Overview

- Part 1: Theory
  - Graphical Models
  - Inference in Factor Graphs
  - Approximate Message Passing
  - Distributed Message Passing
- Part 2: Applications
  - TrueSkill: Gamer Rating and Matchmaking
  - TrueSkill Through Time: History of Chess
  - Click-Through Rate Prediction in Online Advertising
  - Matchbox: Recommendation Systems
Overview

- **Graphical Models**
- Inference in Factor Graphs
- Approximate Message Passing
- Distributed Message Passing
**Probabilities and Beliefs**

- **Design:** System must assign degree of plausibility $P(A)$ to each logical statement $A$.

- **Axiom:**
  1. $P(A)$ is a real number
  2. $P(A)$ is independent of Boolean rewrite
  3. $P(A|C') > P(A|C)$ and $P(B|AC') = P(B|AC)$ then $P(AB|C') \geq P(AB|C)$

$P$ must be a probability measure!

**Infer-Predict-Decide Cycle**

**Decision Making:**
Loss($\text{Action}, \text{Data}$) + $P(\text{Data})$ \rightarrow $\text{Action}$
- Business-loss not learning-loss!
- Often involves optimization!

**Inference:**
P($\text{Parameters}$) + $P(\text{Data})$ \rightarrow $P(\text{Parameters} | \text{Data})$
- Requires a (structural) model $P(\text{Data} | \text{Parameters})$
- Allows to incorporate prior information $P(\text{Parameters} | \text{Data})$

**Prediction:**
P($\text{Parameters}$) + $\text{Data}$ \rightarrow $P(\text{Data})$
- Requires integration/summation of parameter uncertainty
- Does not change state!
Graphical Models

• **Definition:** Graphical representation of joint probability distribution
  - Nodes: □ = Variables
  - Edges: Relationship between variables

• **Variables:**
  - Observed Variables: Data
  - Unobserved Variables: ‘Causes’ + Temporary/Latent

• **Key Questions:**
  - **(Conditional) Dependency:** $p(a, b|c) \neq p(a|c) \cdot p(b|c)$
  - Inference/Marginalisation: $p(a, b) = \sum_c p(a, b, c)$

Directed Models: Bayesian Networks

• **Definition:** Graphical representation of joint probability distribution (Pearl, 1988)
  - Nodes: □ = Variables
  - Directed Edges: Conditional probability distribution

• **Semantic:**
  - Ancestral relationship of dependency
    $p(x) = \prod_i p(x_i|\text{Parents}(i))$
    $p(a, b, c) = p(a) \cdot p(b) \cdot p(c|a, b)$
**Undirected Models: Markov Networks**

- **Definition**: Graphical representation of joint probability distribution (Pearl, 1988)
  - Nodes: \( \bigcirc \) = Variables
  - Edges: Dependency between variables

- **Semantic**:
  
  \[
P(x) = \frac{1}{Z} \prod_c \phi(x_c) \quad \phi \geq 0
  \]
  
  - Local potentials over cliques
    
    \[
p(a, b, c) = \frac{1}{Z} \cdot \phi_{ab}(a, c) \cdot \phi_{bc}(b, c)
    \]
    
    \[
    Z = \sum_a \sum_b \sum_c \phi_{ab}(a, c) \cdot \phi_{bc}(b, c)
    \]

**Factor Graphs**

- **Definition**: Graphical representation of product structure of a function (Wiberg, 1996)
  - Nodes: \( \square \) = Factors \( \bigcirc \) = Variables
  - Edges: Dependencies of factors on variables.

- **Semantic**:
  
  \[
P(x) = \prod_f f(x_V(f))
  \]
  
  - Local variable dependency of factors
    
    \[
p(a, b, c) = f_1(a) \cdot f_2(b) \cdot f_3(a, b, c)
    \]
Factor Graphs are Powerful!

Undirected graphical models can hide the factorisation within a clique!

Factor Graphs and Bayes’ Law

- Bayes’ law
  \[ p(s|y) \propto p(y|s) \cdot p(s) \]

- Factorising prior
  \[ p(s) = p(s_1) \cdot p(s_2) \]

- Factorising likelihood
  \[ p(y, t, d|s) = \prod_t p(t_1|s_1) \cdot p(d|t_1, t_2) \cdot p(y|d) \]

- Inference: Sum out latent variables
  \[ p(y|s) = \sum_t \sum_d p(y, t, d|s) \]
## Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>Dependency</th>
<th>Efficient Inference</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian Networks</td>
<td>Yes</td>
<td>Somewhat</td>
<td>Ancestral Generative Process</td>
</tr>
<tr>
<td>Markov Networks</td>
<td>Yes</td>
<td>No</td>
<td>Local Couplings and Potentials</td>
</tr>
<tr>
<td>Factor Graphs</td>
<td>No</td>
<td>Yes</td>
<td>Efficient, distributed inference</td>
</tr>
</tbody>
</table>

## Overview

- Graphical Models
- Inference in Factor Graphs
- Approximate Message Passing
- Distributed Message Passing
Factor Graphs and Factor Trees

- **Factor Graphs**: Arbitrary functions
  - Bayesian Networks
  - Markov Networks

- **Factor Trees**: Functions where the variable indices never decrease from left to right

- **Factor Graph ➔ Factor Tree**:
  1. Pick an arbitrary node
  2. Build the spanning tree

Observation: Sum of products becomes product of sums of all messages from neighbouring factors to variable!
Messages: From Factors To Variables

Observation: Factors only need to sum out all their local variables!

Messages: From Variables To Factors

Observation: Variables pass on the product of all incoming messages!
The Sum-Product Algorithm

- Three update equations (Aji & McEliece, 1997)
  \[ p(t) = \prod_{f \in F} m_{f \rightarrow t}(t) \]
  \[ m_{f \rightarrow t_1}(t_1) = \sum_{t_2} \cdots \sum_{t_n} f(t_1, t_2, t_3, \ldots) \prod_{i > 1} m_{t_i \rightarrow f}(t_i) \]
  \[ m_{t \rightarrow f}(t) = \prod_{f \notin F \setminus \{f\}} m_{f \rightarrow t}(t) \]

- Update equations can be directly derived from the distributive law.
- Calculate all marginals at the same time!
- Only need to pass messages twice along each edge.

Practical Considerations I

- Log-Transform:
  \[ \lambda_{f \rightarrow t}(t) := \log[m_{f \rightarrow t}(t)] \]
  \[ \log[p(t)] = \sum_{f \in F} \lambda_{f \rightarrow t}(t) \]
  \[ \lambda_{f \rightarrow t_1}(t_1) = \sum_{t_2} \cdots \sum_{t_n} f(t_1, t_2, t_3, \ldots) \exp\left[\sum_{i > 1} \lambda_{t_i \rightarrow f}(t_i)\right] \]
  \[ \lambda_{t \rightarrow f}(t) = \sum_{f \notin F \setminus \{f\}} \lambda_{f \rightarrow t}(t) \]

- Exponential Family Messages:
  \[ m(t) \propto \exp(\psi(t) \cdot \theta) \]

- Message updates are just additions of the parameters \( \theta \).
Exponential Families

- (Univariate) Gaussian: \( \theta := \frac{\mu}{\sigma^2}, \frac{1}{\sigma^2} \)
- Bernoulli: \( \theta := \log \left( \frac{p}{1-p} \right) \)
- Binomial: \( \theta := \log \left( \frac{p}{1-p} \right) \)
- Beta: \( \theta := (\alpha, \beta) \)
- Gamma: \( \theta := (\alpha, \frac{1}{\beta}) \)

Practical Considerations II

- Redundant computations:
  \[
p(t) = \prod_{f \in \mathcal{F}} m_{f \rightarrow t}(t)\]
  \[
m_{t \rightarrow f}(t) = \prod_{f_j \in \mathcal{F} \setminus \{f\}} m_{f_j \rightarrow t}(t)\]
  \[
p(t) = m_{t \rightarrow f}(t) \cdot m_{f \rightarrow t}(t)\]

- Caching: Only store \( p(t) \) and \( m_{f \rightarrow t}(t) \), then
  \[
m_{t \rightarrow f}(t) = \frac{p(t)}{m_{f \rightarrow t}(t)}\]
Overview

- Graphical Models
- Inference in Factor Graphs
- **Approximate Message Passing**
- Distributed Message Passing

Approximate Message Passing

- **Problem:** The exact messages from factors to variables may not be closed under products.

- **Solution:** Approximate each marginal as well as possible in using a divergence measure on beliefs.

- **General Idea:** Leave-one out approximation

\[
\bar{\rho}(t) = \arg\min_{\hat{\rho}} D \left[ m_{f \rightarrow t} \cdot \bar{m}_{t \rightarrow f} \| \hat{\rho} \right]
\]

\[
\bar{m}_{f \rightarrow t}(t) = \frac{\bar{\rho}(t)}{\bar{m}_{t \rightarrow f}(t)}
\]
Approximate Message Passing

\[ \begin{align*}
\hat{m}_{t \rightarrow f}(t) & \ast \hat{m}_{f \rightarrow u}(t) = m_{f \rightarrow u}(t) \\
\hat{m}_{t \rightarrow f}(t) & \ast ? = p(t)
\end{align*} \]

Divergence Measures

- **Kullback-Leibler Divergence**: Expected log-odd ratio between two distributions:
  \[ \text{KL}(p, q) := \sum_t p(t) \log \left( \frac{p(t)}{q(t)} \right) \]

- **Minimizer for Exponential Families**: Matching the moments of the distribution \( p(t) \):

- **General \( \alpha \)-Divergence**:
  \[ D_\alpha(p, q) := \frac{1 - \sum_t p^{\alpha-1}(t)}{\alpha(1 - \alpha)} \]

- **Special Cases**:
  \[ D_0(p, q) = \text{KL}(q, p) \]
  \[ D_1(p, q) = \text{KL}(p, q) \]
α-Divergence in Pictures

Overview

- Graphical Models
- Inference in Factor Graphs
- Approximate Message Passing
- Distributed Message Passing
Large-Data Challenge

- **Large Data (e.g. Facebook user actions)**
  - 500m daily users
  - 3 bln daily likes & comments

- **Two types of variables**
  - Observed \( \Rightarrow \) Data Factors
  - Latent \( \Rightarrow \) Model parameters

- **Discriminative Models**
  - Given the model parameters, data variables are independent

\[
p(\theta | X, Y) \propto \prod_i p(y_i | \theta, x_i) \cdot \prod_j p(\theta_j)
\]

Distributed Message Passing

- **Idea:** Group variables and send messages across system boundaries

\[
\prod_i p(y_i | \theta, x_i) \cdot p(\theta) = \prod_k p(y_k | \theta, x_k) \cdot \prod_l p(\theta_l)
\]

- **Data factors:** \( f_k(X_k, Y_k, \theta) \)
  - Know exactly which model parameter messages get updated

- **Parameter factors:** \( g_l(\theta_l) \)
  - Need to keep track of which data factors need message update
Distributed Conditional Models

Belief Store ("Memory")
Message Passing ("Communicate")
Data Messages ("Compute")

A Systems Service View

Compute | Communicate | Store
--- | --- | ---
Train Request | Train Request | Train Request
Predict Request | Predict Request | Predict Request
Relation to Map-Reduce

- **Map-Reduce**
  - **Map**: Data nodes compute messages $m_{F_k \rightarrow \mu}$ from data $y_i$ and $m_{\mu \rightarrow F_k}$
  - **Reduce**: Combine messages $m_{F_k \rightarrow \mu}$ into $p_{\mu}$ by multiplication
  - Vanilla MR is a single pass only!

- **Caveats**:
  - Approximate data factors need all incoming message $m_{F_k \rightarrow \mu}$!
  - Each machine needs to be able to store the belief over $\mu$

---

Approximation Quality

$$
P(y_i|\theta, x_i) = \Phi(y_i \theta^T x_i)$$

$$
P(\theta) = \prod_j \mathcal{N}(\theta_j; \mu_j, \sigma_j^2)$$

Sequential | Parallel
Approximation Quality

\[ \mathbf{x} = [1; 1; \ldots; 1]^T \]

Single Bias Feature

100 Bias Features

Solution: Dampening!

\[ \lambda_f \rightarrow \theta \Rightarrow \alpha \cdot \lambda_f \rightarrow \theta \]

First Step

Second Step
Break!

Part 2: Applications
Overview

- TrueSkill: Gamer Rating and Matchmaking
- Click-Through Rate Prediction in Online Advertising
- Matchbox: Recommendation Systems

TrueSkill™

Joint work with Thore Graepel, Tom Minka & Phillip Trelford
• Competition is central to our lives
  – Innate biological trait
  – Driving principle of many sports
• Chess Rating for fair competition
  – ELO: Developed in 1960 by Árpád Imre Élő
  – Matchmaking system for tournaments
• Challenges of online gaming
  – Learn from few match outcomes efficiently
  – Support multiple teams and multiple players per team

The Skill Rating Problem

• Given:
  – Match outcomes: Orderings among \( k \) teams

• Questions:
  – Skill ratings for each player such that
    – Global ranking among all players
    – Fair matches between teams of players

The Skill Rating Problem
Two Player Match Outcome Model

- Latent Gaussian performance model for fixed skills
- Possible outcomes: Player 1 wins over 2 (and vice versa)

Two Team Match Outcome Model

- Skill of a team is the sum of the skills of its members

\[ P(y_{12} = (1, 2)|p_1, p_2) = \mathbb{I}(p_1 > p_2) \]

\[ P(t_{1|s_1, s_2}) = \mathcal{N}(t_1; s_1 + s_2, 2 \cdot \beta^2) \]
• Possible outcomes: Permutations of the teams

\[ P(\mathbf{y}|t_1, t_2, t_3) = \mathbb{I}(\mathbf{y} = (i, j, k)) \text{ where } t_i > t_j > t_k \]

• But we are interested in the (Gaussian) posterior!

\[ P(s_i|\mathbf{y} = (1, 2, 3)) = \mathcal{N}(s_i; \mu_i, \sigma_i^2) \]
Efficient Approximate Inference

Gaussian Prior Factors

Fast and efficient approximate message passing using Expectation Propagation

Ranking Likelihood Factors

Applications to Online Gaming

- **Leaderboard**
  - Global ranking of all players
  \[ \mu_i = 3 \cdot \sigma_i \]

- **Matchmaking**
  - For gamers: Most uncertain outcome
  - For inference: Most informative
  - Both are equivalent!

**Leaderboard**

<table>
<thead>
<tr>
<th>Level</th>
<th>Gamertag</th>
<th>Avg. Life</th>
<th>Best Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Shukki</td>
<td>00:00:40</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>1st</td>
<td>StripedEye</td>
<td>00:00:41</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>1st</td>
<td>DrsThrapisi</td>
<td>00:01:07</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>1st</td>
<td>Sazidemo</td>
<td>00:00:59</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

**Matchmaking**

- Matchmaking for gamers:
  - Most uncertain outcome

- Matchmaking for inference:
  - Most informative

\[ P(y_i) \approx P(p_i | \mu_i, \sigma^2) = 0, \sigma^2 + \sigma^2_i = 0 \]
Experimental Setup

- **Data Set: Halo 2 Beta**
  - 3 game modes
    - Free-for-All
    - Two Teams
    - 1 vs. 1
  - > 60,000 match outcomes
  - ≈ 6,000 players
  - 6 weeks of game play
  - Publically available

---

Convergence Speed

![Diagram showing convergence speed over the number of games played, comparing different players and ranks.](image)
**Convergence Speed (ctd.)**

![Graph showing convergence speed for different scenarios: char wins, SQLWildman wins, and both players draw.](image)

**Number of games played**

- **Winning probability**
  - 0% to 100%
  - 0 to 500 games

**Game scenarios**

- **char wins**
- **SQLWildman wins**
- **Both players draw**

**Xbox 360 & Halo 3**

- **Xbox 360 Live**
  - Launched in September 2005
  - Every game uses TrueSkill™ to match players
  - > 10 million players
  - > 2 million matches per day
  - > 2 billion hours of gameplay

- **Halo 3**
  - Launched on 25th September 2007
  - Largest entertainment launch in history
  - > 200,000 player concurrently (peak: 1,000,000)
Halo 3 in Action

Halo 3 Public Beta Analysis

1 games played
Skill Distributions of Online Games

- **Golf (18 holes):** 60 levels.
- **Car racing (3-4 laps):** 40 levels.
- **UNO (chance game):** 10 levels.

TrueSkill™ Through Time: Chess

- Model time-series of skills by smoothing across time
- History of Chess
  - 3.5M game outcomes (ChessBase)
  - 20 million variables (each of 200,000 players in each year of lifetime + latent variables)
  - 40 million factors
Why Predict Probability-of-Click?

The Scale of Things

• Several weeks of data in training: 7,000,000,000 impressions

• 2 weeks of CPU time during training: 2 wks × 7 days × 86,400 sec/day = 1,209,600 seconds

• Learning algorithm speed requirement:
  - 5,787 impression updates / sec
  - 172.8 μs per impression update

Advantages of improved probability estimates:

– Increase user satisfaction by better targeting
– Fairer charges to advertisers
– Increase revenue by showing ads with high click-through rate
The Flow of Information

- Why structured data?
  - Data validation and cleaning
  - Principled feature transformations

Uncertainty: Bayesian Probabilities

Client IP
102.34.12.201
15.70.165.9
221.98.2.187
92.154.3.86

Match Type
Exact Match
Broad Match

Position
ML-1
SB-1
SB-2

\( p(p_{\text{Click}}) \)
Principled Exploration

Training Algorithm in Action
Inference: An Optimization View

\[ \mu_i \leftarrow \mu_i + \frac{\sigma_i^2}{s} \cdot h \left( \frac{\sum_{j=1}^{d} \mu_j}{s} \right) \]

\[ \sigma_i^2 \leftarrow \sigma_i^2 \left( 1 - \frac{\sigma_i^2}{s^2} \cdot g \left( \frac{\sum_{j=1}^{d} \mu_j}{s} \right) \right) \]

\[ s^2 = \beta^2 + \sum_{j=1}^{d} \sigma_j^2 \]

\( h(t) = \mathcal{N}(t; 0, 1) / \Phi(t) \)

\( g(t) = h(t) \cdot [h(t) + t] \)

Client IP: Mean & Variance

ClientIP Parameters

Low clickers

High clickers
UserAgent: Mean Posterior Effects

Accuracy

Empirical CTR

Predicted CTR
MatchBox

Joint work with Thore Graepel, Joaquin Quillenro Candela, David Stern.
**Matchbox With Metadata**

User: \[ s = Ux \]

Item: \[ t = Vy \]

Rating potential: \[ \mathcal{N}(s^T t, \beta^2) \]

**Recommender System: MatchBox**

User:
- mark
- sheryl
- tao
- ralf

Gender:
- Male
- Female

Movie:
- Social Network
- Heat
- The Rock
- The Godfather

Director:
- R. Scott
- C. Eastwood
- Q. Tarantino
- R. Howard
Message Passing For Matchbox

User/Item Trait Space

'Preference Cone' for user 145035
Incremental Training with ADF

ADF: Message Passing Iteration 1
Message Passing Iteration 4

feedback models
Feedback Models

Message Passing: Compositionality

User Model

Item Model

Context Model

Feedback Model

$q > 0$

$r$

$q$

$r$

$q$

$x_1 x_2 x_3 x_4$

$q$

$q > 0$

$s_1 u_{11} u_{12} u_{13} u_{14}$

$s_2 v_{11} v_{12} v_{13} v_{14}$

$t_1$

$t_2$

$u_{21} u_{22} u_{23} u_{24}$

$v_{21} v_{22} v_{23} v_{24}$
accuracy

Performance and Accuracy

MovieLens Data
- 1 million ratings
- 3,900 movies / 6,040 users
- User / movie metadata
MovieLens – 1,000,000 ratings

6,040 users

<table>
<thead>
<tr>
<th>User ID</th>
<th>User Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>&lt;18</td>
</tr>
<tr>
<td>Academic</td>
<td>18-25</td>
</tr>
<tr>
<td>Artist</td>
<td>25-34</td>
</tr>
<tr>
<td>Admin</td>
<td>35-44</td>
</tr>
<tr>
<td>Student</td>
<td>45-49</td>
</tr>
<tr>
<td>Customer</td>
<td>50-55</td>
</tr>
<tr>
<td>Health Care</td>
<td>50-55</td>
</tr>
<tr>
<td>Managerial</td>
<td>Male</td>
</tr>
<tr>
<td>Farmer</td>
<td>Unemployed</td>
</tr>
<tr>
<td>Homemaker</td>
<td>Writer</td>
</tr>
</tbody>
</table>

3,900 movies

<table>
<thead>
<tr>
<th>Movie ID</th>
<th>Movie Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Horror</td>
</tr>
<tr>
<td>Adventure</td>
<td>Musical</td>
</tr>
<tr>
<td>Animation</td>
<td>Mystery</td>
</tr>
<tr>
<td>Children’s</td>
<td>Romance</td>
</tr>
<tr>
<td>Comedy</td>
<td>Thriller</td>
</tr>
<tr>
<td>Crime</td>
<td>Sci-Fi</td>
</tr>
<tr>
<td>Documentary</td>
<td>War</td>
</tr>
<tr>
<td>Drama</td>
<td>Western</td>
</tr>
<tr>
<td>Fantasy</td>
<td>Film Noir</td>
</tr>
</tbody>
</table>

MovieLens with Thresholds Model

(ADF), Training Time= 1 Minute

Mean Absolute Error

- MetaData Off
- MetaData On

Lam et al.
MovieLens Error with Thresholds

![Graph showing Mean Absolute Error vs EP Iterations with different thresholds (K=0, K=2, K=5, K=10, K=20) and Lam et al. comparison.]

Recommendation Speed
**Recommendation Speed**

- **Goal:**
  find N items with highest predicted rating.

- **Challenge:**
  potentially have to consider all items.

- **Two approaches to make this faster:**
  - Locality Sensitive Hashing
  - KD Trees

- **Locality Sensitive Hash:**
  \[ P(h(x) = h(y)) = \text{sim}(x, y) \]

---

**Random Projection Hashing**

- **Random Projections:**
  - Generate random hyper planes
    (m random vectors, \(a_i\)).
  - Gives m bit hash, \(\{x_0, x_1, \ldots, x_m\}\), by:
    \[ x_i = 1[a_i \cdot t > 0] \]

- p(all bits match) \(\propto\) cosine similarity.

- Store items in buckets indexed by keys.

- Given a user trait vector:
  1. Generate key, q.
  2. Search buckets by hamming distance from q until find N items.
Accuracy and Speedup

![Graphs showing predicted rating and cost per recommendation vs. hash key bits.]

Thanks!