Nonparametric Bayesian Methods in Machine Learning: Gaussian Processes, Dirichlet Processes and Extensions

Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London

Center for Automated Learning and Discovery
Carnegie Mellon University

Work done in collaboration with

Wei Chu (Gatsby Unit)
Katherine Heller (Gatsby Unit)
Hyun-Chul Kim (POSTECH, Korea)
Tom Minka (MSR Cambridge)
Carl Edward Rasmussen (Max-Planck Tübingen)
Edward Snelson (Gatsby Unit)

Sheffield Machine Learning Workshop, Sep 2004
Outline

• Part I: Gaussian Processes
  – Part Ia: A Brief Tutorial on Gaussian Processes for Regression
  – Part Ib: Warped Gaussian Processes
  – Part Ic: Gaussian Processes for Ordinal Regression
  – Part Id: Other Extensions

• Part II: Dirichlet Process Mixtures
  – Part IIa: A Brief Tutorial on Dirichlet Process Mixtures
  – Part IIb: Bayesian Hierarchical Clustering and DPMs
  – Part IIc: Expectation Propagation for DPMs

• Conclusions and Future Work
Part Ia: A Brief Tutorial on Gaussian Process for Regression

Two ways of understanding Gaussian processes (GPs)...  
- Starting from multivariate Gaussians  
- Starting from linear regression
...from multivariate Gaussians to GPs...

univariate Gaussian density for $t$

$$
p(t) = (2\pi \sigma^2)^{-1/2} \exp \left\{ -\frac{t^2}{2\sigma^2} \right\}
$$

multivariate Gaussian density for $t = (t_1, t_2, t_3, \ldots t_N)^\top$

$$
p(t) = |2\pi \Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} t^\top \Sigma^{-1} t \right\}
$$

$\Sigma$ is an $N \times N$ covariance matrix.

Imagine that $\Sigma_{ij}$ depends on $i$ and $j$ and we plot samples of $t$ as if they were functions...

gpdemo2 gpdemo
...from linear regression to GPs...

- Linear regression with inputs $x_i$ and outputs $t_i$:
  \[ t_i = \sum_d w_d x_{id} + \epsilon_i \]

- Linear regression with basis functions ("kernel trick"): $t_i = \sum_d w_d \phi_d(x_i) + \epsilon_i$

- **Bayesian** linear regression with basis functions:
  \[
  w_d \sim \mathcal{N}(0, \beta_d) \quad [ \text{independent of } w_\ell ], \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)
  \]

- Integrating out the weights, $w_d$, we find:
  \[
  E[t_i] = 0, \quad E[t_i t_j] = \sum_d \beta_d \phi_d(x_i) \phi_d(x_j) + \delta_{ij} \sigma^2 \equiv C_{ij}
  \]

This is a Gaussian process with covariance function $C(x_i, x_j) \equiv C_{ij}$.

This Gaussian process has a finite number of basis functions. Many useful GP covariance functions correspond to infinitely many basis functions.
Gaussian Process Regression

A Gaussian Process (GP) places a prior directly on the space of functions such that at any finite selection of points \(x^{(1)}, \ldots, x^{(N)}\) the corresponding function values \(t^{(1)}, \ldots, t^{(N)}\) have a multivariate Gaussian distribution.

The covariance between two function values \(t^{(i)}\) and \(t^{(j)}\) under the prior is given by the covariance function \(C(x^{(i)}, x^{(j)})\), which typically decays monotonically with \(\|x^{(i)} - x^{(j)}\|\), encoding smoothness.

GPs are “Bayesian Kernel Regression Machines”
Gaussian processes for regression

Given dataset \( D = (X_N, t_N) \), where \( X_N \equiv \{x^{(n)}\}_{n=1}^N \) and \( t_N \equiv \{t^{(n)}\}_{n=1}^N \).

We wish to predict \( t^{(N+1)} \) given a new input vector \( x^{(N+1)} \), i.e. \( p(t^{(N+1)}|x^{(N+1)}, D) \).

Assuming Gaussian noise, the joint on \( (t_N, t^{(N+1)}) \) is a Gaussian.

A typical covariance function is:

\[
C_{mn} = v_1 \exp \left[ - \sum_{d=1}^{D} \left( \frac{x_d^{(m)} - x_d^{(n)}}{r_d} \right)^2 \right] + v_0 \delta_{mn}.
\]

With parameters \( \Theta = (v_0, v_1, r_1, \ldots, r_D) \).
Training and making predictions with a GP

The (negative log) likelihood is used to train the model, usually by conjugate gradient descent on $\Theta$, the parameters of the covariance matrix $C_N$:

$$L(\Theta) = -\log p(t_N|X_N, \Theta) = \frac{1}{2} \log \det C_N + \frac{1}{2} t_N^T C_N^{-1} t_N + \frac{N}{2} \log 2\pi.$$ 

The predictive distribution for a new point given the observed data, $p(t^{(N+1)}|t_N, X_{N+1}, \Theta)$, is also Gaussian.

The mean and variance are calculated from the inverse covariance matrix $C_{N+1}^{-1}$, and the target values $t_N$. 
Using Gaussian Processes for Classification

Binary classification problem: Given a data set $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, where $y^{(n)} \in \{-1, +1\}$, infer class label probabilities at new points.

There are many ways to relate function values $f^{(n)}$ to class probabilities:

$$p(y|f) = \begin{cases} \frac{1}{1 + \exp(-yf)} & \text{sigmoid (logistic)} \\ \Phi(yf) & \text{cumulative normal (probit)} \\ H(yf) & \text{threshold} \\ \epsilon + (1 - 2\epsilon)H(yf) & \text{robust threshold} \end{cases}$$
Part Ib: Warped Gaussian Process Regression

(with Ed Snelson, 2004)
Motivations

- The Gaussian process (GP) is a way of performing \textit{nonlinear nonparametric regression}, which gives excellent performance, and allows \textit{full Bayesian predictions}

- However, it assumes the target data to be distributed as a multivariate Gaussian, with Gaussian noise

- This is often an unreasonable assumption to make for the raw data

- Standard practice is to apply a \textit{preprocessing transformation} in an ad-hoc manner preceding the modelling with the GP

- The warped GP incorporates the preprocessing as an \textit{integral part of the probabilistic model}

- This makes GPs more flexible so they can model \textit{non-Gaussian processes}
Warping the observation space

1. We assume a vector of latent targets $z_N$ that is modelled by a GP.

2. We make a transformation from the true observation space $t$ to the latent space $z$ by mapping each observation through the same function $f$,

$$z^{(n)} = f(t^{(n)}; \Psi) \quad \forall n.$$ 

3. We require $f$ to be monotonic and mapping on to the whole of the real line; otherwise probability measure will not be conserved in the transformation, and we will not induce a valid distribution over the targets.
Training the warped GP

The negative log likelihoods:

**GP:**  \[ L(\Theta) = - \log p(t_N|X_N, \Theta) = \frac{1}{2} \log \det C_N + \frac{1}{2} t_N^\top C_N^{-1} t_N + \frac{N}{2} \log 2\pi , \]

**WGP:**  \[ L(\Theta, \Psi) = - \log p(t_N|X_N, \Theta, \Psi) = \frac{1}{2} \log \det C_N + \frac{1}{2} f(t_N)^\top C_N^{-1} f(t_N) - \sum_{n=1}^{N} \log \left. \frac{\partial f(t)}{\partial t} \right|_{t_{(n)}} + \frac{N}{2} \log 2\pi \]

This is used to train the warped GP exactly as for the regular GP - both the hyperparameters of the covariance matrix, and the parameters of the warping function are learnt simultaneously.
Predictions with the warped GP

The predictive distribution at a new point in *latent space* is as for a regular GP:

\[
p(z^{(N+1)} | x^{(N+1)}, D, \Theta) = \mathcal{N}\left(\hat{z}^{(N+1)}, \sigma_{N+1}^2\right)
\]

To find the distribution in the *observation space* we pass that Gaussian through the nonlinear warping function, giving

\[
p(t^{(N+1)} | x^{(N+1)}, D, \Theta, \Psi) = \frac{f'(t^{(N+1)})}{\sqrt{2\pi\sigma^2_{N+1}}} \exp\left[ -\frac{1}{2} \left( \frac{f(t^{(N+1)}) - \hat{z}^{(N+1)}}{\sigma_{N+1}} \right)^2 \right].
\]
Choosing the warping function

The warping function has to be chosen according to some constraints:

• monotonicity

• covering the whole real line

• derivatives with respect to $t$ and the parameters $\Psi$ easily evaluated.

We chose a neural net style sum of $\tanh$ functions, although several other choices are possible:

$$f(t; \Psi) = t + \sum_{i=1}^{I} a_i \tanh (b_i (t + c_i)) \quad a_i, b_i \geq 0 \quad \forall i.$$
A simple 1D regression task

- 101 points, regularly spaced from $-\pi$ to $\pi$ on the $x$ axis, were generated with Gaussian noise about a sine function.

- Warped these points through $t = z^{1/3}$ to get this data.
### Results for some real datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$D$</th>
<th>$t_{\text{min}}$</th>
<th>$t_{\text{max}}$</th>
<th>$N_{\text{train}}$</th>
<th>$N_{\text{test}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>creep</td>
<td>30</td>
<td>18 MPa</td>
<td>530 MPa</td>
<td>800</td>
<td>1266</td>
</tr>
<tr>
<td>abalone</td>
<td>8</td>
<td>1 yr</td>
<td>29 yrs</td>
<td>1000</td>
<td>3177</td>
</tr>
<tr>
<td>ailerons</td>
<td>40</td>
<td>$-3.0 \times 10^{-3}$</td>
<td>$-3.5 \times 10^{-4}$</td>
<td>1000</td>
<td>6154</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Absolute error</th>
<th>Squared error</th>
<th>$-\log p(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>creep</td>
<td>GP</td>
<td>16.4</td>
<td>654</td>
<td>4.46</td>
</tr>
<tr>
<td></td>
<td>GP + log</td>
<td>15.6</td>
<td>587</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td>warped GP</td>
<td>15.0</td>
<td>554</td>
<td>4.19</td>
</tr>
<tr>
<td>abalone</td>
<td>GP</td>
<td>1.53</td>
<td>4.79</td>
<td>2.19</td>
</tr>
<tr>
<td></td>
<td>GP + log</td>
<td>1.48</td>
<td>4.62</td>
<td>2.01</td>
</tr>
<tr>
<td></td>
<td>warped GP</td>
<td>1.47</td>
<td>4.63</td>
<td>1.96</td>
</tr>
<tr>
<td>ailerons</td>
<td>GP</td>
<td>$1.23 \times 10^{-4}$</td>
<td>$3.05 \times 10^{-8}$</td>
<td>-7.31</td>
</tr>
<tr>
<td></td>
<td>warped GP</td>
<td>$1.18 \times 10^{-4}$</td>
<td>$2.72 \times 10^{-8}$</td>
<td>-7.45</td>
</tr>
</tbody>
</table>

(a) sine  
(b) creep  
(c) abalone  
(d) ailerons
Part Ib: Conclusions

- We have shown that the warped GP is capable of finding extra structure in the data through the transformations it learns.

- It allows standard preprocessing transforms, such as \( \log \), to be discovered automatically and improved on, rather than be applied in an ad-hoc manner.

- Computationally cheap to apply as an add-on to the regular GP.

- Forms of non-Gaussianity which do not directly depend on the output values (such as heavy tailed noise) are not captured by the method proposed here...

- ...but GPs can also be extended to include both warping and non-Gaussian noise.
Part Ic: Gaussian Processes for Ordinal Regression

(with Wei Chu, 2004)

- Ordinal Regression: target variable is discrete with values that are ordered.

- Examples:
  - Movie Rankings (0-5 stars)
  - Severity of Disease (low, medium, high)
  - Grades (A-F)

- Related to predicting rankings and Collaborative Filtering.
A Likelihood Function for Ordinal Regression

- Likelihood defined using thresholds on real line:

\[
P_1(y_i|f(x_i)) = \begin{cases} 
1 & b_{y_i-1} \leq f(x_i) \leq b_{y_i} \\ 
0 & \text{otherwise}
\end{cases}
\]

- Thresholds \( b_0 = -\infty < b_1 < b_2 < \ldots < b_{r-1} < b_r = +\infty \) divide the real line into \( r \) contiguous intervals. This maps the real function value \( f(x_i) \) into the discrete variable \( y_i \in \{1, \ldots r\} \) while enforcing the ordinal constraints.

- Adding Gaussian noise to \( f \) with zero mean and unknown variance \( \sigma^2 \):

\[
P(y_i|f(x_i)) = \int P_1(y_i|f(x_i) + \delta_i) \mathcal{N}(\delta_i; 0, \sigma^2) \, d\delta_i = \Phi(z_1^i) - \Phi(z_2^i)
\]

where \( z_1^i = \frac{b_{y_i-1} - f(x_i)}{\sigma} \), \( z_2^i = \frac{b_{y_i} - f(x_i)}{\sigma} \), and \( \Phi(z) = \int_{-\infty}^{z} \mathcal{N}(\zeta; 0, 1) \, d\zeta \).

- Note that binary classification is a special case of ordinal regression when \( r = 2 \), and in this case the likelihood function becomes the **probit function**.
Approximate Inference

• Given a Gaussian process prior $P(f)$ and this ordinal likelihood function, the posterior over function values is *non-Gaussian*:

$$P(f|D) \propto P(D|f)P(f)$$

• We approximate it using two different algorithms:
  – Laplace approximation
  – Expectation Propagation (EP)

• We compare to a competing SVM-based approach
Results: Abalone

Output discretized into four ordinal values ($\leq 6, = 8, = 11 \geq 14$).

Data set of 1983 points.

AAE=average absolute error (treating ranks as consecutive integers)

Error bars over 20 repetitions
Results: EachMovie Collaborative Filtering

Movie rankings (0 to 5 stars) from Compaq’s EachMovie service

\[ d = 72916 \text{ users (features)} \]
\[ n = 449 \text{ movies (data points)} \]
very sparse data 2.4% non-empty
test on new movies for user “id=52647”
Microarray data of 12600 genes that might be related to prostate cancer

\( n = 52 \) samples of prostate tumor

Gleason score \( y \) from 6-10 given by the pathologist reflecting the level of
differentiation of the glands in the prostate tumour.

Irrelevant genes were removed using ARD, down to 40 genes.

26 training data points, 26 test data points, 20 splits
Part Id: Other Extensions

- Mixtures of Gaussian Process Experts (with Carl Rasmussen)
- EM-EP for Gaussian Process Classification (with Hyun-Chul Kim)
Mixtures of Gaussian Process Experts

Motivation:

(with Carl Rasmussen, 2002)

1. Difficult to specify flexible GP covariance structures:

   eg, varying spatial frequency, varying signal amplitude, varying noise etc.

2. Predictions and training requires $C^{-1}$ which has $O(n^3)$ complexity.

Solution: the divide and conquer strategy of Mixture of Experts.

A (countably infinite) mixture of Gaussian Processes, allows:

- different covariance functions in different parts of space
- divide-and-conquer efficiency (by splitting $O(n^3)$ between experts).
EM-EP algorithm for Gaussian Process Classification
(with Hyun-Chul Kim, 2003)

Motivation: To learn the kernel parameters for classification.

An Approach:

• Treat the Gaussian process as a latent variable model.

• Use EM Algorithm
  – Infer the latent function values (E-step)
  – Optimize the kernel parameters (M-step)

• In the E step we approximate $P(f|D, \Theta)$ with a multivariate Gaussian $Q(f)$ using the Expectation Propagation (EP) algorithm

Related work by Opper and Winther; Csato; Minka; Seeger.
Conclusions

- Gaussian processes provide a general framework for non-parametric function approximation.

- Gaussian processes can be extended in several ways:
  - classification
  - warped regression
  - ordinal regression
  - mixtures of Gaussian process experts

- It is straightforward to infer the kernel from data using Bayesian methods (e.g. MCMC, Laplace, EM-EP)
Part II: Dirichlet Process Mixtures (Infinite Mixtures)
Part IIa: A Brief Tutorial on Dirichlet Process Mixtures

Consider using a \textit{finite mixture} of $K$ components to model a data set $\mathcal{D} = \{x^{(1)}, \ldots, x^{(n)}\}$

\[
p(x^{(i)}|\theta) = \sum_{j=1}^{K} \pi_j p_j(x^{(i)}|\theta_j)
= \sum_{j=1}^{K} P(s^{(i)} = j|\pi) p_j(x^{(i)}|\theta_j, s^{(i)} = j)
\]

The setting of the \texttt{indicator variable} $s^{(i)} = j$ means that data point $x^{(i)}$ was generated from Gaussian $j$.

Distribution of indicators $s = (s^{(1)}, \ldots, s^{(n)})$ given $\pi$ is \texttt{multinomial}

\[
P(s^{(1)}, \ldots, s^{(n)}|\pi) = \prod_{j=1}^{K} \pi_j^{n_j}, \quad n_j \overset{\text{def}}{=} \sum_{i=1}^{n} \delta(s^{(i)}, j).
\]
Dirichlet Process Mixtures (Infinite Mixtures) - II

Distribution of indicators $s = (s^{(1)}, \ldots, s^{(n)})$ given $\pi$ is multinomial

$$P(s^{(1)}, \ldots, s^{(n)}|\pi) = \prod_{j=1}^{K} \pi_{j}^{n_{j}}, \quad n_{j} \overset{\text{def}}{=} \sum_{i=1}^{n} \delta(s^{(i)}, j).$$

If we choose the mixing proportions $\pi$ to have a Dirichlet prior ...

$$p(\pi|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^{K}} \prod_{j=1}^{K} \pi_{j}^{\alpha/K-1}$$

... then we can integrate out the mixing proportions, $\pi$, to obtain

$$P(s^{(1)}, \ldots, s^{(n)}|\alpha) = \int d\pi \ P(s|\pi)P(\pi|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(n + \alpha)} \prod_{j=1}^{K} \frac{\Gamma(n_{j} + \alpha/K)}{\Gamma(\alpha/K)}$$

We can do this even if $K \to \infty$ !

This is an infinite mixture model (or Dirichlet Process Mixture)
Dirichlet Process Mixtures (Infinite Mixtures) - III

DPMs have the property that the prior probability of a new data point joining a cluster is proportional to the number of points already in that cluster.

Starting from

$$P(s|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(n + \alpha)} \prod_{j=1}^{K} \frac{\Gamma(n_j + \alpha/K)}{\Gamma(\alpha/K)}$$

- **Conditional Probabilities: Finite K**
  
  $$P(s^{(i)} = j|s_{-i}, \alpha) = \frac{n_{-i,j} + \alpha/K}{n - 1 + \alpha}$$

  where $s_{-i}$ denotes all indices except $i$, and $n_{-i,j} \overset{\text{def}}{=} \sum_{\ell \neq i} \delta(s^{(\ell)}, j)$

- **Conditional Probabilities: Infinite $K$**
  
  Taking the limit as $K \to \infty$ yields the conditionals

  $$P(s^{(i)} = j|s_{-i}, \alpha) = \begin{cases} \frac{n_{-i,j}}{n-1+\alpha} & j \text{ represented} \\ \frac{\alpha}{n-1+\alpha} & \text{all } j \text{ not represented} \end{cases}$$

  Left over mass, $\alpha$, ⇒ **countably infinite** number of indicator settings.
Inference in DPMs

- Markov chain Monte Carlo (Escobar and West 1995; Neal 1998; Rasmussen 2000)

- Expectation Propagation (Minka and Ghahramani, 2003)

- Variational Bayes (Blei and Jordan, 2004)

- Hierarchical Clustering (Ghahramani and Heller, 2004)
Hierarchical Clustering Algorithm:

Input: data $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$, model $p(\mathbf{x}|\theta)$, prior $p(\theta|\beta)$
Initialise number of clusters $c = n$, and $\mathcal{D}_i = \{\mathbf{x}^{(i)}\}$ for $i = 1 \ldots n$

while $c > 1$ do
  Find the pair $\mathcal{D}_i$ and $\mathcal{D}_j$ which have the highest posterior probability of merging:
  $$r_k = \frac{\pi_k p(\mathcal{D}_k|\mathcal{H}_1)}{p(\mathcal{D}_k|\mathcal{H}_k)}$$
  Merge $\mathcal{D}_k \leftarrow \mathcal{D}_i \cup \mathcal{D}_j$, $T_k \leftarrow (T_i, T_j)$
  Delete $\mathcal{D}_i$ and $\mathcal{D}_j$, $c \leftarrow c - 1$
end while
Output: Bayesian mixture model where each tree node is a mixture component
The tree can be cut at points where $r_k < 0.5$

When merging $T_i$ and $T_j$ to form $T_k$ the algorithm computes:

$$P(\mathcal{D}_k|T_k) = \pi_k P(\mathcal{D}_k|\mathcal{H}_1) + (1 - \pi_k) P(\mathcal{D}_i|T_i) P(\mathcal{D}_j|T_j)$$

where $\pi_k$ is the prior on merging which is computed recursively.

This estimates the marginal likelihood of the data $\mathcal{D}_k = \mathcal{D}_i \cup \mathcal{D}_j$. 
Inference in DPMs by Hierarchical Clustering

**Lemma**: The marginal likelihood of a Dirichlet Process Mixture is:

\[
p(D_k) = \sum_{v \in \mathcal{V}} p(D_k | v) p(v) = \sum_{v \in \mathcal{V}} \prod_{\ell=1}^{m_v} p(D^v_\ell) \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n^v_\ell)}{\Gamma(n_k + \alpha) \Gamma(\alpha)}
\]

where \( \mathcal{V} \) is the set of all *partitionings* of data set \( D_k \).

**Theorem**: The Bayesian Hierarchical Clustering algorithm computes:

\[
p(D_k | T_k) = \sum_{v \in \mathcal{V}_k} \prod_{\ell=1}^{m_v} p(D^v_\ell) \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n^v_\ell)}{d_k}
\]

where \( \mathcal{V}_k \) is the set of all *tree-consistent partitionings* of data set \( D_k \) for tree \( T_k \), and \( d_k \) is computed recursively.

**Corollary**: For any tree \( T_k \), the following is a lower bound on the marginal likelihood of a DPM:

\[
d_k \frac{\Gamma(\alpha)}{\Gamma(n_k + \alpha)} p(D_k | T_k) \leq p(D_k)
\]

**Note**: This is a *new* lower bound which can be simply and deterministically computed in \( O(n \log n) \). A greedy tree can be built in \( O(n^2) \).
Consider the above tree and all 15 possible partitions of \{1,2,3,4\}:

- (2 4)(1)(3),  (3 4)(1)(2),  (1 2)(3 4),  (1 3)(2 4),  (1 4)(2 3),
- (1 2 3)(4),  (1 2 4)(3),  (1 3 4)(2),  (2 3 4)(1),  (1 2 3 4)

- (1 2) (3) (4) and (1 2 3) (4) are tree-consistent partitions
- (1)(2 3)(4) and (1 3)(2 4) are not tree-consistent partitions
Part IIc: Inference in DPMs by Expectation Propagation

(with Tom Minka, 2003)

EP for Infinite Mixtures
Expectation Propagation (EP)

Data (iid) \( \mathcal{D} = \{ x^{(1)} \ldots, x^{(N)} \} \), model \( p(x|\theta) \), with parameter prior \( p(\theta) \).

The parameter posterior is:

\[
p(\theta|\mathcal{D}) = \frac{1}{p(\mathcal{D})} p(\theta) \prod_{i=1}^{N} p(x^{(i)}|\theta)
\]

We can write this as product of factors over \( \theta \):

\[
p(\theta) \prod_{i=1}^{N} p(x^{(i)}|\theta) = \prod_{i=0}^{N} f_i(\theta)
\]

where \( f_0(\theta) \overset{\text{def}}{=} p(\theta) \) and \( f_i(\theta) \overset{\text{def}}{=} p(x^{(i)}|\theta) \) and we will ignore the constants.

We wish to approximate this by a product of \textit{simpler} terms:

\[
q(\theta) \overset{\text{def}}{=} \prod_{i=0}^{N} \tilde{f}_i(\theta)
\]

\[
\min_{q(\theta)} \text{KL} \left( \prod_{i=0}^{N} f_i(\theta) \left\| \prod_{i=0}^{N} \tilde{f}_i(\theta) \right\| \right) \quad \text{(intractable)}
\]

\[
\min_{\tilde{f}_i(\theta)} \text{KL} \left( f_i(\theta) \left\| \tilde{f}_i(\theta) \right\| \right) \quad \text{(simple, non-iterative, inaccurate)}
\]

\[
\min_{\tilde{f}_i(\theta)} \text{KL} \left( f_i(\theta) \prod_{j \neq i} \tilde{f}_j(\theta) \left\| \tilde{f}_i(\theta) \prod_{j \neq i} \tilde{f}_j(\theta) \right\| \right) \quad \text{(simple, iterative, accurate)} \leftarrow \text{EP}
\]
Expectation Propagation

Input $f_0(\theta) \ldots f_N(\theta)$
Initialize $\tilde{f}_0(\theta) = f_0(\theta), \tilde{f}_i(\theta) = 1$ for $i > 0$, $q(\theta) = \prod_i \tilde{f}_i(\theta)$
repeat
  for $i = 0 \ldots N$ do
    Deletion: $q_{\backslash i}(\theta) \leftarrow \frac{q(\theta)}{\tilde{f}_i(\theta)} = \prod_{j \neq i} \tilde{f}_j(\theta)$
    Projection: $\tilde{f}^\text{new}_i(\theta) \leftarrow \arg \min_{f(\theta)} \text{KL}(f_i(\theta)q_{\backslash i}(\theta) \| f(\theta)q_{\backslash i}(\theta))$
    Inclusion: $q(\theta) \leftarrow \tilde{f}^\text{new}_i(\theta) q_{\backslash i}(\theta)$
  end for
until convergence

The EP algorithm. Some variations are possible: here we assumed that $f_0$ is in the exponential family, and we updated sequentially over $i$. The names for the steps (deletion, projection, inclusion) are not the same as in (Minka, 2001)

- Minimizes the opposite KL to variational methods
- $\tilde{f}_i(\theta)$ in exponential family $\rightarrow$ projection step is moment matching
- Loopy belief propagation and assumed density filtering are special cases
- No convergence guarantee (although convergent forms can be developed)
Conclusions

• Two examples of non-parametric Bayesian methods
  – Gaussian Processes: for regression, classification, ordinal regression, etc
  – Dirichlet Process Mixtures: for density estimation and clustering

• Both can be thought of as models with “infinitely” many parameters

• Real data require flexible models

• Many different and fast approximate inference methods are available:
  – MCMC
  – Variational methods
  – EP
  – Hierarchical clustering