussian messages by Gaussians.
- Local linearisation depends on central point of distribution ⇒ approximation degrades with increased state uncertainty. May work acceptably for close-to-linear systems.

Can base EM-like algorithm on EKF/EKS (or alternatives).

Other message approximations

Consider the forward messages on a latent chain:

$$P(\mathbf{z}_t|\mathbf{x}_{1:t}) = \frac{1}{Z} P(\mathbf{x}_t|\mathbf{z}_t) \int d\mathbf{z}_{t-1}\; P(\mathbf{z}_t|\mathbf{z}_{t-1}) P(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})$$

We want to approximate the messages to retain a tractable form (e.g. Gaussian).

$$\tilde{P}(\mathbf{z}_t|\mathbf{x}_{1:t}) \approx \frac{1}{Z} P(\mathbf{x}_t|\mathbf{z}_t) \int d\mathbf{z}_{t-1} \quad \underbrace{P(\mathbf{z}_t|\mathbf{z}_{t-1})}_{\mathcal{N}(f(\mathbf{z}_{t-1}), Q)} \quad \underbrace{\tilde{P}(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})}_{\mathcal{N}(\hat{\mathbf{z}}_{t-1}, V_{t-1})}$$

- Linearisation at the peak (EKF) is only one approach.
- Laplace filter: use mode and curvature of integrand.
- Sigma-point ("unscented") filter: next slide.
- Parametric variational:

$$\operatorname{argmin} \mathbf{KL}\left[\mathcal{N}(\hat{\mathbf{z}}_t, \hat{V}_t) \,\middle\|\, \int d\mathbf{z}_{t-1} \ldots\right].$$

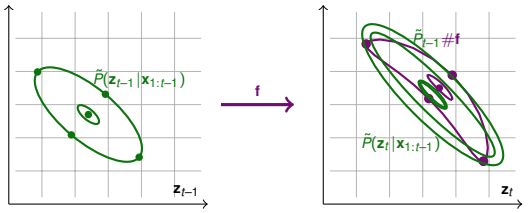
  Needs Gaussian expectations of log $\int$ ⇒ Monte-Carlo integration (later lecture).
- The other KL:

$$\operatorname{argmin} \mathbf{KL}\left[\int d\mathbf{z}_{t-1} \,\middle\|\, \mathcal{N}(\hat{\mathbf{z}}_t, \hat{V}_t)\right]$$

  needs only first and second moments of nonlinear message ⇒ EP.

The Sigma-point filter

- Historical interest, but also a useful intuition for what comes next.



- Approximates pushed-forward belief from time t–1.
- Evaluate $\mathbf{f}(\hat{\mathbf{z}}_{t-1})$, $\mathbf{f}(\hat{\mathbf{z}}_{t-1} \pm \sqrt{\lambda}\mathbf{v})$ for eigenvalues, eigenvectors $\hat{V}_{t-1}\mathbf{v} = \lambda\mathbf{v}$.
- "Fit" Gaussian to these $2K+1$ $\boldsymbol{\sigma}-$points:

$$\mathcal{N}\Big(\underbrace{\tfrac{1}{2K+1}\sum_{i=1}^{2K+1}\mathbf{f}(\boldsymbol{\sigma}_i)}_{\boldsymbol{\mu}}, \tfrac{1}{2K+1}\sum_{i=1}^{2K+1}\mathbf{f}(\boldsymbol{\sigma}_i)\mathbf{f}(\boldsymbol{\sigma}_i)^\mathsf{T} - \boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T} + \mathsf{Q}\Big)$$

- Incorporate noise process.
- Equivalent to evaluation of mean and covariance of $\tilde{P}_{t-1}\#\mathbf{f}$ by Gaussian quadrature.
- One form of "Assumed Density Filtering" (and of calculations for EP).

Approximating the posterior

Linearisation (or local Laplace, sigma-point and other such approaches) seem *ad hoc*. A more principled approach might look for an approximate q that is closest to P in some sense.

$$q = \operatorname*{argmin}_{q\in\mathcal{Q}} D(P \leftrightarrow q)$$

Open choices:

- form of the metric D
- nature of the constraint space $\mathcal{Q}$

- Variational methods: $D = \mathbf{KL}[q\|P]$.
  - Choosing $\mathcal{Q}$ = {tree-factored distributions} leads to efficient message passing.
- Can we use other divergences?

Variational learning

Free energy:

$$\mathcal{F}(q,\theta) = \langle \log P(\mathcal{X},\mathcal{Z}|\theta)\rangle_{q(\mathcal{Z}|\mathcal{X})} + \mathbf{H}[q] = \log P(\mathcal{X}|\theta) - \mathbf{KL}[q(\mathcal{Z})\|P(\mathcal{Z}|\mathcal{X},\theta)] \leq \ell(\theta)$$

E-steps:

- Exact EM: $q(\mathcal{Z}) = \operatorname*{argmax}_q \mathcal{F} = P(\mathcal{Z}|\mathcal{X},\theta)$
  - Saturates bound: converges to local maximum of likelihood.
- (Factored) variational approximation:

  $$q(\mathcal{Z}) = \operatorname*{argmax}_{q_1(\mathcal{Z}_1)q_2(\mathcal{Z}_2)} \mathcal{F} = \operatorname*{argmin}_{q_1(\mathcal{Z}_1)q_2(\mathcal{Z}_2)} \mathbf{KL}[q_1(\mathcal{Z}_1)q_2(\mathcal{Z}_2)\|P(\mathcal{Z}|\mathcal{X},\theta)]$$

  - Increases bound: converges, but not necessarily to ML.
- Other approximations: $q(\mathcal{Z}) \approx P(\mathcal{Z}|\mathcal{X},\theta)$
  - Usually no guarantees, but if learning converges it may be more accurate than the factored approximation

The other KL

What about the 'other' KL ($q = \operatorname{argmin} \mathbf{KL}[P\|q]$)?

For a factored approximation the (clique) marginals obtained by minimising this KL are correct:

$$\begin{aligned}
\operatorname*{argmin}_{q_i} \mathbf{KL}\Big[P(\mathcal{Z}|\mathcal{X})\Big\|\prod q_j(\mathcal{Z}_j|\mathcal{X})\Big] &= \operatorname*{argmin}_{q_i} -\int d\mathcal{Z}\; P(\mathcal{Z}|\mathcal{X}) \log \prod_j q_j(\mathcal{Z}_j|\mathcal{X}) \\
&= \operatorname*{argmin}_{q_i} -\sum_j \int d\mathcal{Z}\; P(\mathcal{Z}|\mathcal{X}) \log q_j(\mathcal{Z}_j|\mathcal{X}) \\
&= \operatorname*{argmin}_{q_i} -\int d\mathcal{Z}_i\; P(\mathcal{Z}_i|\mathcal{X}) \log q_i(\mathcal{Z}_i|\mathcal{X}) \\
&= P(\mathcal{Z}_i|\mathcal{X})
\end{aligned}$$

and the marginals are what we need for learning (although if factored over disjoint sets as in the variational approximation some cliques will be missing).

Perversely, this means finding the best q for this KL is intractable!

But it raises the hope that approximate minimisation might still yield useful results.

Approximate optimisation

The posterior distribution in a graphical model is a (normalised) product of factors:

$$P(\mathcal{Z}|\mathcal{X}) = \frac{P(\mathcal{Z},\mathcal{X})}{P(\mathcal{X})} = \frac{1}{Z}\prod_i P(Z_i|\,\mathrm{pa}(Z_i)) \propto \prod_{i=1}^{N} f_i(\mathcal{Z}_i)$$

where the $\mathcal{Z}_i$ are not necessarily disjoint. In the language of EP the f_i are called sites.

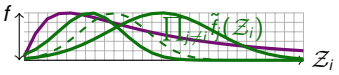
Consider q with the same factorisation, but potentially approximated sites: $q(\mathcal{Z}) \propto \prod_{i=1}^{N} \tilde{f}_i(\mathcal{Z}_i)$.

We would like to minimise (at least in some sense) $\mathbf{KL}[P\|q]$.
Possible optimisations:

$$\min_{\{\tilde{f}_i\}} \mathbf{KL}\left[\frac{1}{Z}\prod_{i=1}^{N} f_i(\mathcal{Z}_i)\middle\|\frac{1}{\tilde{Z}}\prod_{i=1}^{N}\tilde{f}_i(\mathcal{Z}_i)\right] \qquad \text{(global: intractable)}$$

$$\min_{\tilde{f}_i} \mathbf{KL}\left[f_i(\mathcal{Z}_i)\middle\|\tilde{f}_i(\mathcal{Z}_i)\right] \qquad \text{(local, fixed: simple, inaccurate)}$$

$$\min_{\tilde{f}_i} \mathbf{KL}\left[f_i(\mathcal{Z}_i)\prod_{j\neq i}\tilde{f}_j(\mathcal{Z}_j)\middle\|\tilde{f}_i(\mathcal{Z}_i)\prod_{j\neq i}\tilde{f}_j(\mathcal{Z}_j)\right] \qquad \text{(local, contextual: iterative, accurate)} \leftarrow \text{EP}$$



Expectation? Propagation?

EP is really two ideas:

- Approximation of factors.
  - Usually by "projection" to exponential families.
  - This involves finding expected sufficient statistics, hence expectation.
- Local divergence minimization in the context of other factors.
  - This leads to a message passing approach, hence propagation.

Note: we will ignore normalisation for now, but return to this later.

Local updates

Each EP update involves a KL minimisation:

$$\tilde{f}_i^{\mathrm{new}}(\mathcal{Z}) \leftarrow \operatorname*{argmin}_{f\in\{\tilde{f}\}} \mathbf{KL}[f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})\|f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})] \qquad \left[q_{\neg i}(\mathcal{Z}) \stackrel{\mathrm{def}}{=} \prod_{j\neq i}\tilde{f}_j(\mathcal{Z}_j)\right]$$

Separate the contextual factor: $q_{\neg i}(\mathcal{Z}) = q_{\neg i}(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_{\neg i}|\mathcal{Z}_i)$ $\qquad [\mathcal{Z}_{\neg i} \stackrel{\mathrm{def}}{=} \mathcal{Z}\backslash\mathcal{Z}_i]$

Then:

$$\begin{aligned}
&\min_f \mathbf{KL}[f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})\|f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})] \\
&\quad = \max_f \int d\mathcal{Z}\, f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}) \log f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}) \\
&\quad = \max_f \int d\mathcal{Z}_i d\mathcal{Z}_{\neg i}\, f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_{\neg i}|\mathcal{Z}_i)\big(\log f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i) + \log q_{\neg i}(\mathcal{Z}_{\neg i}|\mathcal{Z}_i)\big) \\
&\quad = \max_f \int d\mathcal{Z}_i\, f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)\big(\log f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)\big)\int d\mathcal{Z}_{\neg i}\, q_{\neg i}(\mathcal{Z}_{\neg i}|\mathcal{Z}_i) \\
&\quad = \min_f \mathbf{KL}[f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)\|f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)]
\end{aligned}$$

$q_{\neg i}(\mathcal{Z}_i)$ is sometimes called the cavity distribution.

Expectation Propagation (EP)

Input $f_1(\mathcal{Z}_1)\ldots f_N(\mathcal{Z}_N)$

Initialize $\tilde{f}_1(\mathcal{Z}_1) = \operatorname*{argmin}_{f\in\{\tilde{f}\}} \mathbf{KL}[f_1(\mathcal{Z}_1)\|f_1(\mathcal{Z}_1)]$, $\tilde{f}_i(\mathcal{Z}_i) = 1$ for $i > 1$, $q(\mathcal{Z}) \propto \prod_i \tilde{f}_i(\mathcal{Z}_i)$

repeat

  for $i = 1\ldots N$ **do**

    Delete: $q_{\neg i}(\mathcal{Z}) \leftarrow \dfrac{q(\mathcal{Z})}{\tilde{f}_i(\mathcal{Z}_i)} = \prod_{j\neq i}\tilde{f}_j(\mathcal{Z}_j)$

    Project: $\tilde{f}_i^{\mathrm{new}}(\mathcal{Z}) \leftarrow \operatorname*{argmin}_{f\in\{\tilde{f}\}} \mathbf{KL}[f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)\|f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}_i)]$

    Include: $q(\mathcal{Z}) \leftarrow \tilde{f}_i^{\mathrm{new}}(\mathcal{Z}_i)\, q_{\neg i}(\mathcal{Z})$

  end for

until convergence

Message Passing

- The cavity distribution (in a tree) can be further broken down into a product of terms from each neighbouring clique:

$$q_{\neg i}(\mathcal{Z}_i) = \prod_{j \in \mathrm{ne}(i)} M_{j \to i}(\mathcal{Z}_j \cap \mathcal{Z}_i)$$

- Once the ith site has been approximated, the messages can be passed on to neighbouring cliques by marginalising to the shared variables (SSM example follows). $\Rightarrow$ belief propagation.

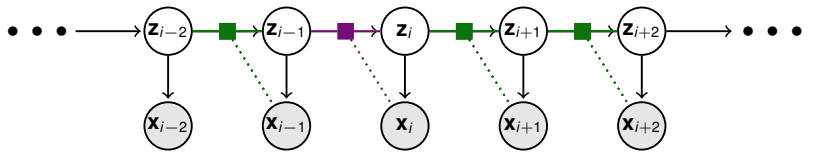
- In loopy graphs, we can use loopy belief propagation. In that case

$$q_{\neg i}(\mathcal{Z}_i) = \prod_{j \in \mathrm{ne}(i)} M_{j \to i}(\mathcal{Z}_j \cap \mathcal{Z}_i)$$

becomes an approximation to the **true** cavity distribution (or we can recast the approximation directly in terms of messages $\Rightarrow$ later lecture).

- For some approximations (e.g. Gaussian) may be able to compute true loopy cavity using approximate sites, even if computing exact message would have been intractable.
- In either case, message updates can be scheduled in any order.
- No guarantee of convergence (but see "power-EP" methods).

EP for a NLSSM



$$P(\mathbf{z}_i|\mathbf{z}_{i-1}) = \phi_i(\mathbf{z}_i, \mathbf{z}_{i-1}) \qquad \textit{e.g. } \exp(-\|\mathbf{z}_i - h_s(\mathbf{z}_{i-1})\|^2/2\sigma^2)$$
$$P(\mathbf{x}_i|\mathbf{z}_i) = \psi_i(\mathbf{z}_i) \qquad \textit{e.g. } \exp(-\|\mathbf{x}_i - h_o(\mathbf{z}_i)\|^2/2\sigma^2)$$

Then $f_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \phi_i(\mathbf{z}_i, \mathbf{z}_{i-1})\psi_i(\mathbf{z}_i)$. As ϕ_i and ψ_i are non-linear, inference is not generally tractable.
Assume $\tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1})$ is Gaussian. Then,

$$q_{\neg i}(\mathbf{z}_i, \mathbf{z}_{i-1}) = \int_{\substack{\mathbf{z}_1 \ldots \mathbf{z}_{i-2} \\ \mathbf{z}_{i+1} \ldots \mathbf{z}_i}} \prod_{i' \neq i} \tilde{f}_{i'}(\mathbf{z}_{i'}, \mathbf{z}_{i'-1}) = \underbrace{\int_{\mathbf{z}_1 \ldots \mathbf{z}_{i-2}} \prod_{i' < i} \tilde{f}_{i'}(\mathbf{z}_{i'}, \mathbf{z}_{i'-1})}_{\alpha_{i-1}(\mathbf{z}_{i-1})} \underbrace{\int_{\mathbf{z}_{i+1} \ldots \mathbf{z}_n} \prod_{i' > i} \tilde{f}_{i'}(\mathbf{z}_{i'}, \mathbf{z}_{i'-1})}_{\beta_i(\mathbf{z}_i)}$$

with both α and β Gaussian.

$$\tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \operatorname*{argmin}_{f \in \mathcal{N}} \mathbf{KL}\left[\phi_i(\mathbf{z}_i, \mathbf{z}_{i-1})\psi_i(\mathbf{z}_i)\alpha_{i-1}(\mathbf{z}_{i-1})\beta_i(\mathbf{z}_i) \middle\| f(\mathbf{z}_i, \mathbf{z}_{i-1})\alpha_{i-1}(\mathbf{z}_{i-1})\beta_i(\mathbf{z}_i)\right]$$

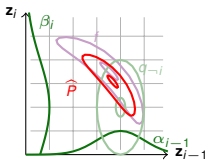
NLSSM EP message updates

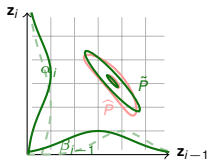
$$\tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \operatorname*{argmin}_{f \in \mathcal{N}} \mathbf{KL}\left[f(\mathbf{z}_i, \mathbf{z}_{i-1})q_{\neg i}(\mathbf{z}_i, \mathbf{z}_{i-1}) \middle\| f(\mathbf{z}_i, \mathbf{z}_{i-1})q_{\neg i}(\mathbf{z}_i, \mathbf{z}_{i-1})\right] = \operatorname*{argmin}_{f \in \mathcal{N}} \mathbf{KL}\Big[\underbrace{\phi_i(\mathbf{z}_i, \mathbf{z}_{i-1})\psi_i(\mathbf{z}_i)\alpha_{i-1}}_{\widehat{P}(\mathbf{z}_{i-1}, \mathbf{z}_i)}$$

$$\tilde{P}(\mathbf{z}_{i-1}, \mathbf{z}_i) = \operatorname*{argmin}_{P \in \mathcal{N}} \mathbf{KL}\left[\widehat{P}(\mathbf{z}_{i-1}, \mathbf{z}_i) \middle\| P(\mathbf{z}_{i-1}, \mathbf{z}_i)\right] \qquad \tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \frac{\tilde{P}(\mathbf{z}_{i-1}, \mathbf{z}_i)}{\alpha_{i-1}(\mathbf{z}_{i-1})\beta_i(\mathbf{z}_i)}$$

$$\alpha_i(\mathbf{z}_i) = \int_{\mathbf{z}_1 \ldots \mathbf{z}_{i-1}} \prod_{i' < i+1} \tilde{f}_{i'}(\mathbf{z}_{i'}, \mathbf{z}_{i'-1}) = \int_{\mathbf{z}_{i-1}} \alpha_{i-1}(\mathbf{z}_{i-1})\tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \frac{1}{\beta_i(\mathbf{z}_i)} \int_{\mathbf{z}_{i-1}} \tilde{P}(\mathbf{z}_{i-1}, \mathbf{z}_i)$$

$$\beta_{i-1}(\mathbf{z}_{i-1}) = \int_{\mathbf{z}_{i+1} \ldots \mathbf{z}_{i'>i}} \prod \tilde{f}_{i'}(\mathbf{z}_{i'}, \mathbf{z}_{i'-1}) = \int_{\mathbf{z}_i} \beta_i(\mathbf{z}_i)\tilde{f}_i(\mathbf{z}_i, \mathbf{z}_{i-1}) = \frac{1}{\alpha_{i-1}(\mathbf{z}_{i-1})} \int_{\mathbf{z}_i} \tilde{P}(\mathbf{z}_{i-1}, \mathbf{z}_i)$$





Moment Matching

Each EP update involves a KL minimisation:

$$\tilde{f}_i^{\mathrm{new}}(\mathcal{Z}) \leftarrow \operatorname*{argmin}_{f \in \{\tilde{f}\}} \mathbf{KL}[f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z}) \| f(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})]$$

Usually, both $q_{\neg i}(\mathcal{Z}_i)$ and $\tilde{f}$ are in the same exponential family. Let $q(x) = \frac{1}{Z(\boldsymbol{\theta})} e^{\mathsf{T}(x)\cdot\boldsymbol{\theta}}$. Then

$$\begin{aligned}
\operatorname*{argmin}_q \mathbf{KL}\left[p(x) \middle\| q(x)\right] &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbf{KL}\left[p(x) \middle\| \frac{1}{Z(\boldsymbol{\theta})} e^{\mathsf{T}(x)\cdot\boldsymbol{\theta}}\right] \\
&= \operatorname*{argmin}_{\boldsymbol{\theta}} - \int dx\ p(x) \log \frac{1}{Z(\boldsymbol{\theta})} e^{\mathsf{T}(x)\cdot\boldsymbol{\theta}} \\
&= \operatorname*{argmin}_{\boldsymbol{\theta}} - \int dx\ p(x)\mathsf{T}(x)\cdot\boldsymbol{\theta} + \log Z(\boldsymbol{\theta}) \\
\frac{\partial}{\partial\boldsymbol{\theta}} &= -\int dx\ p(x)\mathsf{T}(x) + \frac{1}{Z(\boldsymbol{\theta})} \frac{\partial}{\partial\boldsymbol{\theta}} \int dx\ e^{\mathsf{T}(x)\cdot\boldsymbol{\theta}} \\
&= -\langle \mathsf{T}(x) \rangle_p + \frac{1}{Z(\boldsymbol{\theta})} \int dx\ e^{\mathsf{T}(x)\cdot\boldsymbol{\theta}} \mathsf{T}(x) \\
&= -\langle \mathsf{T}(x) \rangle_p + \langle \mathsf{T}(x) \rangle_q
\end{aligned}$$

So minimum is found by matching sufficient stats or moment matching.

Numerical issues

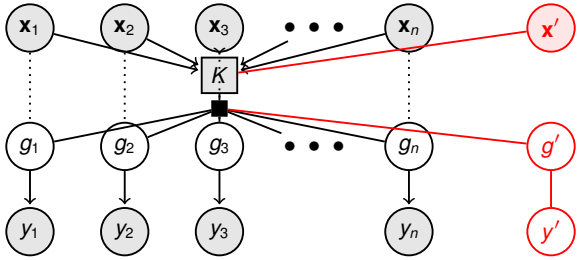
How do we calculate $\langle \mathsf{T}(x) \rangle_{\widehat{P}}$?

Often analytically tractable, but even if not requires a (relatively) low-dimensional integral:

- Quadrature methods.
  - Classical Gaussian quadrature (same Gauss, but nothing to do with the distribution) gives an iterative version of Sigma-point methods.
  - Positive definite joints, but not guaranteed to give positive definite messages.
  - Heuristics include skipping non-positive-definite steps, or damping messages by interpolation or exponentiating to power < 1.
  - Other quadrature approaches (*e.g.* GP quadrature) may be more accurate, and may allow formal constraint to pos-def cone.
- Laplace approximation.
  - Equivalent to Laplace propagation.
  - As long as messages remain positive definite will converge to global Laplace approximation.

EP for Gaussian process classification

EP provides a succesful framework for Gaussian-process modelling of non-Gaussian observations (*e.g.* for classification).

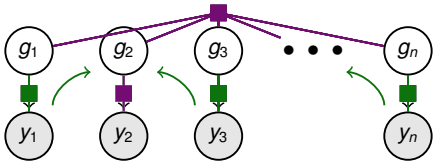


Recall:

- A GP defines a multivariate Gaussian distribution on any finite subset of random vars $\{g_1 \dots g_n\}$ drawn from a (usually uncountable) potential set indexed by "inputs" $\mathbf{x}_i$.
- The Gaussian parameters depend on the inputs: $(\boldsymbol{\mu} = [\mu(\mathbf{x}_i)],\ \Sigma = [K(\mathbf{x}_i, \mathbf{x}_j)])$.
- If we think of the gs as function values, a GP provides a prior over functions.
- In a GP regression model, noisy observations y_i are conditionally independent given g_i.
- No parameters to learn (though often hyperparameters); instead, we make predictions on test data directly: [assuming $\mu = 0$, and matrix Σ incorporates diagonal noise]

$$P(y'|\mathbf{x}', \mathcal{D}) = \mathcal{N}\left(\Sigma_{x',X}\Sigma_{X,X}^{-1}\mathbf{z},\ \Sigma_{x',x'} - \Sigma_{x',X}\Sigma_{X,X}^{-1}\Sigma_{X,x'}\right)$$

GP EP updates



- We can write the GP joint on g_i and y_i as a factor graph:

$$P(g_1 \dots g_n, y_1, \dots y_n) = \underbrace{\mathcal{N}(g_1 \dots g_n | \mathbf{0}, K)}_{f_0(\mathcal{G})} \prod_i \underbrace{\mathcal{N}(y_i | g_i, \sigma_i^2)}_{f_i(g_i)}$$

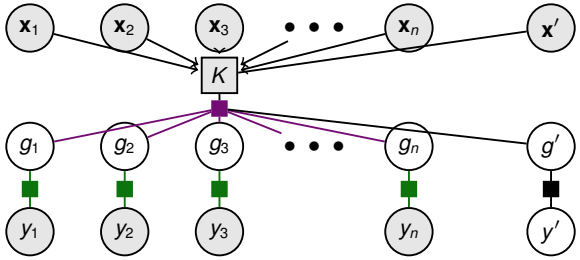
- The same factorisation applies to non-Gaussian $P(y_i|g_i)$ (*e.g.* $P(y_i{=}1) = 1/(1 + e^{-g_i})$).
- EP: approximate non-Gaussian $f_i(g_i)$ by Gaussian $\tilde{f}_i(g_i) = \mathcal{N}\left(\tilde{\mu}_i, \tilde{\psi}_i^2\right)$.
- $q_{\neg i}(g_i)$ can be constructed by the usual GP marginalisation. If $\Sigma = K + \mathrm{diag}\left[\tilde{\psi}_1^2 \dots \tilde{\psi}_n^2\right]$

$$q_{\neg i}(g_i) = \mathcal{N}\left(\Sigma_{i,\neg i}\Sigma_{\neg i,\neg i}^{-1}\tilde{\boldsymbol{\mu}}_{\neg i},\ K_{i,i} - \Sigma_{i,\neg i}\Sigma_{\neg i,\neg i}^{-1}\Sigma_{\neg i,i}\right)$$

- The EP updates thus require calculating Gaussian expectations of $f_i(g)g^{\{1,2\}}$:

$$\tilde{f}_i^{\text{new}}(g_i) = \mathcal{N}\left(\int dg\, q_{\neg i}(g) f_i(g) g,\ \int dg\, q_{\neg i}(g) f_i(g) g^2 - (\tilde{\mu}_i^{\text{new}})^2\right) / q_{\neg i}(g_i)$$

EP GP prediction



- Once appoximate site potentials have stabilised, they can be used to make predictions.
- Introducing a test point changes K, but does not affect the *marginal* $P(g_1 \dots g_n)$ (by consistency of the GP).
- The unobserved output factor provides no information about g' ($\Rightarrow$ constant factor on g')
- Thus no change is needed to the approximating potentials $\tilde{f}_i$.
- Predictions are obtained by marginalising the approximation: [let $\tilde{\Psi} = \mathrm{diag}\left[\tilde{\psi}_1^2 \dots \tilde{\psi}_n^2\right]$]

$$P(y'|\mathbf{x}', \mathcal{D}) = \int dg'\, P(y'|g') \mathcal{N}\Big(g' \mid K_{x',X}(K_{X,X} + \tilde{\Psi})^{-1}\tilde{\boldsymbol{\mu}},\ K_{x',x'} - K_{x',X}(K_{X,X} + \tilde{\Psi})^{-1}K_{X,x'}\Big)$$

Normalisers

- As long as our approximating class is a tractable exponential family, normalisers can be computed as needed.
- Consider an approximating class written

  $$\tilde{f}_i(\mathcal{Z}_i) \propto e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_i - \Phi(\boldsymbol{\theta}_i)}$$

  i.e., define a single sufficient statistic vector on all latents, setting entries in $\boldsymbol{\theta}_i$ to 0 for suff stat functions that take cliques other than $\mathcal{Z}_i$.
- Then

  $$q(\mathcal{Z}) \propto \prod_i \tilde{f}_i \propto e^{T(\mathcal{Z})\cdot\sum\theta_i - \sum\Phi(\theta_i)}$$

  and so we can simply renormalise at the end as usual:

  $$q(\mathcal{Z}) = e^{T(\mathcal{Z})\cdot\sum\theta_i - \Phi(\sum\theta_i)}\,.$$

- However, to compute an approximation to the likelihood $\int d\mathcal{Z}\ \prod_i f_i(\mathcal{Z}_i)$ we need to keep track of the site integrals.

Computing likelihoods – keeping track of normalisers

- Define unnormalised ExpFam approximating sites $\tilde{f}_i = \tilde{C}_i e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_i}$.

  Write $\boldsymbol{\theta} = \sum\boldsymbol{\theta}_j$ for the natural parameters of $q(\mathcal{Z})$ and $\boldsymbol{\theta}_{\neg i} = \sum_{j\neq i}\boldsymbol{\theta}_j$ for the natural parameters of $q_{\neg i}(\mathcal{Z})$.

  Let $\Phi(\boldsymbol{\theta}) = \log\int e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}}$ be the (tractable) ExpFam log normaliser.
- Now, at each EP step minimise the "unnormalised KL":

  $$\mathbf{KL}[p\|q] = \int dx\, p(x)\log\frac{p(x)}{q(x)} + \int dx\,\left(q(x) - p(x)\right)$$

  This matches the zeroth moment of $f_i(\mathcal{Z}_i)q_{\neg i}(\mathcal{Z})$ as well as the expected sufficient statistics as before. That is:

  $$\int \tilde{C}_i e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_i}\prod_{\neg i}\tilde{C}_j e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_j} = \int f_i(\mathcal{Z}_i)\prod_{\neg i}\tilde{C}_j e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_j} \quad\Rightarrow\quad \tilde{C}_i = e^{\Phi_i(\boldsymbol{\theta}_{\neg i})-\Phi(\boldsymbol{\theta})}$$

  where Φ_i is the log-normaliser of the "tilted" ExpFam $\widehat{P}_i(\mathcal{Z}) \propto f(\mathcal{Z}_i)e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}}$.
- The likelihood approximation is then:

  $$\log\int\prod_{i=1}^N f_i(\mathcal{Z}_i) \approx \log\int\prod_{i=1}^N\tilde{f}_i(\mathcal{Z}_i) = \Phi(\boldsymbol{\theta}) + \sum\log\tilde{C}_i \stackrel{\text{def}}{=} \tilde{\ell}$$

Learning

EP yields approximate *inferential* posteriors. To learn (hyper)parameters we can use:

- Approximate Bayesian inference (analagous to VB)
  - may be difficult to construct a coherent normalisable exponential family approximation on both latents and parameters.
- Approximate EM – maximize $\langle\log P(\mathcal{X},\mathcal{Z})\rangle_{q_{EP}(\mathcal{Z})}$.
  - Practical, but no coherent cost function (unlike variational inference), so no guarantee of convergence even if EP itself converges.
- Direct maximisation of EP log-likelihood estimate.
  - Consistent, although convergence guarantees still difficult.
  - Seems challenging as we need to differentiate through (iteration-based) dependence of approximate $q(\mathcal{Z})$ and $\tilde{C}_i$s.
  - However, proves to be simpler than it sounds.

EP log-likelihood optimisation for learning

Let true potentials f_i depend on model (hyper)parameters η.
We have

$$\nabla_\eta\tilde{\ell} = \nabla_\eta\Phi(\boldsymbol{\theta}) + \sum_{i=1}^N\nabla_\eta\log\tilde{C}_i = \boldsymbol{\mu}\cdot\nabla_\eta\boldsymbol{\theta} + \sum_{i=1}^N\nabla_\eta\log\tilde{C}_i \qquad (*)$$

using the standard ExpFam moment-generating result with mean parameters $\boldsymbol{\mu} = \langle T(\mathcal{Z})\rangle_{q(\mathcal{Z})}$.
Now, zeroth-moment matching implies that at EP convergence:

$$\log\tilde{C}_i = \Phi_i(\boldsymbol{\theta}_{\neg i}) - \Phi(\boldsymbol{\theta}) \Rightarrow \nabla_\eta\log\tilde{C}_i = \nabla_\eta\Phi_i(\boldsymbol{\theta}_{\neg i}) - \boldsymbol{\mu}\cdot\nabla_\eta\boldsymbol{\theta} \qquad (**)$$

but $\Phi_i(\boldsymbol{\theta}_{\neg i}) = \log\int f_i(\mathcal{Z}_i)e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_{\neg i}}$ depends on η in two ways: *directly* through f_i and *indirectly* through the converged $\boldsymbol{\theta}_{\neg i}$.

$$\begin{aligned}\nabla_\eta\Phi_i(\boldsymbol{\theta}_{\neg i}) &= \partial_{\boldsymbol{\theta}_{\neg i}}\Phi_i(\boldsymbol{\theta}_{\neg i})\cdot\nabla_\eta\boldsymbol{\theta}_{\neg i} + e^{-\Phi_i(\boldsymbol{\theta}_{\neg i})}\int\nabla_\eta f_i(\mathcal{Z}_i)\ e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_{\neg i}}\,d\mathcal{Z}\\ &= \langle T(\mathcal{Z})\rangle_{\widehat{P}_i}\cdot\nabla_\eta\boldsymbol{\theta}_{\neg i} + \int\nabla_\eta\log f_i(\mathcal{Z}_i)\ f_i(\mathcal{Z}_i)e^{\mathbf{T}(\mathcal{Z})\cdot\boldsymbol{\theta}_{\neg i}-\Phi_i(\boldsymbol{\theta}_{\neg i})}\,d\mathcal{Z}\\ &= \boldsymbol{\mu}\cdot\nabla_\eta\boldsymbol{\theta}_{\neg i} + \langle\nabla_\eta\log f_i(\mathcal{Z}_i)\rangle_{\widehat{P}_i} \qquad (***)\end{aligned}$$

by EP moment matching at convergence!

EP log-likelihood optimisation for learning

So putting it all together:

$$\begin{aligned}
\nabla_\eta \tilde{\ell} &= \boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta} + \sum_{i=1}^{N} \nabla_\eta \log \tilde{C}_i && (*)\\
&= \boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta} + \sum_{i=1}^{N} \Big(\nabla_\eta \Phi_i(\boldsymbol{\theta}_{\neg i}) - \boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta} \Big) && (**)\\
&= \boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta} + \sum_{i=1}^{N} \Big(\boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta}_{\neg i} - \boldsymbol{\mu} \cdot \nabla_\eta \boldsymbol{\theta} + \langle \nabla_\eta \log f_i(\mathcal{Z}_i) \rangle_{\widehat{P}_i} \Big) && (***)\\
&= \boldsymbol{\mu} \cdot \nabla_\eta \Big(\boldsymbol{\theta} + \sum_{i=1}^{N} (\boldsymbol{\theta}_{\neg i} - \boldsymbol{\theta}) \Big) + \sum_{i=1}^{N} \langle \nabla_\eta \log f_i(\mathcal{Z}_i) \rangle_{\widehat{P}_i}\\
&= \boldsymbol{\mu} \cdot \nabla_\eta \Big(\sum_{i=1}^{N} \boldsymbol{\theta}_i + \sum_{i=1}^{N} (\boldsymbol{\theta}_{\neg i} - \boldsymbol{\theta}) \Big) + \sum_{i=1}^{N} \langle \nabla_\eta \log f_i(\mathcal{Z}_i) \rangle_{\widehat{P}_i}\\
&= \boldsymbol{\mu} \cdot \nabla_\eta \sum_{i=1}^{N} (\boldsymbol{\theta} - \boldsymbol{\theta}) + \sum_{i=1}^{N} \langle \nabla_\eta \log f_i(\mathcal{Z}_i) \rangle_{\widehat{P}_i}\\
&= \sum_{i=1}^{N} \langle \nabla_\eta \log f_i(\mathcal{Z}_i) \rangle_{\widehat{P}_i}
\end{aligned}$$

and the gradient can be computed provided EP converges.

A final generalisation: alpha divergences and Power EP

- Alpha divergences

$$D_\alpha[p\|q] = \frac{1}{\alpha(1-\alpha)} \int dx \, \big(\alpha p(x) + (1-\alpha) q(x) - p(x)^\alpha q(x)^{1-\alpha}\big)$$

$$D_{-1}[p\|q] = \frac{1}{2} \int dx \, \frac{(p(x) - q(x))^2}{p(x)}$$

$$\lim_{\alpha \to 0} D_\alpha[p\|q] = \mathbf{KL}[q\|p] \qquad \text{Note: } \lim_{\alpha \to 0} \frac{(p(x)/q(x))^\alpha}{\alpha} = \log \frac{p(x)}{q(x)}$$

$$D_{\frac{1}{2}}[p\|q] = 2 \int dx \, \big(p(x)^{\frac{1}{2}} - q(x)^{\frac{1}{2}}\big)^2$$

$$\lim_{\alpha \to 1} D_\alpha[p\|q] = \mathbf{KL}[p\|q]$$

$$D_2[p\|q] = \frac{1}{2} \int dx \, \frac{(p(x) - q(x))^2}{q(x)}$$

- Local (EP) minimisation gives fixed-point updates that blend messages (to power α) with previous site approximations.

$$\tilde{f}_i^{\text{new}} = \operatorname*{argmin}_{f \in \{\tilde{f}\}} \mathbf{KL}\big[f_i(\mathcal{Z}_i)^\alpha \tilde{f}_i(\mathcal{Z}_i)^{1-\alpha} q_{\neg i}(\mathcal{Z}) \big\| f(\mathcal{Z}_i) q_{\neg i}(\mathcal{Z}) \big]$$

- Small changes (for $\alpha < 1$) lead to more stable updates, and more reliable convergence.

Probabilistic & Unsupervised Learning
Approximate Inference

Belief Propagation

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

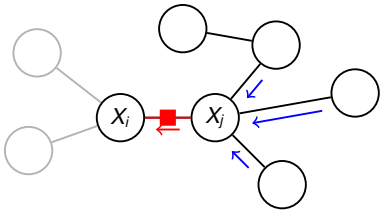
Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2022

Recall: Belief Propagation on undirected trees

Joint distribution of undirected tree:

$$p(\mathcal{X}) = \frac{1}{Z} \prod_{\text{nodes } i} f_i(X_i) \prod_{\text{edges } (ij)} f_{ij}(X_i, X_j)$$



Messages computed recursively:

$$M_{j\to i}(X_i) := \sum_{X_j} f_{ij}(X_i, X_j) f_j(X_j) \prod_{l \in \text{ne}(j)\setminus i} M_{l\to j}(X_j)$$

Marginal distributions:

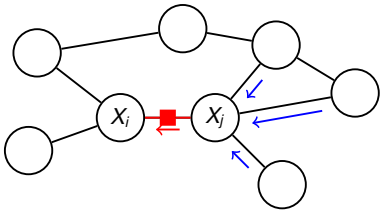
$$p(X_i) \propto f_i(X_i) \prod_{k\in\text{ne}(i)} M_{k\to i}(X_i)$$

$$p(X_i, X_j) \propto f_{ij}(X_i, X_j) f_i(X_i) f_j(X_j) \prod_{k\in\text{ne}(i)\setminus j} M_{k\to i}(X_i) \prod_{l\in\text{ne}(j)\setminus i} M_{l\to j}(X_j)$$

Loopy Belief Propagation

Joint distribution of undirected graph:

$$p(\mathcal{X}) = \frac{1}{Z} \prod_{\text{nodes } i} f_i(X_i) \prod_{\text{edges } (ij)} f_{ij}(X_i, X_j)$$



Messages computed recursively (with few guarantees of convergence):

$$M_{j\to i}(X_i) := \sum_{X_j} f_{ij}(X_i, X_j) f_j(X_j) \prod_{l \in \text{ne}(j)\setminus i} M_{l\to j}(X_j)$$

Marginal distributions are approximate in general:

$$p(X_i) \approx b_i(X_i) \propto f_i(X_i) \prod_{k\in\text{ne}(i)} M_{k\to i}(X_i)$$

$$p(X_i, X_j) \approx b_{ij}(X_i, X_j) \propto f_{ij}(X_i, X_j) f_i(X_i) f_j(X_j) \prod_{k\in\text{ne}(i)\setminus j} M_{k\to i}(X_i) \prod_{l\in\text{ne}(j)\setminus i} M_{l\to j}(X_j)$$

Dealing with loops

- **Accuracy**: BP posterior marginals are approximate on all non-trees because evidence is over counted, but converged approximations are frequently found to be good (particularly in their means).
- **Convergence**: no general guarantee, but BP does converge in some cases:
  - Trees.
  - Graphs with a single loop.
  - Distributions with sufficiently weak interactions.
  - Graphs with long (and weak) loops
  - Gaussian networks: means correct, variances may also converge.
- **Damping**: Common approach to encourage convergence (cf EP)

$$M^{\text{new}}_{i\to j}(X_j) := (1-\alpha) M^{\text{old}}_{i\to j}(X_j) + \alpha \sum_{X_i} f_{ij}(X_i, X_j) f_i(X_i) \prod_{k\in\text{ne}(i)\setminus j} M_{k\to i}(X_i)$$

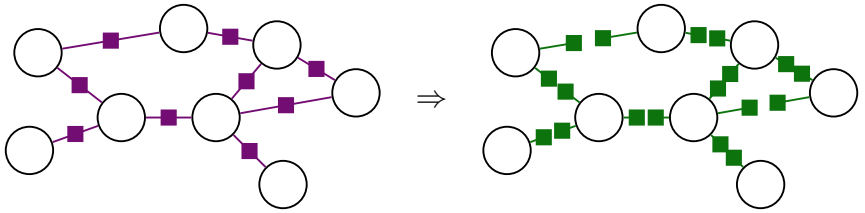
- **Grouping variables**: Variables can be grouped into cliques to improve accuracy.
  - Region graph approximations.
  - Cluster variational method.
  - Junction graph.

Different Interpretations of Loopy Belief Propagation

Loopy BP can be interpreted as a fixed point algorithm from a few different perspectives:

- Expectation propagation.
- Tree-based reparametrization.
- Bethe free energy.

Loopy BP as message-based Expectation Propagation



Approximate pairwise factors f_{ij} by product of messages:

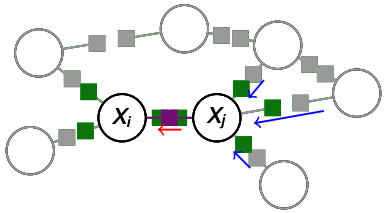
$$f_{ij}(X_i, X_j) \approx \tilde{f}_{ij}(X_i, X_j) = M_{i\to j}(X_j) M_{j\to i}(X_i)$$

Thus, the full joint is approximated by a factorised distribution:

$$p(\mathcal{X}) \approx \frac{1}{Z} \prod_{\text{nodes } i} f_i(X_i) \prod_{\text{edges } (ij)} \tilde{f}_{ij}(X_i, X_j) = \frac{1}{Z} \prod_{\text{nodes } i} \left(f_i(X_i) \prod_{j \in \text{ne}(i)} M_{j\to i}(X_i) \right) = \prod_{\text{nodes } i} b_i(X_i)$$

but with multiple factors for most X_i.

Loopy BP as message-based EP



Then the EP updates to the messages are:

- Deletion:

$$q_{\neg ij}(X_i, X_j) = f_i(X_i) f_j(X_j) \prod_{k \in \text{ne}(i)\setminus j} M_{k\to i}(X_i) \prod_{l \in \text{ne}(j)\setminus i} M_{l\to j}(X_j) \prod_{s \neq i,j} f_s(X_s) \prod_{t \in \text{ne}(s)} M_{t\to s}(X_s)$$

- Projection:

$$\{M^{\text{new}}_{i\to j}, M^{\text{new}}_{j\to i}\} = \text{argmin}\, \mathbf{KL}[f_{ij}(X_i, X_j) q_{\neg ij}(X_i, X_j) \| M_{j\to i}(X_i) M_{i\to j}(X_j) q_{\neg ij}(X_i, X_j)]$$

Now, $q_{\neg ij}()$ factors $\Rightarrow$ rhs factors $\Rightarrow$ min is achieved by marginals of $f_{ij}() q_{\neg ij}()$

$$M^{\text{new}}_{j\to i}(X_i) q_{\neg ij}(X_i) = \sum_{X_j} \left(f_{ij}(X_i, X_j) f_j(X_j) \prod_{l \in \text{ne}(j)\setminus i} M_{l\to j}(X_j) \right) \underbrace{f_i(X_i) \prod_{k \in \text{ne}(i)\setminus j} M_{k\to i}(X_i)}_{q_{\neg ij}(X_i)}$$

$$\Rightarrow M^{\text{new}}_{j\to i}(X_i) = \sum_{X_j} \left(f_{ij}(X_i, X_j) f_j(X_j) \prod_{l \in \text{ne}(j)\setminus i} M_{l\to j}(X_j) \right)$$

Message-based EP

- Thus message-based EP in a loopy graph need not be seen as two separate approximations (one to the sites and one to the cavity) as we had in the EP lecture.
- Instead, we can see it as a more severe constraint on the approximate sites: not just to an ExpFam factor, but to a product of ExpFam messages.
- On a tree-structured graph the message-factored version of EP finds the same marginals as standard EP.
  - Messages are calculated in exactly the same way as before (cf NLSSM).
  - Pairwise marginals can be found after convergence by computing $\tilde{P}(\mathbf{z}_{i-1}, \mathbf{z}_i)$ as required (cf Forward-backward for HMMs).
  - Would not be true of fully-factored variational approximation.
- Factorisation view remains valid even when original sites lie in the appropriate ExpFam already – so loopy BP in (eg) discrete graphs can be seen as a form of EP.
- However, this view does not help us understand the convergence properties of BP.

Loopy BP as tree-based reparametrisation

Tree-structured distributions can be parametrised in many ways:

$$p(\mathcal{X}) = \frac{1}{Z}\prod_{\text{nodes } i} f_i(X_i)\prod_{\text{edges}(ij)} f_{ij}(X_i, X_j) \qquad \text{undirected tree} \qquad (1)$$

$$= p(X_r)\prod_{i\neq r} p(X_i|X_{\text{pa}(i)}) \qquad \text{directed (rooted) tree} \qquad (2)$$

$$= \prod_{\text{nodes } i} p(X_i)\prod_{\text{edges } (ij)} \frac{p(X_i, X_j)}{p(X_i)p(X_j)} \qquad \text{pairwise marginals} \qquad (3)$$

where (3) requires that $\sum_{X_j} p(X_i, X_j) = p(X_i)$.

The undirected tree representation is not unique—multiplying a factor $f_{ij}(X_i, X_j)$ by $g(X_i)$ and dividing $f_i(X_i)$ by the same $g(X_i)$ does not change the distribution.

BP can be seen as an iterative replacement of $f_i(X_i)$ by the local marginal of $p_{ij}(X_i, X_j)$, along with the corresponding reparametrisation of $f_{ij}(X_i, X_j)$. Cf. Hugin propagation.

Converged BP on a tree finds $p(X_i)$ and $p(X_i, X_j)$, allowing us to transform (1) to (3).

Reparametrisation on non-trees

- If BP converges on a non-tree, it will have successfully reparametrised the distribution to have locally consistent beliefs:

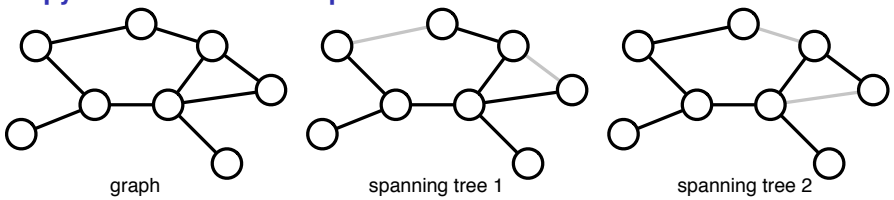
$$p(\mathcal{X}) \propto \prod_i b(X_i)\prod_{(ij)} \frac{b(X_i, X_j)}{b(X_i)b(X_j)} \quad \text{with} \quad \sum_{X_j} b(X_i, X_j) = b(X_i) \text{ etc.}$$

- However, the marginals will not usually be correct or globally consistent. That is

$$\sum_{\mathcal{X}_{\neg i}} \left(\prod_i b(X_i)\prod_{(ij)} \frac{b(X_i, X_j)}{b(X_i)b(X_j)}\right) \neq b(X_i)$$

  and the product will not generally be normalised.
- What can be said about these pseudomarginals?
- Consider the following (theoretical) message scheduling scheme:
  - Identify all the spanning trees of the graph.
  - Pass messages along edges of each spanning tree in turn.
  - Iterate over spanning trees to convergence

Loopy BP as tree-based reparametrisation



graph spanning tree 1 spanning tree 2

$$p(\mathcal{X}) = \frac{1}{Z}\prod_{\text{nodes } i} f_i^0(X_i)\prod_{\text{edges } (ij)} f_{ij}^0(X_i, X_j)$$

$$= \frac{1}{Z}\prod_{\text{nodes } i\in T_1} f_i^0(X_i)\prod_{\text{edges } (ij)\in T_1} f_{ij}^0(X_i, X_j)\prod_{\text{edges } (ij)\notin T_1} f_{ij}^0(X_i, X_j)$$

$$= \frac{1}{Z}\prod_{\text{nodes } i\in T_1} f_i^1(X_i)\prod_{\text{edges } (ij)\in T_1} f_{ij}^1(X_i, X_j)\prod_{\text{edges } (ij)\notin T_1} f_{ij}^1(X_i, X_j)$$

where $f_i^1(X_i) = p^{T_1}(X_i)$, $f_{ij}^1(X_i, X_j) = \frac{p^{T_1}(X_i, X_j)}{p^{T_1}(X_i)p^{T_1}(X_j)}$, $f_{ij}^1 = f_{ij}^0$.

$$= \frac{1}{Z}\prod_{\text{nodes } i\in T_2} f_i^1(X_i)\prod_{\text{edges } (ij)\in T_2} f_{ij}^1(X_i, X_j)\prod_{\text{edges } (ij)\notin T_2} f_{ij}^1(X_i, X_j)$$

...

Loopy BP as tree-based reparametrisation

At convergence, loopy BP has reparametrised the joint distribution as:

$$p(\mathcal{X}) = \frac{1}{Z}\prod_{\text{nodes } i} f_i^\infty(X_i)\prod_{\text{edges } (ij)} f_{ij}^\infty(X_i, X_j)$$

where for any tree T embedded in the graph,

$$f_i^\infty(X_i) = p^T(X_i)$$

$$f_{ij}^\infty(X_i, X_j) = \frac{p^T(X_i, X_j)}{p^T(X_i)p^T(X_j)}$$

Thus, the local marginals of all subtrees are locally consistent with each other, and the pseudomarginals represent valid beliefs for any of the subtrees.

$$p(\mathcal{X}) = \frac{1}{Z}\prod_{\text{nodes } i} b_i(X_i)\prod_{\text{edges } (ij)} \frac{b_{ij}(X_i, X_j)}{b_i(X_i)b_j(X_j)}$$

Loopy BP and Bethe free energy

In the reparametrisation view, BP solves for marginal beliefs $b_{ij}(X_i, X_j)$ and $b_i(X_i) = \sum_{X_j} b_{ij}(X_i, X_j)$ such that

$$p(\mathcal{X}) \propto \prod_i f_i(X_i) \prod_{(ij)} f_{ij}(X_i, X_j) \propto \prod_i b_i(X_i) \prod_{(ij)} \frac{b_{ij}(X_i, X_j)}{b_i(X_i) b_j(X_j)}$$

Another view of loopy BP is as a set of fixed point equations for finding stationary points of an objective function called the Bethe free energy, which is defined in terms of the locally consistent beliefs (or pseudomarginals) $b_i \geq 0$ and $b_{ij} \geq 0$:

$$\sum_{x_i} b_i(x_i) = 1 \qquad \forall i$$

$$\sum_{x_j} b_{ij}(x_i, x_j) = b_i(x_i) \qquad \forall i,\ j \in \mathrm{ne}(i), x_i$$

Loopy BP and Bethe free energy

Recall that the variational free energy is: $\mathcal{F}(q) = \langle \log P(\mathcal{X}) \rangle_q + \mathbf{H}[q]$

We define the (negative) Bethe free energy: $\mathcal{F}_{\text{bethe}}(b) = \mathcal{E}_{\text{bethe}}(b) + \mathcal{H}_{\text{bethe}}(b)$ where both terms are approximations to the corresponding variational likelihood terms.

- The Bethe average energy is the expected log-joint evaluated as though the pseudomarginals were correct:

$$\mathcal{E}_{\text{bethe}}(b) = \sum_i \sum_{x_i} b_i(x_i) \log f_i(x_i) + \sum_{(ij)} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log f_{ij}(x_i, x_j)$$

- The Bethe entropy is the sum of the pseudomarginal entropies corrected for pairwise (pseudo)interactions, but neglecting higher-order dependence:

$$\begin{aligned}\mathcal{H}_{\text{bethe}}(b) &= \sum_i \mathbf{H}[b_i] - \sum_{(ij)} \mathbf{KL}[b_{ij} \| b_i b_j] \\ &= -\sum_i \sum_{x_i} b_i(x_i) \log b_i(x_i) - \sum_{(ij)} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)}\end{aligned}$$

- On a tree, both the beliefs and the Bethe entropy expression are correct, so $\mathcal{F}_{\text{bethe}} = \mathcal{F}$.
- Message updates in loopy BP can now be derived by finding the stationary points of a Lagrangian with local consistency and normalisation constraints. The BP messages are related to the Lagrange multipliers.

Bethe fixed point equations

The Bethe free-energy Lagrangian is:

$$\begin{aligned}\mathcal{L} = &\sum_i \sum_{x_i} b_i(x_i) \log f_i(x_i) + \sum_{(ij)} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log f_{ij}(x_i, x_j) && [\mathcal{E}_{\text{bethe}}] \\ &- \sum_i \sum_{x_i} b_i(x_i) \log b_i(x_i) - \sum_{(ij)} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} && [\mathcal{H}_{\text{bethe}}] \\ &+ \sum_i \xi_i \Big(\sum_{x_i} b_i(x_i) - 1 \Big) && [\text{norm } \forall i] \\ &+ \sum_{(ij)} \Big[\sum_{x_i} \xi_{ij}(x_i) \Big(\sum_{x_j} b_{ij}(x_i, x_j) - b_i(x_i) \Big) + \sum_{x_j} \xi_{ji}(x_j) \Big(\sum_{x_i} b_{ij}(x_i, x_j) - b_j(x_j) \Big) \Big] && [\text{marg } \forall i, j, x_i]\end{aligned}$$

Setting derivatives wrt beliefs to 0 gives

$$\frac{\partial \mathcal{L}}{\partial b_i(x_i)} = \log f_i(x_i) - \log b_i(x_i) + \sum_{j \in \mathrm{ne}(i)} \underbrace{\sum_{x_j} \frac{b_{ij}(x_i, x_j)}{b_i(x_i)}}_{=1 \text{ by constraint}} + \xi_i - \sum_{j \in \mathrm{ne}(i)} \xi_{ij}(x_i) + \mathit{const} = 0$$

$$\Rightarrow b_i(x_i) \propto f_i(x_i) \prod_{j \in \mathrm{ne}(i)} e^{-\xi_{ij}(x_i)}$$

$$\frac{\partial \mathcal{L}}{\partial b_{ij}(x_i, x_j)} = \log f_{ij}(x_i, x_j) - \log b_{ij}(x_i, x_j) + \log b_i(x_i) b_j(x_j) + \xi_{ij}(x_i) + \xi_{ji}(x_j) + \mathit{const} = 0$$

$$\Rightarrow b_{ij}(x_i, x_j) \propto f_{ij}(x_i, x_j) b_i(x_i) b_j(x_j) e^{\xi_{ij}(x_i)} e^{\xi_{ji}(x_j)}$$

Bethe fixed point messages

The Bethe Lagrangian fixed point equations are:

$$b_i(x_i) \propto f_i(x_i) \prod_{j \in \mathrm{ne}(i)} e^{-\xi_{ij}(x_i)}$$

$$b_{ij}(x_i, x_j) \propto f_{ij}(x_i, x_j) b_i(x_i) b_j(x_j) e^{\xi_{ij}(x_i)} e^{\xi_{ji}(x_j)}$$

Comparison with BP suggests that messages should have the form $M_{j \to i}(x_i) = e^{-\xi_{ij}(x_i)}$.

Indeed, solving for $\xi_{ij}(x_i)$ by enforcing the constraint $\sum_{x_j} b_{ij}(x_i, x_j) = b_i(x_i)$ we have:

$$\sum_{x_j} b_{ij}(x_i, x_j) \propto \sum_{x_j} f_{ij}(x_i, x_j) b_i(x_i) b_j(x_j) e^{\xi_{ij}(x_i)} e^{\xi_{ji}(x_j)}$$

$$\Rightarrow b_i(x_i) \propto b_i(x_i) e^{\xi_{ij}(x_i)} \sum_{x_j} f_{ij}(x_i, x_j) b_j(x_j) e^{\xi_{ji}(x_j)}$$

$$\Rightarrow e^{-\xi_{ij}(x_i)} \propto \sum_{x_j} f_{ij}(x_i, x_j) b_j(x_j) e^{\xi_{ji}(x_j)}$$

$$= \sum_{x_j} f_{ij}(x_i, x_j) f_j(x_j) \prod_{l \in \mathrm{ne}(j) \setminus i} e^{-\xi_{jl}(x_j)}$$

thus recovering the BP message passing rules.

Loopy BP and Bethe free energy

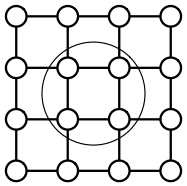
- Fixed points of loopy BP are exactly the stationary points of the Bethe free energy.
- Stable fixed points of loopy BP are local maxima of Bethe free enegy (note the negative definition of free energy for consistency with the variational free energy).
- For binary attractive networks, Bethe free energy at fixed points of loopy BP provides an upper bound on the log partition function $\log Z$—this is useful for learning undirected graphical models as it leads to a lower bound on the log likelihood.

Loopy BP vs mean-field approximation

- Beliefs b_i and b_{ij} in loopy BP are only locally consistent pseudomarginals, not necessarily consistent marginals of the implied joint distribution.
- Bethe free energy accounts for interactions between different sites, while variational free energy assumes independence.
- The loop series or Plefka expansion of the log partition function Z: the variational free energy forms the first order terms, while Bethe free energy contains higher order terms (involving generalized loops).
- Loopy BP tends to be signficantly more accurate whenever it converges.

Extensions and variations

- Generalized BP: group variables together to treat their interactions exactly.



- Convergent alternatives: Fixed points of loopy BP are stationary points of the Bethe free enegy. We can also derive algorithms that **increase** the Bethe free energy at every step, and are thus are guaranteed to converge.
- Convex alternatives: We can derive convex cousins of the negative of the Bethe free energy. These give rise to algorithms that will converge to a unique global maximum.
- We have considered sum-product loopy BP to compute marginals. The treatment of loopy Viterbi or max-product algorithms is different.

References

- Probabilistic Reasoning in Intelligent Systems. J. Pearl. Morgan Kaufman, 1988.
- Turbo decoding as an instance of Pearl's belief propagation algorithm. R. J. McEliece, D. J. C. MacKay and J. F. Cheng. IEEE Journal on Selected Areas in Communication, 1998, 16(2):140-152.
- Iterative decoding of compound codes by probability propagation in graphical models. F. Kschischang and B. Frey. IEEE Journal on Selected Areas in Communication, 1998, 16(2):219-230.
- A family of algorithms for approximate Bayesian inference. T. Minka. PhD Thesis, 2001.
- Tree-based reparameterization framework for analysis of sum-product and related algorithms. M. J. Wainwright, T. S. Jaakkola and A. S. Willsky. IEEE Transactions on Information Theory, 2004, 49(5).
- Constructing free energy approximations and generalized belief propagation algorithms. J. S. Yedidia, W. T. Freeman and Y. Weiss. IEEE Transactions on Information Theory, 2005, 51:2282-2313.

Probabilistic & Unsupervised Learning
Approximate Inference

Exponential families: convexity, duality and free energies

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2021

Exponential families: the log partition function

Consider an exponential family distribution with sufficient statistic $s(X)$ and natural parameter $\boldsymbol{\theta}$ (and no base factor in X alone). We can write its probability or density function as

$$p(X|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^\mathsf{T} s(X) - \Phi(\boldsymbol{\theta})\right)$$

where $\Phi(\boldsymbol{\theta})$ is the log partition function

$$\Phi(\boldsymbol{\theta}) = \log \sum_x \exp\left(\boldsymbol{\theta}^\mathsf{T} s(x)\right)$$

$\Phi(\boldsymbol{\theta})$ plays an important role in the theory of the exponential family. For example, it maps natural parameters to the moments of the sufficient statistics:

$$\frac{\partial}{\partial \boldsymbol{\theta}}\Phi(\boldsymbol{\theta}) = e^{-\Phi(\boldsymbol{\theta})}\sum_x s(x) e^{\boldsymbol{\theta}^\mathsf{T} s(x)} = \mathbb{E}_{\boldsymbol{\theta}}\left[s(X)\right] = \boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\mu}$$

$$\frac{\partial^2}{\partial \boldsymbol{\theta}^2}\Phi(\boldsymbol{\theta}) = e^{-\Phi(\boldsymbol{\theta})}\sum_x s(x)^2 e^{\boldsymbol{\theta}^\mathsf{T} s(x)} - e^{-2\Phi(\boldsymbol{\theta})}\Big[\sum_x s(x) e^{\boldsymbol{\theta}^\mathsf{T} s(x)}\Big]^2 = \mathbb{V}_{\boldsymbol{\theta}}\left[s(X)\right]$$

The second derivative is thus positive semi-definite, and so $\Phi(\boldsymbol{\theta})$ is convex in $\boldsymbol{\theta}$.

Exponential families: mean parameters and negative entropy

A (minimal) exponential family distribution can also be parameterised by the means of the sufficient statistics.

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}\left[s(X)\right]$$

Consider the negative entropy of the distribution as a function of the mean parameter:

$$\Psi(\boldsymbol{\mu}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\log p(X|\boldsymbol{\theta}(\boldsymbol{\mu}))\right] = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta})$$

so

$$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} = \Phi(\boldsymbol{\theta}) + \Psi(\boldsymbol{\mu})$$

The negative entropy is dual to the log-partition function. For example,

$$\begin{aligned}\frac{d}{d\boldsymbol{\mu}}\Psi(\boldsymbol{\mu}) &= \frac{\partial}{\partial \boldsymbol{\mu}}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta})) + \frac{d\boldsymbol{\theta}}{d\boldsymbol{\mu}}\frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta})) \\ &= \boldsymbol{\theta} + \frac{d\boldsymbol{\theta}}{d\boldsymbol{\mu}}(\boldsymbol{\mu} - \boldsymbol{\mu}) = \boldsymbol{\theta}\end{aligned}$$

Exponential families: duality

The log partition function and negative entropy are Legendre dual or convex conjugate functions.

Consider the KL divergence between distributions with natural parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$:

$$\begin{aligned}\mathbf{KL}\left[\boldsymbol{\theta}\middle\|\boldsymbol{\theta}'\right] &= \mathbf{KL}\left[p(X|\boldsymbol{\theta})\middle\|p(X|\boldsymbol{\theta}')\right] = \mathbb{E}_{\boldsymbol{\theta}}\left[-\log p(X|\boldsymbol{\theta}') + \log p(X|\boldsymbol{\theta})\right] \\ &= -\boldsymbol{\theta}'^\mathsf{T}\boldsymbol{\mu} + \Phi(\boldsymbol{\theta}') + \Psi(\boldsymbol{\mu}) \geq 0 \\ \Rightarrow \Psi(\boldsymbol{\mu}) &\geq \boldsymbol{\theta}'^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta}')\end{aligned}$$

where $\boldsymbol{\mu}$ are the mean parameters corresponding to $\boldsymbol{\theta}$.

Now, the minimum KL divergence of zero is reached iff $\boldsymbol{\theta} = \boldsymbol{\theta}'$, so

$$\Psi(\boldsymbol{\mu}) = \sup_{\boldsymbol{\theta}'}\left[\boldsymbol{\theta}'^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta}')\right] \qquad \text{and, if finite} \qquad \boldsymbol{\theta}(\boldsymbol{\mu}) = \operatorname*{argmax}_{\boldsymbol{\theta}'}\left[\boldsymbol{\theta}'^\mathsf{T}\boldsymbol{\mu} - \Phi(\boldsymbol{\theta}')\right]$$

The left-hand equation is the definition of the conjugate dual of a convex function.

Continuous functions are reciprocally dual, so we also have:

$$\Phi(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}'}\left[\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu}' - \Psi(\boldsymbol{\mu}')\right] \qquad \text{and, if finite} \qquad \boldsymbol{\mu}(\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\mu}'}\left[\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\mu}' - \Psi(\boldsymbol{\mu}')\right]$$

Thus, duality gives us another relation between $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$.

Duality, inference and the free energy

Consider a joint exponential family distribution on observed $\mathbf{x}$ and latent $\mathbf{z}$.

$$p(\mathbf{x}, \mathbf{z}) = \exp\left[\boldsymbol{\theta}^\mathsf{T} s(\mathbf{x}, \mathbf{z}) - \Phi_{XZ}(\boldsymbol{\theta})\right]$$

The posterior on $\mathbf{z}$ is also in the exponential family, with the clamped sufficient statistic $s_Z(\mathbf{z}; \mathbf{x}) = s_{XZ}(\mathbf{x}^{\text{obs}}, \mathbf{z})$; the same (now possibly redundant) natural parameter $\boldsymbol{\theta}$; and partition function $\Phi_Z(\boldsymbol{\theta}) = \log \sum_{\mathbf{z}} \exp \boldsymbol{\theta}^\mathsf{T} s_Z(\mathbf{z})$.

The likelihood is

$$\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{z}} e^{\boldsymbol{\theta}^\mathsf{T} s(\mathbf{x},\mathbf{z}) - \Phi_{XZ}(\boldsymbol{\theta})} = \sum_{\mathbf{z}} e^{\boldsymbol{\theta}^\mathsf{T} s_Z(\mathbf{z};\mathbf{x})} e^{-\Phi_{XZ}(\boldsymbol{\theta})} = \exp[\Phi_Z(\boldsymbol{\theta}) - \Phi_{XZ}(\boldsymbol{\theta})]$$

So we can write the log-likelihood as

$$\ell(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}_Z} [\underbrace{\boldsymbol{\theta}^\mathsf{T} \boldsymbol{\mu}_Z - \Phi_{XZ}(\boldsymbol{\theta})}_{\langle \log p(\mathbf{x},\mathbf{z}) \rangle_q} - \underbrace{\Psi(\boldsymbol{\mu}_Z)}_{-\mathbf{H}[q]}] = \sup_{\boldsymbol{\mu}_Z} \mathcal{F}(\boldsymbol{\theta}, \boldsymbol{\mu}_Z)$$

This is the familiar free energy with $q(\mathbf{z})$ represented by its mean parameters $\boldsymbol{\mu}_Z$!

Inference with mean parameters

We have described inference in terms of the distribution q, approximating as needed, then computing expected suff stats. Can we describe it instead as an optimisation over $\boldsymbol{\mu}$ directly?

$$\boldsymbol{\mu}_Z^* = \operatorname*{argmax}_{\boldsymbol{\mu}_Z} [\boldsymbol{\theta}^\mathsf{T} \boldsymbol{\mu}_Z - \Psi(\boldsymbol{\mu}_Z)]$$

Concave maximisation(!), but two complications:

- The optimum must be found over feasible means. Interdependance of the sufficient statistics may prevent arbitrary sets of mean sufficient statistics being achieved
  - Feasible means are convex combinations of all the single-configuration sufficient statistics.

    $$\boldsymbol{\mu} = \sum_{\mathbf{x}} \nu(\mathbf{x}) s(\mathbf{x}) \qquad \sum_{\mathbf{x}} \nu(\mathbf{x}) = 1$$

    - Take a Boltzmann machine on two variables, x_1, x_2.
    - The sufficient stats are $s(\mathbf{x}) = [x_1, x_2, x_1 x_2]$.
    - Clearly only the stats $\mathcal{S} = \{[0,0,0], [0,1,0], [1,0,0], [1,1,1]\}$ are possible.
    - Thus $\boldsymbol{\mu} \in$ convex hull$(\mathcal{S})$.
  - For a discrete distribution, this space of possible means is bounded by exponentially many hyperplanes connecting the discrete configuration stats: called the marginal polytope.
- Even when restricted to the marginal polytope, evaluating $\Psi(\boldsymbol{\mu})$ can be challenging.

Convexity and undirected trees

- We can parametrise a discrete pairwise MRF as follows:

  $$p(\mathbf{X}) = \frac{1}{Z} \prod_i f_i(X) \prod_{(ij)} f_{ij}(X_i, X_j)$$

  $$= \exp\left(\sum_i \sum_k \boldsymbol{\theta}_i(k) \delta(X_i = k) + \sum_{(ij)} \sum_{k,l} \boldsymbol{\theta}_{ij}(k,l) \delta(X_i = k) \delta(X_j = l) - \Phi(\boldsymbol{\theta}) \right)$$

- So discrete MRFs are always exponential family, with natural and mean parameters:

  $$\boldsymbol{\theta} = \left[\boldsymbol{\theta}_i(k), \boldsymbol{\theta}_{ij}(k,l) \quad \forall i, j, k, l\right]$$
  $$\boldsymbol{\mu} = \left[p(X_i = k), p(X_i = k, X_j = l) \quad \forall i, j, k, l\right]$$

  In particular, the mean parameters are just the singleton and pairwise probability tables.
- If the MRF has tree structure T, the negative entropy can be written in terms of the single-site entropies and mutual informations on edges:

  $$\Psi(\boldsymbol{\mu}_T) = \mathbb{E}_{\boldsymbol{\theta}_T} \left[\log \prod_i p(X_i) \prod_{(ij) \in T} \frac{p(X_i, X_j)}{p(X_i) p(X_j)} \right]$$
  $$= -\sum_i H(X_i) + \sum_{(ij) \in T} I(X_i, X_j)$$

The Bethe free energy again

We can see the Bethe free energy problem as a relaxation of the true free-energy optimisation:

$$\boldsymbol{\mu}_Z^* = \operatorname*{argmax}_{\boldsymbol{\mu}_Z \in \mathcal{M}} [\boldsymbol{\theta}^\mathsf{T} \boldsymbol{\mu}_Z - \Psi(\boldsymbol{\mu}_Z)]$$

where $\mathcal{M}$ is the set of feasible means.

1. Relax $\mathcal{M} \to \mathcal{L}$, where $\mathcal{L}$ is the set of locally consistent means (i.e. all nested means marginalise correctly).
2. Approximate $\Psi(\boldsymbol{\mu}_Z)$ by the tree-structured form

   $$\Psi_{\text{Bethe}}(\boldsymbol{\mu}_Z) = -\sum_i H(X_i) + \sum_{(ij) \in G} I(X_i, X_j)$$

$\mathcal{L}$ is still a convex set (polytope for discrete problems). However Ψ_{Bethe} is not convex.

Convexifying BP

Consider instead an upper bound on $\Phi(\boldsymbol{\theta})$:

Imagine a set of spanning trees T for the MRF, each with its own parameters $\boldsymbol{\theta}_T, \boldsymbol{\mu}_T$. By padding entries corresponding to off-tree edges with zero, we can assume that $\boldsymbol{\theta}_T$ has the same dimensionality as $\boldsymbol{\theta}$.

Suppose also that we have a distribution β over the spanning trees so that $\mathbb{E}_\beta[\boldsymbol{\theta}_T] = \boldsymbol{\theta}$.

Then by the convexity of $\Phi(\boldsymbol{\theta})$,

$$\Phi(\boldsymbol{\theta}) = \Phi(\mathbb{E}_\beta[\boldsymbol{\theta}_T]) \leq \mathbb{E}_\beta[\Phi(\boldsymbol{\theta}_T)]$$

If we were to tighten the upper bound we might obtain a good approximation to Φ:

$$\Phi(\boldsymbol{\theta}) \leq \inf_{\beta, \boldsymbol{\theta}_T : \mathbb{E}_\beta[\boldsymbol{\theta}_T] = \boldsymbol{\theta}} \mathbb{E}_\beta[\Phi(\boldsymbol{\theta}_T)]$$

Convex Upper Bounds on the Log Partition Function

$$\Phi(\boldsymbol{\theta}) \leq \inf_{\boldsymbol{\theta}_T : \mathbb{E}_\beta[\boldsymbol{\theta}_T] = \boldsymbol{\theta}} \mathbb{E}_\beta[\Phi(\boldsymbol{\theta}_T)] \stackrel{\text{def}}{=} \Phi_\beta(\boldsymbol{\theta})$$

Solve the constrained optimisation problem using Lagrange multipliers:

$$\mathcal{L} = \mathbb{E}_\beta[\Phi(\boldsymbol{\theta}_T)] - \boldsymbol{\lambda}^\mathsf{T}(\mathbb{E}_\beta[\boldsymbol{\theta}_T] - \boldsymbol{\theta})$$

Setting the derivatives wrt $\boldsymbol{\theta}_T$ to zero, we get:

$$\frac{\partial}{\partial \boldsymbol{\theta}_T} \sum_T \beta(T)\Phi(\boldsymbol{\theta}_T) - \boldsymbol{\lambda}^\mathsf{T} \frac{\partial}{\partial \boldsymbol{\theta}_T} \sum_T \beta(T)\boldsymbol{\theta}_T = 0$$

$$\beta(T)\boldsymbol{\mu}_T - \beta(T)\Pi_T(\boldsymbol{\lambda}) = 0$$

$$\boldsymbol{\mu}_T = \Pi_T(\boldsymbol{\lambda})$$

where $\Pi_T(\boldsymbol{\lambda})$ selects the Lagrange multipliers corresponding to elements of $\boldsymbol{\theta}$ that are non-zero in the tree T.

Although each tree has its own parameters $\boldsymbol{\theta}_T$, at the optimum they are all constrained: their mean parameters are all consistent with each other (c.f. the tree-reparametrisation view of BP) and with the Lagrange multipliers $\boldsymbol{\lambda}$.

Convex Upper Bounds on the Log Partition Function

$$\begin{aligned}
\Phi_\beta(\boldsymbol{\theta}) &= \sup_{\boldsymbol{\lambda}} \inf_{\boldsymbol{\theta}_T} \mathbb{E}_\beta[\Phi(\boldsymbol{\theta}_T)] - \boldsymbol{\lambda}^\mathsf{T}(\mathbb{E}_\beta[\boldsymbol{\theta}_T] - \boldsymbol{\theta}) \\
&= \sup_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\theta} + \mathbb{E}_\beta\left[\inf_{\boldsymbol{\theta}_T} \Phi(\boldsymbol{\theta}_T) - \boldsymbol{\theta}_T^\mathsf{T}\Pi_T(\boldsymbol{\lambda})\right] \\
&= \sup_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\theta} + \mathbb{E}_\beta[-\Psi(\Pi_T(\boldsymbol{\lambda}))] \\
&= \sup_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\theta} + \mathbb{E}_\beta\left[\sum_i H_{\boldsymbol{\lambda}}(X_i) - \sum_{(ij)\in T} I_{\boldsymbol{\lambda}}(X_i, X_j)\right] \\
&= \sup_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\theta} + \sum_i H_{\boldsymbol{\lambda}}(X_i) - \sum_{(ij)} \beta_{ij} I_{\boldsymbol{\lambda}}(X_i, X_j)
\end{aligned}$$

- This is a **convexified** version of the Bethe free energy.
- Optimisation wrt $\boldsymbol{\lambda}$is approximate inference applied to the tighest bound on $\Phi(\theta)$ for fixed β.
- The bound holds for any β and can be tightened by further minimisation.

EP free energy

A Bethe-like approach also casts EP as a variational energy fixed point method.

Consider finding marginals of a (posterior) distribution defined by clique potentials:

$$P(\mathcal{Z}) \propto f_0(\mathcal{Z}) \prod_i f_i(\mathcal{Z}_i)$$

where all factor have exponential form, f_0 is in a tractable exponential family (possibly uniform) bu the f_i are jointly intractable – i.e. product cannot be marginalised, although individual terms may be (numerically) tractable.

Augment by including tractable ExpFam terms with zero natural parameters

$$P(\mathcal{Z}) \propto e^{\boldsymbol{\theta}_0^\mathsf{T}\mathbf{s}_0(\mathcal{Z})} \prod_i e^{\boldsymbol{\theta}_i^\mathsf{T}\mathbf{s}_i(\mathcal{Z}_i)} e^{\mathbf{0}^\mathsf{T}\tilde{\mathbf{s}}_i(\mathcal{Z}_i)} = e^{\boldsymbol{\theta}_0^\mathsf{T}\mathbf{s}_0(\mathcal{Z}) + \sum_i \left(\boldsymbol{\theta}_i^\mathsf{T}\mathbf{s}_i(\mathcal{Z}_i) + \tilde{\boldsymbol{\theta}}^\mathsf{T}\tilde{\mathbf{s}}(\mathcal{Z}_i)\right)}$$

Now, the variational dual principle tells us that the expected sufficient statistics:

$$\boldsymbol{\mu}_0^* = \langle \mathbf{s}_0 \rangle_P; \quad \boldsymbol{\mu}_i^* = \langle \mathbf{s}_i(\mathcal{Z}_i) \rangle_P; \quad \tilde{\boldsymbol{\mu}}_i^* = \langle \tilde{\mathbf{s}}_i \rangle_P$$

are given by

$$\{\boldsymbol{\mu}_0^*, \boldsymbol{\mu}_i^*, \tilde{\boldsymbol{\mu}}_i^*\} = \operatorname*{argmax}_{\{\boldsymbol{\mu}_0, \boldsymbol{\mu}_i, \tilde{\boldsymbol{\mu}}_i\} \in \mathcal{M}} \left[\boldsymbol{\theta}_0^\mathsf{T}\boldsymbol{\mu}_0 + \sum_i \left(\boldsymbol{\theta}_i^\mathsf{T}\boldsymbol{\mu}_i + \mathbf{0}^\mathsf{T}\tilde{\boldsymbol{\mu}}_i\right) - \Psi(\boldsymbol{\mu}_0, \boldsymbol{\mu}_i, \tilde{\boldsymbol{\mu}}_i)\right]$$

EP relaxation

The EP algorithm relaxes this optimisation:

- Relax $\mathcal{M}$ to locally consistent marginals, retaining consistency across each edge connecting $\{\boldsymbol{\mu}_0, \tilde{\boldsymbol{\mu}}_i\}$ (as in BP on a junction graph); and between pairs $(\boldsymbol{\mu}_i, \tilde{\boldsymbol{\mu}}_i)$.
- Replace negative entropy by $\Psi_{\text{Bethe}}(\{\boldsymbol{\mu}_0, \tilde{\boldsymbol{\mu}}_i\}) - \sum_i (\mathbf{H}[\boldsymbol{\mu}_i, \tilde{\boldsymbol{\mu}}_i] - \mathbf{H}[\tilde{\boldsymbol{\mu}}_i])$.
- In effect, drop links between different $\boldsymbol{\mu}_i$ and run reparameterisation on a junction graph.

The free-energy-based approximate marginals include $\boldsymbol{\mu}_i$ which are refined during updates.

- Direct learning on the EP free-energy uses these marginals rather than the approximate ones and a local normaliser formed by integrating over $f_i(\mathcal{Z}_i) q_{\neg i}(\mathcal{Z}_i)$.
- A different derivation to the result from that in EP lecture.

References

- **Graphical Models, Exponential Families, and Variational Inference. Wainwright and Jordan. Foundations and Trends in Machine Learning, 2008 1:1-305.**
- Exact Maximum A Posteriori Estimation for Binary Images. Greig, Porteous and Seheult, Journal of the Royal Statistical Society B, 51(2):271-279, 1989.
- Fast Approximate Energy Minimization via Graph Cuts. Boykov, Veksler and Zabih, International Conference on Computer Vision 1999.
- MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. Wainwright, Jaakkola and Willsky, IEEE Transactions on Information Theory, 2005, 51(11):3697-3717.
- Learning Associative Markov Networks. Taskar, Chatalbashev and Koller, International Conference on Machine Learning, 2004.
- A New Class of Upper Bounds on the Log Partition Function. Wainwright, Jaakkola and Willsky. IEEE Transactions on Information Theory, 2005, 51(7):2313-2335.
- MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. Weiss, Yanover and Meltzer, Uncertainty in Artificial Intelligence, 2007.

Probabilistic & Unsupervised Learning
Approximate Inference

Parametric Variational Methods and Recognition Models

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2023

Variational methods

- Our treatment of variational methods has (except EP) emphasised 'natural' choices of variational family – often factorised using the same functional (ExpFam) form as joint.
  - mostly restricted to joint exponential families – facilitates hierarchical and distributed models, but not non-linear/non-conjugate.

- Consider parametric variational approximations using a constrained family $q(\mathcal{Z};\rho)$.

  The constrained (approximate) variational E-step becomes:

$$q(\mathcal{Z}) := \operatorname*{argmax}_{q\in\{q(\mathcal{Z};\rho)\}} \mathcal{F}\big(q(\mathcal{Z}),\theta^{(k-1)}\big) \quad\Rightarrow\quad \rho^{(k)} := \operatorname*{argmax}_{\rho} \mathcal{F}\big(q(\mathcal{Z};\rho),\theta^{(k-1)}\big)$$

  and so we can replace constrained optimisation of $\mathcal{F}(q,\theta)$ with unconstrained optimisation of a constrained $\mathcal{F}(\rho,\theta)$:

$$\mathcal{F}(\rho,\theta) = \Big\langle \log P(\mathcal{X},\mathcal{Z}|\theta^{(k-1)})\Big\rangle_{q(\mathcal{Z};\rho)} + \mathbf{H}[\rho]$$

  It might still be valuable to use coordinate ascent in ρ and θ, although this is no longer necessary.

Optimising the variational parameters

$$\mathcal{F}(\rho,\theta) = \Big\langle \log P(\mathcal{X},\mathcal{Z}|\theta^{(k-1)})\Big\rangle_{q(\mathcal{Z};\rho)} + \mathbf{H}[\rho]$$

- In some special cases, the expectations of the log-joint under $q(\mathcal{Z};\rho)$ can be expressed in closed form, but these are rare.
- Otherwise we might seek to follow $\nabla_\rho\mathcal{F}$.
- Naively, this requires evaluting a high-dimensional expectation wrt $q(\mathcal{Z},\rho)$ as a function of ρ – not simple.
- At least three solutions:
  - "Score-based" gradient estimate, and Monte-Carlo (Ranganath et al. 2014).
  - Recognition network trained in separate phase – not strictly variational (Dayan et al. 1995).
  - Recognition network trained simultaneously with generative model using "frozen" samples (Kingma and Welling 2014; Rezende et al. 2014).

Score-based gradient estimate

We have:

$$\begin{aligned}\nabla_\rho\mathcal{F}(\rho,\theta) &= \nabla_\rho \int d\mathcal{Z}\, q(\mathcal{Z};\rho)(\log P(\mathcal{X},\mathcal{Z}|\theta) - \log q(\mathcal{Z};\rho))\\ &= \int d\mathcal{Z}\,\Big([\nabla_\rho q(\mathcal{Z};\rho)](\log P(\mathcal{X},\mathcal{Z}|\theta) - \log q(\mathcal{Z};\rho))\\ &\qquad + q(\mathcal{Z};\rho)\nabla_\rho[\cancel{\log P(\mathcal{X},\mathcal{Z}|\theta)} - \cancel{\log q(\mathcal{Z};\rho)}]\Big)\end{aligned}$$

Now,

$$\begin{aligned}\nabla_\rho \log P(\mathcal{X},\mathcal{Z}|\theta) &= 0 && \text{(no direct dependence)}\\ \int d\mathcal{Z}\, q(\mathcal{Z};\rho)\nabla_\rho \log q(\mathcal{Z};\rho) &= \overbrace{\int d\mathcal{Z}\,\nabla_\rho q(\mathcal{Z};\rho)}^{1} = 0 && \text{(always normalised)}\\ \nabla_\rho q(\mathcal{Z};\rho) &= q(\mathcal{Z};\rho)\nabla_\rho \log q(\mathcal{Z};\rho) && \leftarrow \text{"score trick"}\end{aligned}$$

So,

$$\nabla_\rho\mathcal{F}(\rho,\theta) = \Big\langle [\nabla_\rho \log q(\mathcal{Z};\rho)](\log P(\mathcal{X},\mathcal{Z}|\theta) - \log q(\mathcal{Z};\rho))\Big\rangle_{q(\mathcal{Z};\rho)}$$

Reduced gradient of expectation to expectation of gradient – easier to compute. Also called the REINFORCE trick.

Factorisation

$$\nabla_\rho \mathcal{F}(\rho,\theta) = \left\langle [\nabla_\rho \log q(\mathcal{Z};\rho)](\log P(\mathcal{X},\mathcal{Z}|\theta) - \log q(\mathcal{Z};\rho)) \right\rangle_{q(\mathcal{Z};\rho)}$$

- Still requires a high-dimensional expectation, but can now be evaluated by Monte-Carlo.
- Dimensionality reduced by factorisation (particularly where $P(\mathcal{X},\mathcal{Z})$ is factorised).

  Let $q(\mathcal{Z}) = \prod_i q(\mathcal{Z}_i|\rho_i)$ factor over disjoint cliques; let $\bar{\mathcal{Z}}_i$ be the minimal Markov blanket of $\mathcal{Z}_i$ in the joint; $P_{\bar{\mathcal{Z}}_i}$ be the product of joint factors that include any element of $\mathcal{Z}_i$ (so the union of their arguments is $\bar{\mathcal{Z}}_i$); and $P_{\neg\bar{\mathcal{Z}}_i}$ the remaining factors. Then,

$$\begin{aligned}
\nabla_{\rho_i}\mathcal{F}(\{\rho_j\},\theta) &= \left\langle [\nabla_{\rho_i} \textstyle\sum_j \log q(\mathcal{Z}_j;\rho_j)](\log P(\mathcal{X},\mathcal{Z}|\theta) - \textstyle\sum_j \log q(\mathcal{Z}_j;\rho_j)) \right\rangle_{q(\mathcal{Z})} \\
&= \left\langle [\nabla_{\rho_i} \log q(\mathcal{Z}_i;\rho_i)](\log P_{\bar{\mathcal{Z}}_i}(\mathcal{X},\bar{\mathcal{Z}}_i) - \log q(\mathcal{Z}_i;\rho_i) \right\rangle_{q(\bar{\mathcal{Z}}_i)} \\
&\quad + \Big\langle [\nabla_{\rho_i} \log q(\mathcal{Z}_i;\rho_i)] \underbrace{(\log P_{\neg\bar{\mathcal{Z}}_i}(\mathcal{X},\mathcal{Z}_{\neg i}) - \sum_{j\neq i} \log q(\mathcal{Z}_j;\rho_j)}_{\text{constant wrt } \mathcal{Z}_i} \Big\rangle_{q(\mathcal{Z})}
\end{aligned}$$

So the second term is proportional to $\langle \nabla_{\rho_i} \log q(\mathcal{Z}_i;\rho_i)\rangle_{q(\mathcal{Z}_i)}$, this $= 0$ as before.
So expectations are only needed wrt $q(\bar{\mathcal{Z}}_i)$ → variational message passing!

Sampling

So the "black-box" variational approach is as follows:

- Choose a parametric (factored) variational family $q(\mathcal{Z}) = \prod_i q(\mathcal{Z}_i;\rho_i)$.
- Initialise factors.
- Repeat to convergence:
  - **Stochastic VE-step**. For each i:
    - Sample from $q(\bar{\mathcal{Z}}_i)$ and estimate expected gradient $\nabla_{\rho_i}\mathcal{F}$.
    - Update ρ_i along gradient.
  - **Stochastic M-step**. For each i:
    - Sample from each $q(\bar{\mathcal{Z}}_i)$.
    - Update corresponding parameters.

- Stochastic gradient steps may employ a Robbins-Munro step-size sequence to promote convergence.
- Variance of the gradient estimators can also be controlled by clever Monte-Carlo techniques (orginal authors used a "control variate" method that we have not studied).

Recognition Models

We have not generally distinguished between multivariate models and iid data instances, grouping all variables together in $\mathcal{Z}$.

However, even for large models (such as HMMs), we often work with multiple data draws (e.g. multiple strings) and each instance requires a separate variational optimisation.

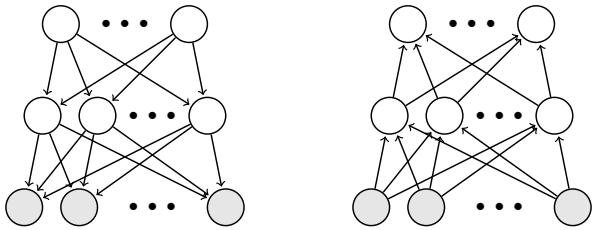
Suppose that we have fixed length vectors $\{(\mathbf{x}_i,\mathbf{z}_i)\}$ ($\mathbf{z}$ is still latent).

- Optimal variational distribution $q^*(\mathbf{z}_i)$ depends on $\mathbf{x}_i$.
- Learn this mapping (in parametric form): $q(\mathbf{z}_i;\rho = f(\mathbf{x}_i;\phi))$.
- Now ρ is the output of a general function approximator f (a GP, neural network or similar) parametrised by ϕ, trained to map $\mathbf{x}_i$ to the variational parameters of $q(\mathbf{z}_i)$.
- The mapping function f is called a recognition model.
- This is approach is now often called amortised inference.

How to learn f?

The Helmholtz Machine

Dayan et al. (1995) originally studied binary sigmoid belief net, with parallel recognition model:



Two phase learning:

- Wake phase: given current f, estimate mean-field representation from data (mean sufficient stats for Bernoulli are just probabilities):

  $$q(\mathbf{z}_i) = \text{Bernoulli}[\hat{\mathbf{z}}_i] \qquad \hat{\mathbf{z}}_i = f(\mathbf{x}_i;\phi)$$

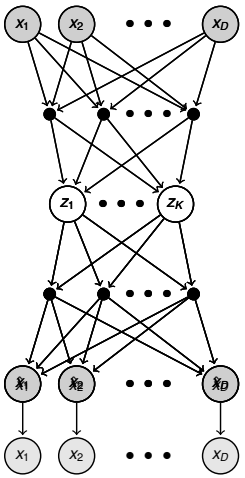
  Update generative parameters θ according to $\nabla_\theta \mathcal{F}(\{\hat{\mathbf{z}}_i\},\theta)$.
- Sleep phase: sample $\{\mathbf{z}_s,\mathbf{x}_s\}_{s=1}^S$ from current generative model. Update recognition parameters ϕ to direct $f(\mathbf{x}_s)$ towards $\mathbf{z}_s$ (simple gradient learning).

  $$\Delta\phi \propto \sum_s (\mathbf{z}_s - f(\mathbf{x}_s;\phi))\nabla_\phi f(\mathbf{x}_s;\phi)$$

The Helmholtz Machine

- Can sample $\mathbf{z}$ from recognition model rather than just evaluate means.
  - Expectations in free-energy can be computed directly rather than by mean substitution.
  - In hierarchical models, output of higher recognition layers then depends on samples at previous stages, which introduces correlations between samples at different layers.
- Recognition model structure need not exactly echo generative model.
- More general approach is to train f to yield mean parameters of ExpFam $q(\mathbf{z})$ (later).
- Sleep phase learning minimises $\mathbf{KL}[p_\theta(\mathbf{z}|\mathbf{x})\|q(\mathbf{z}; f(\mathbf{x}, \phi))]$. Opposite to variational objective, but may not matter if divergence is small enough.

Variational Autoencoders



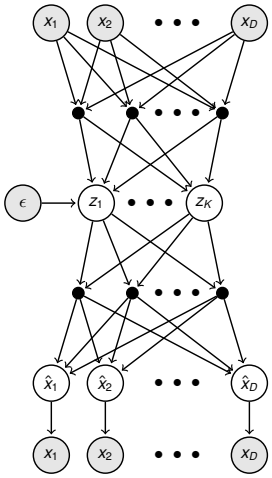
- Fuse wake and sleep phases, optimising $\mathcal{F}$ wrt generative and recognition parameters using reparametrisation.
- Canonical generative conditional is Gaussian with NN (usually MLP) mean (variance may also be parametrised by another NN):

$$\mathrm{P}(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathrm{I})$$
$$\mathrm{P}(\mathbf{x}|\mathbf{z}) = \mathcal{N}\left(\mathbf{g}_{\mathrm{NN}}(\mathbf{z}; \boldsymbol{\theta}), \sigma^2 \mathrm{I}\right)$$

- NN recognition model estimates parameters of posterior: $q(\mathbf{z}|\mathbf{x}; \mathbf{f}(\mathbf{x}, \phi))$.
- Free energy:

$$\mathcal{F}(\theta, \phi) = \sum_{\text{data}} \langle \log \mathrm{P}(\mathbf{x}|\mathbf{z}) \rangle_{q(\mathbf{z}|\mathbf{x})} - \mathbf{KL}[q(\mathbf{z}|\mathbf{x})\|\mathrm{P}(\mathbf{z})]$$
$$= -\sum_{\text{data}} \underbrace{\left\langle \frac{\|\mathbf{x} - \hat{\mathbf{x}}(\mathbf{z})\|^2}{2\sigma^2} \right\rangle_q}_{\text{“reconstruction cost”}} + \underbrace{\mathbf{KL}[q(\mathbf{z}|\mathbf{x})\|\mathrm{P}(\mathbf{z})]}_{\text{“regulariser”}}$$

Variational Autoencoders



- The expectation of a non-linear (NN) function is intractable.
- /reparam/ Generate S samples from $q(\mathbf{z}|\mathbf{x})$ using deterministic transformation of standard random variates (reparametrisation trick).
  - E.g. if $\mathbf{f}$ gives marginal μ_i and σ_i for latents z_i and $\epsilon_i^s \sim \mathcal{N}(0, 1)$, then $z_i^s = \mu_i + \sigma_i \epsilon_i^s$.
- Now generative and recognition parameters can be trained together by gradient descent (backprop), holding ϵ^s fixed.

$$\mathcal{F}_i(\theta, \phi) = \frac{1}{S} \sum_s \log P(\mathbf{x}_i, \mathbf{z}_i^s; \theta) - \log q(\mathbf{z}_i^s; \mathbf{f}(\mathbf{x}_i, \phi))$$
$$\frac{\partial}{\partial \theta} \mathcal{F}_i = \frac{1}{S} \sum_s \nabla_\theta \log P(\mathbf{x}_i, \mathbf{z}_i^s; \theta)$$
$$\frac{\partial}{\partial \phi} \mathcal{F}_i = \frac{1}{S} \sum_s \frac{\partial}{\partial \mathbf{z}_i^s} (\log P(\mathbf{x}_i, \mathbf{z}_i^s; \theta) - \log q(\mathbf{z}_i^s; \mathbf{f}(\mathbf{x}_i))) \frac{d\mathbf{z}_i^s}{d\phi}$$
$$+ \frac{\partial}{\partial \mathbf{f}(\mathbf{x}_i)} \log q(\mathbf{z}_i^s; \mathbf{f}(\mathbf{x}_i)) \frac{d\mathbf{f}(\mathbf{x}_i)}{d\phi}$$

Variational Autoencoders

- Frozen samples ϵ^s can be redrawn to avoid overfitting.
- May be possible to evaluate entropy and $\langle \log P(\mathbf{z}) \rangle$ without sampling, reducing variance.
- Differentiable reparametrisations are available for a number of different distributions.
  - requires approximation for discrete-valued variables (Gumbel or “concrete” distributions)
- Conditional $P(\mathbf{x}|\mathbf{z}, \theta)$ may be more complex: RNNs, transformers, ...
  - May include internal stochastic nodes: requires recognition network to estimate all distributions (see “ladder VAE”).
  - In practice, hierarchical models appear difficult to learn.

More recent work

- Changing the variational cost function (tightening the bound):
  - Importance-Weighted autoencoder (IWAE)
  - Filtering variational objective (FIVO)
  - Thermodynamic variational objective (TVO)
- Flexible variational distributions (and avoiding inference)
  - Normalising flows
  - DDC-Helmholtz machine
  - Amortised learning
  - Diffusion models
- Structured generative models
  - "standard" VAE generative model both too powerful and too simple for learning
  - local conjugate inference – structured VAEs
- Recognition-parametrised models
  - RPMs model (latent-induced) joint dependence, but not marginals of observations

Far from exhaustive . . . these are all areas of active research. We'll survey a few ideas.

Importance-weighted free energy

Another interpretation of $\mathcal{F}$: Jensen bound on importance sampled estimate.

$$\ell(\theta) = \log \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}[p(\mathbf{x})] = \log \mathbb{E}_{\mathbf{z}\sim q}\left[\frac{p(\mathbf{x})p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})}\right] \geq \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]$$

So

$$\mathcal{F}(q,\theta) = \left\langle \log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} \right\rangle_q = \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}) \tfrac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})}\right]$$

-

Suggests more accurate importance sampling:

$$\ell(\theta) = \log \mathbb{E}_{\mathbf{z}_1\ldots\mathbf{z}_K \overset{\text{iid}}{\sim} q}\left[\frac{1}{K}\sum_k \frac{p(\mathbf{x},\mathbf{z}_k)}{q(\mathbf{z}_k)}\right] \geq \mathbb{E}_{\mathbf{z}_1\ldots\mathbf{z}_K \overset{\text{iid}}{\sim} q}\left[\log \frac{1}{K}\sum_k \frac{p(\mathbf{x},\mathbf{z}_k)}{q(\mathbf{z}_k)}\right]$$

Tighter bound, and reparametrisation friendly, but as $K \to \infty$ the signal for learning amortised q grows weaker so VAE learning doesn't always improve.

Normalising flows

$$\mathcal{F}(q,\theta) = \langle \log p(\mathbf{x},\mathbf{z}|\theta)\rangle_q - \langle \log q(\mathbf{z})\rangle_q$$

To evaluate $\mathcal{F}$ (or its gradients) we need to be able to find expectations wrt q (e.g. by Monte Carlo) and evaluate the log-density – usually restricts us to tractable inferential families.

Consider defining a recognition model $q(\mathbf{z})$ implicitly by:

$$\begin{aligned} \mathbf{z}_0 &\sim q_0(\cdot;\mathbf{x}) && \leftarrow \text{ fixed, tractable, e.g. } \mathcal{N}(\mathbf{x}, I) \\ \mathbf{z} &= f_K(f_{K-1}(\ldots f_1(\mathbf{z}_0))) && \leftarrow f_k \text{ smooth, invertible, parametrised by } \phi \end{aligned}$$

Then we can both compute expectations under q and evaluate its log density:

$$\langle F(\mathbf{z})\rangle_q = \langle F(f_K(f_{K-1}(\ldots f_1(\mathbf{z}_0))))\rangle_{q_0}$$

$$\log q(\mathbf{z}) = \log q_0(f_1^{-1}(f_2^{-1}(\ldots f_K^{-1}(\mathbf{z})))) - \sum_k \log|\nabla f_k|$$

where the second result applies from repeated transformations of variables

$$\mathbf{z}_k = f_k(\mathbf{z}_{k-1}) \;\Rightarrow\; q(\mathbf{z}_k) = q(f_k^{-1}(\mathbf{z}_k))\left|\frac{\partial \mathbf{z}_{k-1}}{\partial \mathbf{z}_k}\right| = q(f_k^{-1}(\mathbf{z}_k))|\nabla f_k(\mathbf{z}_{k-1})|^{-1}$$

Normalising flows

So, given a sample $\mathbf{z}_0^s \overset{\text{iid}}{\sim} q_0(\cdot;\mathbf{x})$:

$$\mathcal{F}(\phi,\theta) \approx \frac{1}{S}\sum_s \log p(\mathbf{x}, f_K(\ldots f_1(\mathbf{z}_0^s))) + \mathbf{H}[q_0] + \frac{1}{S}\sum_s\sum_k \log\left|\nabla f_k(f_{k-1}(\ldots f_1(\mathbf{z}_0^s)))\right|$$

and we can compute gradients of this expression wrt θ and ϕ.

Useful fs (from Rezende & Mohammed 2015):

$$\begin{aligned} f(\mathbf{z}) &= \mathbf{z} + \mathbf{u}h(\mathbf{w}^\mathsf{T}\mathbf{z} + b) && \Rightarrow |\nabla f| = \left|1 + \mathbf{u}^\mathsf{T}\psi(\mathbf{z})\right| && \psi(\mathbf{z}) = h'(\mathbf{w}^\mathsf{T}\mathbf{z} + b)\mathbf{w} \\ f(\mathbf{z}) &= \mathbf{z} + \frac{\beta}{\alpha + |\mathbf{z} - \mathbf{z}_0|} && \Rightarrow |\nabla f| = [1 + \beta h]^{d-1}[1 + \beta h + \beta h' r] \\ & && && r = |\mathbf{z} - \mathbf{z}_0|, h = \frac{1}{\alpha + r} \end{aligned}$$

Both can be cascaded to give a flexible variational family.

Diffusion probabilistic models

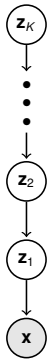
Multi-stage flexible generative process (similar to a normalising flow) but defined by *recognition* model. Recent papers suggest this approach outperforms auto-regressive approaches, including those based on transformers, on image and audio likelihood.
In our notation:

- Define observations $\mathbf{x}$ and latents $\mathbf{z}_1 \dots \mathbf{z}_T$.
- Fix "diffusion" recognition model (the "forward" model)

$$q(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}\left(\sqrt{1-\beta_1}\mathbf{x}, \beta_1 \mathsf{I}\right)$$
$$q(\mathbf{z}_k|\mathbf{z}_{k-1}) = \mathcal{N}\left(\sqrt{1-\beta_k}\mathbf{z}_{k-1}, \beta_k \mathsf{I}\right)$$

- Parametrise generative model (the "backward" model)

$$p(\mathbf{z}_K) = \mathcal{N}(\mathbf{0}, \mathsf{I})$$
$$p(\mathbf{z}_{k-1}|\mathbf{z}_k;\theta) = \mathcal{N}(\mu_\theta(\mathbf{z}_k,k), \Sigma_\theta(\mathbf{z}_k,k))$$
$$p(\mathbf{x}|\mathbf{z}_1;\theta) = \mathcal{N}(\mu_\theta(\mathbf{z}_1,1), \Sigma_\theta(\mathbf{z}_1,1))$$



Diffusion recognition sends $q(\mathbf{z}_K) \overset{K\to\infty}{\to} \mathcal{N}(\mathbf{0}, \mathsf{I})$.
In the limit $\beta_k \to 0$ the reciprocal normal generation is correct.
But as $\beta \to 0$ and $K \to \infty$ the link between observation and $\mathbf{z}_K$ becomes uninformative.

Diffusion models

Free energy

$$\mathcal{F} = \left\langle \log p(\mathbf{x}|\mathbf{z}_1) + \sum_{k=2}^{K} \log p(\mathbf{z}_{k-1}|\mathbf{z}_K) + \log p(\mathbf{z}_k) \right\rangle_{q(\mathbf{z}_{1:K}|\mathbf{x})} - \mathbf{H}[q(\mathbf{z}_{1:K}|\mathbf{x})]$$
$$\mathcal{F} = \langle \log p(\mathbf{x}|\mathbf{z}_1)\rangle_{q(\mathbf{z}_1|\mathbf{x})} + \sum_{k=2}^{K} \langle \log p(\mathbf{z}_{k-1}|\mathbf{z}_k)\rangle_{q(\mathbf{z}_k,\mathbf{z}_{k-1}|\mathbf{x})} + \langle \log p(\mathbf{z}_K)\rangle_{q(\mathbf{z}_K|\mathbf{x})} - \sum_{k=1}^{K} \mathbf{H}[\cdot]$$

So learning requires expectations (usually based on samples) under $q(\mathbf{z}_k)$. The diffusion assumption makes these marginals easy to compute. Let $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$.

$$q(\mathbf{z}_k|\mathbf{x}) = \mathcal{N}(\sqrt{\alpha_k}\mathbf{x}, (1-\bar{\alpha}_k)\mathsf{I})$$
$$q(\mathbf{z}_{k-1}|\mathbf{z}_k, \mathbf{x}) = \mathcal{N}\left(\frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\mathbf{x} + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\mathbf{z}_k, \frac{(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\right)$$

- Given samples of $\mathbf{z}_k$ the (closed form) conditional for $q(\mathbf{z}_{k-1}|\mathbf{z}_k, \mathbf{x})$ gives an efficient update for $\log p(\mathbf{z}_{k-1}|\mathbf{z}_k)$.
- Reparametrisation (as in the VAE) makes it possible to optimise β_k.

DDC Helmholtz machine

A (loosely) neurally inspired idea. Define q as an unnormalisable exponential family with a large set of sufficient statistics

$$q(\mathbf{z}) \propto e^{\sum_i \eta_i \psi_i(\mathbf{z})}$$

and parametrise by mean parameters $\boldsymbol{\mu} = \langle \boldsymbol{\psi}(\mathbf{z})\rangle$: Distributed distributional code (DDC).

Train recognition model using sleep samples:

$$\boldsymbol{\mu} = \langle \boldsymbol{\psi}(\mathbf{z})\rangle_q = f(\mathbf{x};\phi)$$
$$\Delta\phi \propto \sum_s (\boldsymbol{\psi}(\mathbf{z}_s) - f(\mathbf{x}_s;\phi))\nabla_\phi f(\mathbf{x}_s;\phi)$$

Also learn linear approximation $\nabla \log p(\mathbf{x},\mathbf{z}|\theta) \approx A\boldsymbol{\psi}(\mathbf{z})$

$$A = \left(\sum_s \nabla \log p(\mathbf{x}_s,\mathbf{z}_s|\theta)\boldsymbol{\psi}(\mathbf{z}_s)\right)^\mathsf{T}\left(\sum_s \boldsymbol{\psi}(\mathbf{z}_s)\boldsymbol{\psi}(\mathbf{z}_s)^\mathsf{T}\right)^{-1}$$

Then

$$\langle \nabla \log p(\mathbf{x},\mathbf{z})\rangle_q \approx A\langle \boldsymbol{\psi}(\mathbf{z})\rangle_q \approx Af(\mathbf{x},\phi)$$

Approach can be generalised to an infinite dimensional $\boldsymbol{\psi}$ using the kernel trick.

Amortised Learning

If we aren't actually interested in inference, we can short-circuit general recognition and compute expectations for learning directly.

$$\nabla_\theta \ell(\theta) = \partial_\theta \mathcal{F}(q^*,\theta) = \partial_\theta \langle \log p(\mathcal{X},\mathcal{Z}|\theta)\rangle_{q^*} = \langle \partial_\theta \log p(\mathcal{X},\mathcal{Z}|\theta)\rangle_{p(\mathcal{Z}|\mathcal{X},\theta)}$$

Suggests a wake-sleep approach:

- Sample $\{\mathbf{x}_s, \mathbf{z}_s\} \sim p(\mathcal{X},\mathcal{Z}|\theta^k)$.
- Train regressor $\hat{J}_{\theta^k} : \mathbf{x}_s \mapsto \nabla_\theta \log p(\mathbf{x}_s,\mathbf{z}_s|\theta)|_{\theta^k}$
  (or, for specific regressors, $\mapsto \log p(\mathbf{x}_s,\mathbf{z}_s|\theta^k)$ and differentiate prediction)
- Set $\theta^{k+1} = \theta^k + \alpha \sum_i \hat{J}_{\theta^k}(\mathbf{x}_i)$
  (or $= \theta^k + \alpha \sum_i \nabla_\theta \hat{J}_\theta(\mathbf{x}_i)|_{\theta^k}$).

Derivative form works for (kernel/GP) regression for which regressor is linear in targets.

For conditional exponential family models

$$\log p(\mathcal{X},\mathcal{Z}|\theta) = \eta(\mathbf{z},\theta)^\mathsf{T}\mathbf{T}(\mathbf{x}) - \Phi(\mathbf{z},\theta) + \log p(\mathbf{z}|\theta)$$
$$\Rightarrow \langle \log p(\mathcal{X},\mathcal{Z}|\theta)\rangle_{q^*} = \langle \eta(\mathbf{z},\theta)\rangle_{q^*}^\mathsf{T}\mathbf{T}(\mathbf{x}) - \langle \Phi(\mathbf{z},\theta) + \log p(\mathbf{z}|\theta)\rangle_{q^*}$$

and regressors can be trained to functions of $\mathbf{z}$ alone, with $T(\mathbf{x})$ then evaluated on (wake-phase) data.

Generative models

In practice, much of the VAE and related work has used a common generative model:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$$
$$\mathbf{x} \sim \mathcal{N}(\mathbf{g}(\mathbf{z};\theta), \psi I)$$

where g is a neural network.

- Overcomplicated: if $\dim(\mathbf{z})$ is large enough the optimal solution has $\psi \to 0$, $q(\mathbf{z};\mathbf{x}) \to \delta(\mathbf{z} - f(\mathbf{x},\phi))$. In effect, the generative model learns a flow to transform a normal density to the target.
- Oversimplified: if $\dim(\mathbf{z})$ is small, this is just non-linear PCA!

Interesting latent representations are likely to require more structured generative models. Recent work has approached such models in both VAE and DDC frameworks.

Structured VAEs

Consider a model where $p(\mathcal{Z}|\theta)$ has tractable joint exponential-family potentials and

$$p(\mathcal{X}|\mathcal{Z},\Gamma) = \prod_i p(\mathbf{x}_i|\mathbf{z}_i,\gamma_i)$$

are intractable (say neural net + normal) cond ind observations. γ_i might be the same for all i.

Consider factored variational inference $q(\mathcal{Z}) = \prod_i q_i(\mathbf{z}_i)$. With no further constraint,

$$\begin{aligned}\log q_i^*(\mathbf{z}_i) &\underset{+C}{=} \langle \log p(\mathcal{Z},\mathcal{X})\rangle_{q_{\neg i}} \underset{+C}{=} \langle \log p(\mathbf{z}_i|\mathcal{Z}_{\neg i}) + \log p(\mathbf{x}_i|\mathbf{z}_i)\rangle_{q_{\neg i}} \\ &\underset{+C}{=} \langle \boldsymbol{\eta}_{\neg i}\rangle_{q_{\neg i}}^{\mathsf{T}} \boldsymbol{\psi}_i(\mathbf{z}_i) + \log p(\mathbf{x}_i|\mathbf{z}_i)\end{aligned}$$

where we have exploited the exponential-family form of $p(\mathcal{Z})$. $\boldsymbol{\psi}_i$ are effective suff stats – including log normalisers of children in a DAG; $\boldsymbol{\eta}_{\neg i}$ is a function of $\mathcal{Z}_{\neg i}$.

Now, choose the parametric form $q_i(\mathbf{z}_i) = e^{\tilde{\boldsymbol{\eta}}_i^{\mathsf{T}}\boldsymbol{\psi}_i(\mathbf{z}_i) - \Phi_i(\tilde{\boldsymbol{\eta}}_i)}$. Constrained optimum has form

$$\log q_i^*(\mathbf{z}_i) \underset{+C}{=} \langle \boldsymbol{\eta}_{\neg i}\rangle_{q_{\neg i}}^{\mathsf{T}} \boldsymbol{\psi}_i(\mathbf{z}_i) + \boldsymbol{\rho}(\mathbf{x}_i)^{\mathsf{T}} \boldsymbol{\psi}_i(\mathbf{z}_i)$$

for some $\mathbf{x}_i$-dependent natural parameter. Introduce recognition models:

$$\boldsymbol{\rho}(\mathbf{x}_i) = f_i(\mathbf{x}_i, \phi_i)$$

Recognition function f_i might be same for all i if all likelihoods are the same (*e.g.* HMM).

Structured VAE learning

Now, the free-energy can be written as a function of parameters and recognition parameters:

$$\begin{aligned}\mathcal{F}(\theta,\Gamma,\{\phi_i\}) &= \left\langle \sum_i \log p(\mathbf{x}_i|\mathbf{z}_i,\gamma_i) + \log p(\mathcal{Z}|\theta) \right\rangle_{q(\mathcal{Z};\theta,\{\phi_i\})} + \sum_i \mathbf{H}[q_i] \\ &= \sum_i \underbrace{\langle \log p(\mathbf{x}_i|\mathbf{z}_i,\gamma_i)\rangle_{q_i(\mathbf{z}_i;\theta,\phi_i)} + \mathbf{H}[q_i]}_{\mathcal{F}_i} + \langle \log p(\mathcal{Z}|\theta)\rangle_{q(\mathcal{Z};\theta,\{\phi_i\})}\end{aligned}$$

Updates on θ are just as for tractable model.

To update each ϕ_i and γ_i, find $\langle \boldsymbol{\eta}_{\neg i}\rangle_{q_{\neg i}}$ to give the "prior". Generate reparametrised samples $\mathbf{z}_i^s \sim q_i$. Then

$$\frac{\partial}{\partial \gamma_i}\mathcal{F}_i = \sum_s \nabla_{\gamma_i} \log p(\mathbf{x}_i, \mathbf{z}_i^s; \gamma_i)$$

$$\frac{\partial}{\partial \phi_i}\mathcal{F}_i = \sum_s \frac{\partial}{\partial \mathbf{z}_i^s}(\log p(\mathbf{x}_i,\mathbf{z}_i^s;\gamma_i) - \log q(\mathbf{z}_i^s;\mathbf{f}(\mathbf{x}_i)))\frac{d\mathbf{z}_i^s}{d\phi} + \frac{\partial}{\partial \mathbf{f}(\mathbf{x}_i)} \log q(\mathbf{z}_i^s;\mathbf{f}(\mathbf{x}_i))\frac{d\mathbf{f}(\mathbf{x}_i)}{d\phi}$$

as for the standard VAE.

Recognition-parametrised models

Observations $\mathcal{X} = \{\mathbf{x}_j : j = 1 \ldots J\}$ and latents $\mathcal{Z} = \{\mathbf{z}_l : l = 1 \ldots L\}$. Assume all $\mathbf{x}_j \perp\!\!\!\perp \mathbf{x}_{j'} | \mathcal{Z}$.

Factor form: $\mathrm{P}(\mathcal{X},\mathcal{Z}) = \psi^z(\mathcal{Z}) \prod_j \psi_j^x(\mathbf{x}_j) \prod_j \psi_j^{xz}(\mathbf{x}_j,\mathcal{Z})$.

The RPM defines factors:

$\psi^z(\mathcal{Z}) \to p_{\theta_z}(\mathcal{Z})$: normalised (possibly structured) prior.

$\psi_j^x(\mathbf{x}_j) \to p_0(\mathbf{x}_j)$: empirical marginal: $p_0(\mathbf{x}_j) = \frac{1}{N}\sum_n \delta(\mathbf{x}_j - \mathbf{x}_j^{(n)})$

$\psi^{xz}(\mathbf{x}_j,\mathcal{Z}) \to \frac{f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)}{\int d\mathbf{x}_j\, p_0(\mathbf{x}_j) f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)}$: $f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)$ parametrised distribution possibly defined on a subset of the $\mathcal{Z}$ (often a single $\mathbf{z}_l$).

Full joint model depends on observed dataset $\mathbb{X}^{(N)} = \{\mathcal{X}^{(1)} \ldots \mathcal{X}^{(N)}\}$

$$\mathrm{P}_{\mathbb{X}^{(N)},\theta}(\mathcal{X},\mathcal{Z}) = p_{\theta_z}(\mathcal{Z}) \prod_j \left(p_0(\mathbf{x}_j) \frac{f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)}{f_{\theta_j}(\mathcal{Z})} \right),$$

- Generative model only implicit
- Models latent-dependent joint, but not individual marginals.

. . . just a brief survey of a subset of current ideas.

A few things we hope you've learned in this course . . .

- Exponential families are your friends.
- Latent variable models and conditional independence to uncover structured representations.
- Free-energies, maximum likelihood, variational approximation theory and variational Bayes.
- Message passing exploits conditional independence.
- A rich toolkit of approximations, that you can compose in novel and useful ways.
- A theory of many approximations that helps ensure you understand their use and limitations (and may help derive new approaches).