Bayesian paradigm

• Consistent use of probability theory for representing unknowns (parameters, latent variables, missing data)
Bayesian paradigm

• Bayesian posterior distribution summarizes what we’ve learned from training data and prior knowledge
• Can use posterior to:
  – Describe training data
  – Make predictions on test data
  – Incorporate new data (online learning)
• Today’s question: How to efficiently represent and compute posteriors?
Factor graphs

• Shows how a function of several variables can be factored into a product of simpler functions
• \( f(x,y,z) = (x+y)(y+z)(x+z) \)
• Very useful for representing posteriors
Example factor graph

\[ p(x_i \mid m) = N(x_i; m, 1) \]
Two tasks

• Modeling
  – What graph should I use for this data?

• Inference
  – Given the graph and data, what is the mean of $x$ (for example)?
  – Algorithms:
    • Sampling
    • Variable elimination
    • Message-passing (Expectation Propagation, Variational Bayes, …)
A (seemingly) intractable problem
Clutter problem

- Want to estimate $x$ given multiple $y$’s

\[
p(x) = \mathcal{N}(x; 0, 100)
\]

\[
p(y_i|x) = (0.5)\mathcal{N}(y_i; x, 1) + (0.5)\mathcal{N}(y_i; 0, 10)
\]
Exact posterior
Representing posterior distributions

**Sampling**
- Good for complex, multi-modal distributions
- Slow, but predictable accuracy

**Deterministic approximation**
- Good for simple, smooth distributions
- Fast, but unpredictable accuracy
Deterministic approximation

Laplace’s method
- Bayesian curve fitting, neural networks (MacKay)
- Bayesian PCA (Minka)

Variational bounds
- Bayesian mixture of experts (Waterhouse)
- Mixtures of PCA (Tipping, Bishop)
- Factorial/coupled Markov models (Ghahramani, Jordan, Williams)
Moment matching

Another way to perform deterministic approximation
• Much higher accuracy on some problems

Assumed-density filtering

Loopy belief propagation

Expectation Propagation
Today

• Moment matching
  (Expectation Propagation)

Tomorrow

• Variational bounds
  (Variational Message Passing)
Best Gaussian by moment matching
Strategy

• Approximate each factor by a Gaussian in x

\[ p(y_i | x) = (0.5) \mathcal{N}(y_i; x, 1) + (0.5) \mathcal{N}(y_i; 0, 10) \]

\[ \approx \mathcal{N}(x; m_i, v_i) \]
Approximating a single factor
(naïve) \[ f_i(x) \]

\[ \times \]

\[ q^i(x) \]

\[ = \]

\[ p(x) \]
(informed) \( f_i(x) \times q_i(x) = p(x) \)
Single factor with Gaussian context
Gaussian multiplication formula

\[ \mathcal{N}(x; m_1, v_1) \mathcal{N}(x; m_2, v_2) = \mathcal{N}(m_1; m_2, v_1 + v_2) \mathcal{N}(x; m, v) \]

where \( v = \frac{1}{\frac{1}{v_1} + \frac{1}{v_2}} \)

\[ m = v \left( \frac{m_1}{v_1} + \frac{m_2}{v_2} \right) \]

\[ \mathcal{N}(x; m_1, v_1)/\mathcal{N}(x; m_2, v_2) = \frac{v_2 \mathcal{N}(x; m, v)}{(v_2 - v_1) \mathcal{N}(m_1; m_2, v_2 - v_1)} \]

where \( v = \frac{1}{\frac{1}{v_1} - \frac{1}{v_2}} \)

\[ m = v \left( \frac{m_1}{v_1} - \frac{m_2}{v_2} \right) \]
Approximation with narrow context
Approximation with medium context
Approximation with wide context
Two factors

\[ f_1(x) \xrightarrow{x} f_2(x) \]

\[ \tilde{f}_1(x) = \frac{\text{proj}[f_1(x) \tilde{f}_2(x)]}{\tilde{f}_2(x)} \]

\[ \tilde{f}_2(x) = \frac{\text{proj}[f_2(x) \tilde{f}_1(x)]}{\tilde{f}_1(x)} \]

Message passing
Three factors

\[ f_1(x) \quad \square \quad x \quad \square \quad f_2(x) \]

\[ f_3(x) \]

\[ \tilde{f}_1(x) \quad \square \quad x \quad \square \quad \tilde{f}_2(x) \]

\[ \tilde{f}_3(x) \]

Message passing
Message Passing = Distributed Optimization

- Messages represent a simpler distribution \( q(x) \) that approximates \( p(x) \)
  - A distributed representation
- Message passing = optimizing \( q \) to fit \( p \)
  - \( q \) stands in for \( p \) when answering queries
- Choices:
  - What type of distribution to construct (approximating family)
  - What cost to minimize (divergence measure)
Distributed divergence minimization

• Write \( p \) as product of factors:
  \[
p(x) = \prod_a f_a(x)
  \]

• Approximate factors one by one:
  \[
f_a(x) \rightarrow \tilde{f}_a(x)
  \]

• Multiply to get the approximation:
  \[
  q(x) = \prod_a \tilde{f}_a(x)
  \]
Global divergence to local divergence

- Global divergence:
  \[
  D(p(x) \parallel q(x)) = \]
  \[
  D(f_a(x) \prod_{b \neq a} f_b(x) \parallel \tilde{f}_a(x) \prod_{b \neq a} \tilde{f}_b(x))
  \]

- Local divergence:
  \[
  D(f_a(x) \prod_{b \neq a} \tilde{f}_b(x) \parallel \tilde{f}_a(x) \prod_{b \neq a} \tilde{f}_b(x))
  \]
Message passing

- Messages are passed between factors
- Messages are factor approximations: $\tilde{f}_a(x)$
- Factor $a$ receives $\tilde{f}_b(x), b \neq a$
  - Minimize local divergence to get $\tilde{f}_a(x)$
  - Send to other factors
  - Repeat until convergence
Gaussian found by EP
Other methods

\[ p(x, D) \]

\[ \text{vb} \]
\[ \text{laplace} \]
\[ \text{exact} \]
Accuracy

Posterior mean:
  exact = 1.64864
  ep = 1.64514
  laplace = 1.61946
  vb = 1.61834

Posterior variance:
  exact = 0.359673
  ep = 0.311474
  laplace = 0.234616
  vb = 0.171155
Cost vs. accuracy

Deterministic methods improve with more data (posterior is more Gaussian)
Sampling methods do not
Time series problems
Example: Tracking

Guess the position of an object given noisy measurements
Factor graph

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]

\[ y_1 \quad y_2 \quad y_3 \quad y_4 \]

\[ \begin{align*}
\text{e.g.} & \quad x_t &= x_{t-1} + v_t \quad \text{(random walk)} \\
& \quad y_t &= x_t + \text{noise} \\
\end{align*} \]

want distribution of x’s given y’s
Approximate factor graph

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \]
Splitting a pairwise factor

\[ x_1 \longrightarrow \Box \longrightarrow x_2 \]

\[ x_1 \quad \Box \quad \Box \longrightarrow x_2 \]
Splitting in context

$\begin{align*}
x_2 & \quad \quad \quad \quad \quad \quad x_3 \\
\downarrow & \quad \quad \quad \quad \quad \quad \downarrow \\
\quad & \quad \quad \quad \quad \quad \quad \quad \\
\quad & \quad \quad \quad \quad \quad \quad \\
\end{align*}$
Sweeping through the graph
Sweeping through the graph
Sweeping through the graph

\[ x_1 - x_2 - x_3 - x_4 \]
Sweeping through the graph
Example: Poisson tracking

- $y_t$ is a Poisson-distributed integer with mean $\exp(x_t)$
Poisson tracking model

\[ p(x_1) \sim N(0,100) \]

\[ p(x_t | x_{t-1}) \sim N(x_{t-1}, 0.01) \]

\[ p(y_t | x_t) = \exp(y_t x_t - e^{x_t}) / y_t! \]
Factor graph

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \]

\[ y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow y_4 \]

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \]
Approximating a measurement factor
Posterior for the last state
EP for signal detection

(Qi and Minka, 2003)

- Wireless communication problem
- Transmitted signal = \( a \sin(\omega t + \phi) \)
- \((a, \phi)\) vary to encode each symbol
- In complex numbers: \( ae^{i\phi} \)
Binary symbols, Gaussian noise

- Symbols are $s^1 = 1$ and $s^0 = -1$ (in complex plane)
- Received signal $y_t = a \sin(\omega t + \phi) + \text{noise}$
- Optimal detection is easy in this case
Fading channel

- Channel systematically changes amplitude and phase:
  \[ y_t = x_t s_t + \text{noise} \]

- \( s_t \) = transmitted symbol
- \( x_t \) = channel multiplier (complex number)
- \( x_t \) changes over time
Differential detection

- Use last measurement to estimate state:
  \[ x_t \approx \frac{y_{t-1}}{s_{t-1}} \]
- State estimate is noisy – can we do better?
Symbols can also be correlated (e.g. error-correcting code)
Channel dynamics are learned from training data (all 1’s)
(a) Exact posterior $p(s_1, s_2, s_3, x_1, x_2, x_3)$
(b) Approximate posterior $\prod_i q(s_i)q(x_i)$
Splitting a transition factor
Splitting a measurement factor
On-line implementation

- Iterate over the last $\delta$ measurements
- Previous measurements act as prior

- Results comparable to particle filtering, but much faster
Classification problems
Spam filtering by linear separation

Choose a boundary that will generalize to new data
Linear separation

Minimum training error solution (Perceptron)

Too arbitrary – won’t generalize well
Linear separation

Maximum-margin solution (SVM)

Ignores information in the vertical direction
Linear separation

Bayesian solution (via averaging)

Has a margin, and uses information in all dimensions
Geometry of linear separation

Separator is any vector $w$ such that:

$$w^T x_i > 0 \quad \text{(class 1)}$$
$$w^T x_i < 0 \quad \text{(class 2)}$$
$$\|w\| = 1 \quad \text{(sphere)}$$

This set has an unusual shape

SVM: Optimize over it
Bayes: Average over it
Factor graph

\[ p(y_i = \pm 1 \mid x_i, w) = I(y_i x_i^T w > 0) \]
Performance on linear separation

EP Gaussian approximation to posterior
A typical run on the 3-point problem

Error = distance to true mean of w

Billiard = Monte Carlo sampling (Herbrich et al, 2001)

Opper&Winther’s algorithms:
MF = mean-field theory
TAP = cavity method (equiv to Gaussian EP for this problem)
Gaussian kernels

• Map data into high-dimensional space so that

\[ \phi(x_i)^T \phi(x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]
Bayesian model comparison

- Multiple models $M_i$ with prior probabilities $p(M_i)$
- Posterior probabilities:
  \[ p(M_i|D) \propto p(D|M_i)p(M_i) \]
- For equal priors, models are compared using model evidence:
  \[ p(D|M_i) = \int_\theta p(D, \theta|M_i) d\theta \]
Highest-probability kernel
Margin-maximizing kernel
Bayesian feature selection

Synthetic data where 6 features are relevant (out of 20)

Bayes picks 6

Margin picks 13
EP versus Monte Carlo

• Monte Carlo is general but expensive
  – A sledgehammer
• EP exploits underlying simplicity of the problem (if it exists)
• Monte Carlo is still needed for complex problems (e.g. large isolated peaks)
• Trick is to know what problem you have
Software for EP

• Bayes Point Machine toolbox
  http://research.microsoft.com/~minka/papers/ep/bpm/

• Sparse Online Gaussian Process toolbox
  http://www.kyb.tuebingen.mpg.de/bs/people/csatoI/ogp/index.html

• Infer.NET
  http://research.microsoft.com/infernet
Further reading

• EP bibliography

• EP quick reference
Tomorrow

- Variational Message Passing
- Divergence measures
- Comparisons to EP