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
  – Pattern Learning in Go
• Graphical Models
• Inference in Factor Graphs
• Approximate Message Passing
• Distributed Message Passing
Cox Axioms: Probabilities and Beliefs

• **Design:** System must assign degree of plausability $p(A)$ to each logical statement $A$.

• **Axiom:**

  1. $p(A)$ is a real number
  2. $p(A)$ is independent of Boolean rewrite
  3. $p(A|C') > p(A|C)$ \land \ p(B|AC') = p(B|AC) \Rightarrow \ p(AB|C') \geq P(AB|C')$

**P must be a probability measure!**
**Infer-Predict-Decide Cycle**

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

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

**Decision Making:**
\[ \text{Loss(Action,Data)} + P(\text{Data}) \rightarrow \text{Action} \]
- Business-loss not learning-loss!
- Often involves optimization!
• **Definition:** Graphical representation of joint probability distribution
  – Nodes: $\bigcirc = \text{Variables}$
  – Edges: Relationship between variables

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

• **Key Questions:**
  – (Conditional) **Dependency:** $p(a,b|c) \overset{?}{=} p(a|c) \cdot p(b|c)$
  – **Inference/Marginalisation:** $p(a,b) = \sum_c p(a,b,c)$
• **Definition:** Graphical representation of joint probability distribution (Pearl, 1988)
  - Nodes: \( \bigcirc \) = Variables
  - Directed Edges: Conditional probability distribution

• **Semantic:**

\[
p(x) = \prod_{i} p(x_i | x_{\text{parents}(i)})
\]

- Ancestral relationship of dependency

\[
p(a, b, c) = p(a) \cdot p(b) \cdot p(c | a, b)
\]
**Definition:** Graphical representation of joint probability distribution (Pearl, 1988)
- Nodes: $\bigcirc = \text{Variables}$
- Edges: Dependency between variables

**Semantic:**
\[
p(x) = \frac{1}{Z} \cdot \prod_c \phi(x_c) \quad \phi \geq 0
\]
- Local potentials over cliques

\[
p(a, b, c) = \frac{1}{Z} \cdot \phi_{ac}(a, c) \cdot \phi_{bc}(b, c)
\]
\[
Z = \sum_a \sum_b \sum_c \phi_{ac}(a, c) \cdot \phi_{bc}(b, c)
\]
Definition: Graphical representation of product structure of a function (Wiberg, 1996)
- Nodes: □ = Factors  ○ = 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!

\[ f_1(a, b, c) \quad f_1(a, b) \cdot f_2(b, c) \cdot f_3(a, c) \quad \phi(a, b, c) \]
• 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
\[ p(y, t, d|s) = \prod_i p(t_i|s_i) \cdot p(d|t_1, t_2) \cdot p(y|d) \]

• Inference: Sum out latent variables
\[ p(y|s) = \sum_t \sum_d p(y, t, d|s) \]
<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>
• Graphical Models
• Inference in Factor Graphs
• Approximate Message Passing
• Distributed Message Passing
Factor Trees: Separation

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

• Three update equations (Aji & McEliece, 1997)

\[ p(t) = \prod_{f \in F_t} m_{f \rightarrow t}(t) \]

\[ m_{f \rightarrow t_1}(t_1) = \sum_{t_2} \sum_{t_3} \cdots \sum_{t_n} f(t_1, t_2, t_3, \ldots) \prod_{i \geq 1} m_{t_i \rightarrow f}(t_i) \]

\[ m_{t \rightarrow f}(t) = \prod_{f_j \in F_t \setminus \{f\}} m_{f_j \rightarrow t}(t) \]

• 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:**
  \[ \lambda_{f \rightarrow t}(t) := \log \left[ m_{f \rightarrow t}(t) \right] \]
  \[ \log [p(t)] = \sum_{f \in F_t} \lambda_{f \rightarrow t}(t) \]
  \[ \lambda_{f \rightarrow t_1}(t_1) = \sum_{t_2} \sum_{t_3} \cdots \sum_{t_n} f(t_1, t_2, t_3, \ldots) \exp \left[ \sum_{i>1} \lambda_{t_i \rightarrow f}(t_i) \right] \]
  \[ \lambda_{t \rightarrow f}(t) = \sum_{f_j \in F_t \setminus \{f\}} \lambda_{f_j \rightarrow t}(t) \]

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

- **Message updates are just additions of the parameters** \( \theta \)!
• (Univariate) Gaussian: \( \theta := \left( \frac{\mu}{\sigma^2}, \frac{1}{\sigma^2} \right) \)

• Bernoulli: \( \theta := \log \left( \frac{p}{1-p} \right) \)

• Binomial: \( \theta := \log \left( \frac{p}{1-p} \right) \)

• Beta: \( \theta := (\alpha, \beta) \)

• Gamma: \( \theta := \left( \alpha, \frac{1}{\beta} \right) \)
Practical Considerations II

- **Redundant computations:**

\[ p(t) = \prod_{f \in F_t} m_{f \rightarrow t}(t) \]

\[ m_{t \rightarrow f}(t) = \prod_{f_j \in F_t \setminus \{f\}} m_{f_j \rightarrow t}(t) \]

\[ p(t) = m_{t \rightarrow f}(t) \cdot m_{f \rightarrow t}(t) \]

- **Caching:** Only store \( p(t) \) and \( m_{f \rightarrow t}(t) \), then

\[ m_{t \rightarrow f}(t) = \frac{p(t)}{m_{f \rightarrow t}(t)} \]
• Graphical Models
• Inference in Factor Graphs
• Approximate Message Passing
• Distributed 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

\[
\hat{p}(t) = \arg\min_{\hat{p}} \mathbb{D}[m_{f\rightarrow t} \cdot \hat{m}_{t\rightarrow f} \cdot \hat{p}] \\
\hat{m}_{f\rightarrow t}(t) = \frac{\hat{p}(t)}{\hat{m}_{t\rightarrow f}(t)}
\]
Divergence Measures

- **Kullback-Leibler Divergence**: Expected log-odd ratio between two distributions:

\[
KL(p, q) := \sum_t p(t) \log \left( \frac{p(t)}{q(t)} \right)
\]

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

- **General \( \alpha \)-Divergence**:

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

- **Special Cases**:

\[
\begin{align*}
D_0(p, q) &= KL(q, p) \\
D_1(p, q) &= KL(p, q)
\end{align*}
\]
α-Divergence in Pictures

\[ p(t) \]
\[ \arg\min_q D_0(p, q) \]
\[ \arg\min_q D_1(p, q) \]
• Graphical Models
• Inference in Factor Graphs
• Approximate Message Passing
• Distributed Message Passing
## Large-Data Challenge

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of Data Items</th>
<th>Number of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook News Feed</td>
<td>100B news stories / day</td>
<td>650M users / day</td>
</tr>
<tr>
<td>Facebook Social Graph</td>
<td>130B friends connection</td>
<td>1B users</td>
</tr>
<tr>
<td>Google PageRank</td>
<td>~4T web links</td>
<td>1T web pages</td>
</tr>
<tr>
<td>Amazon Forecasting</td>
<td>15.6M products/ day (peak)</td>
<td>20+M products</td>
</tr>
<tr>
<td>Xbox Gamer Ranking</td>
<td>&gt;1M sessions/game (peak)</td>
<td>20+M users</td>
</tr>
</tbody>
</table>

### Important Constants

- Number of seconds / day: $86,400$
- Number of RAM read access / day: $\sim 10^{13}$
- Number of RAM write access / day: $\sim 10^{12}$
- Max network bandwidth: $\sim 8$TB / day
Distributed Conditional Models

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

Belief Store ("Memory")

Message Passing ("Communicate")

Data Messages ("Compute")
Distributed Message Passing

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

\[
\prod_{i} p(y_i | \theta, x_i) \cdot p(\theta) = \prod_{k} \prod_{j=1}^{n_k} p(y_{k,j} | \theta, x_{k,j}) \cdot \prod_{l} \prod_{r=1}^{m_l} p(\theta_{l,r})
\]

- **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
A Systems Service View

Train Request

Train Request

Predict Request

Compute

Communicate

Store

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( f_k(X_{k-1}, Y_{k-1}, \theta) \)

\( \theta_{l,r}, m_{f \rightarrow \theta_{l,r}} \)

\( \theta_{l,r}, p_{\theta_{l,r}}, \Delta_{l,r} \)

\( p(\theta_{l,r}) \)

\( p(\theta_{l,r}) \)

\( p(\theta_{l,r}) \)

\( p(\theta_{l,r}) \)

\( p(\theta_{l,r}) \)

\( p(\theta_{l,r}) \)
Additional Technical Challenges

- Shard <-> Machine Consistency
- High Performance (Asynchronous programming)
- Reliability, Maintainability
  - All parameters are stored in RAM ➔ “Checkpoint” or Redundancy
  - Canary procedure is unsafe ➔ Traffic proxy
  - Central model management and model management tools
• 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

\[ p(y_i|\theta, x_i) = \Phi(y_i\theta^T x_i) \]
\[ p(\theta) = \prod_j \mathcal{N}(\theta_j; \mu_j, \sigma_j^2) \]
Approximation Quality

\[ x = [1; 1; \ldots; 1]^T \]

Single Bias Feature

100 Bias Features
Solution: Dampening!

\[ \lambda_{f \to \theta} \Rightarrow \alpha \cdot \lambda_{f \to \theta} \]

First Step

Second Step

![Graphs showing the relationship between μ/σ² and 1/σ² for Sequential and Parallel processes.](Image)
Part 2: Applications
• TrueSkill: Gamer Rating and Matchmaking
• Click-Through Rate Prediction in Online Advertising
• Matchbox: Recommendation Systems
• Pattern Learning in Go
Motivation

• 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
Given:
- Match outcomes: Orderings among $k$ teams consisting of $n_1$, $n_2$, ..., $n_k$ players, respectively

Questions:
- Skill ratings $s_i$ 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)

\[ P(y_{12} = (1, 2) | p_1, p_2) = \mathbb{I}(p_1 > p_2) \]
Skill of a team is the sum of the skills of its members.

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

\[
P(y|t_1, t_2, t_3) = \mathbb{I}(y = (i, j, k)) \text{ where } t_i > t_j > t_k
\]
But we are interested in the (Gaussian) posterior!

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

- \( s_1 \)
- \( s_2 \)
- \( s_3 \)
- \( s_4 \)
- \( t_1 \)
- \( t_2 \)
- \( t_3 \)
- \( y_{12} = (1, 2) \)
- \( y_{23} = (2, 3) \)
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

\[
P(p_i \approx p_j | \mu_i - \mu_j, \sigma_i^2 + \sigma_j^2) \]

\[
P(p_i \approx p_j | \mu_i - \mu_j = 0, \sigma_i^2 + \sigma_j^2 = 0)\]
• **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 (ctd.)

- **Winning probability**
  - Red: char wins
  - Blue: SQLWildman wins
  - Green: Both players draw

- **Number of games played**
  - x-axis: 0 to 500
  - y-axis: 0% to 100%

- 5/8 games won by char

- Graph shows the convergence speed of the winning probability over the number of games played.
• **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
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
Online Advertising

Joint work with Thore Graepel, Joaquin Quiñonero Candela, Onno Zoeter, Tom Borchert, Phillip Trelford
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

Why Predict Probability-of-Click?

\[ c_i = b_{i+1} \cdot \frac{p_{i+1}}{p_i} \]

\[ b_1 \cdot p_1 \geq b_2 \cdot p_2 \geq \ldots \]
• Several weeks of data in training: 7,000,000,000 impressions

• 2 weeks of CPU time during training: 
  
  \[2 \text{ wks} \times 7 \text{ days} \times 86,400 \text{ sec/day} = 1,209,600 \text{ seconds}\]

• Learning algorithm speed requirement: 
  – 5,787 impression updates / sec
  – 172.8 μs per impression update
The Flow of Information

- **User interaction** ➔ **Raw Logs** ➔ **Structured Data**

- **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

\( p(p_{\text{Click}}) \)
average: 25% (3 clicks out of 12 impressions)

average: 30% (30 clicks out of 100 impressions)
Training Algorithm in Action

No Click

Click

w_1 + z

w_2

Prediction

Training/Update
Inference: An Optimization View

\[ \mu_i \leftarrow \mu_i + \frac{\sigma_i^2}{s} \cdot h \]

\[ \sigma_i^2 \leftarrow \sigma_i^2 \left( 1 - \frac{\sigma_i^2}{s^2} \cdot g \right) \]

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

\[ h(t) = \frac{\mathcal{N}(t; 0, 1)}{\Phi(t)} \]

\[ g(t) = h(t) \cdot [h(t) + t] \]
Client IP: Mean & Variance

ClientIP Parameters

$\sigma^2$

$\mu$

Low clickers

High clickers
UserAgent: Mean Posterior Effects

User Agent Feature

- MetaSearch
- FileTransfer
- Bot
- FEX
- OperaNonLinux
- Linux
- FireFoxNonLinux
Accuracy
Joint work with Thore Graepel, Joaquin Quiñonero Candela, David Stern
User Metadata

Item Metadata

User  \[ s = Ux \]

Item  \[ t = Vy \]

Rating potential  \[ \sim \mathcal{N}(s^\top t, \beta^2) \]
Recommender System: MatchBox

User
- mark
- ralf
- tao
- sheryl

Gender
- Male
- Female

User likes Movie

User dislikes Movie

Social Network
- Heat
- The Rock
- The Godfather

Movie

Director
- R. Scott
- C. Eastwood
- Q. Tarantino
- R. Howard
Message Passing For Matchbox
User/Item Trait Space

-1.5 -1 -0.5 0 0.5 1 1.5

Users
Movies

24: Season 3
Adaptation
24: Season 2
A Clockwork Orange
A Knights Tale
AI: Artificial Intelligence
A Cinderella Story

‘Preference Cone’ for user 145035

-1.5 -1 -0.5 0 0.5 1 1.5
Message Passing Iteration 2

![Graph showing data points in a scatter plot with x and y axes ranging from -1.5 to 1.5. The data points are marked in red.]
feedback models
Feedback Models
Feedback Models
Feedback Models

![Feedback Model Diagram]

- r
- q
- >
- <
- t₀
- t₁
- t₂
- t₃

Rating: ⭐⭐⭐⭐
Message Passing: Compositionality

User Model

Item Model

Context Model

Feedback Model
accuracy
Performance and Accuracy

MovieLens Data

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

### User ID

<table>
<thead>
<tr>
<th>User Job</th>
<th>User Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>Male</td>
</tr>
<tr>
<td>Academic</td>
<td>Female</td>
</tr>
<tr>
<td>Artist</td>
<td>Male</td>
</tr>
<tr>
<td>Admin</td>
<td>Female</td>
</tr>
<tr>
<td>Student</td>
<td>Male</td>
</tr>
<tr>
<td>Customer Service</td>
<td>Female</td>
</tr>
<tr>
<td>Health Care</td>
<td>Female</td>
</tr>
<tr>
<td>Managerial</td>
<td>Male</td>
</tr>
<tr>
<td>Farmer</td>
<td>Female</td>
</tr>
<tr>
<td>Homemaker</td>
<td>Male</td>
</tr>
</tbody>
</table>

### User Age

- <18
- 18-25
- 25-34
- 35-44
- 45-49
- 50-55
- >55

### User Gender

- Male
- Female

### Movie ID

- Action
- Adventure
- Animation
- Children’s
- Comedy
- Crime
- Documentary
- Drama
- Farmer
- Fantasy
- Film Noir

### Movie Genre

- Horror
- Musical
- Mystery
- Romance
- Thriller
- Sci-Fi
- War
- Western

### MovieLens – 1,000,000 ratings

- 6,040 users
- 3,900 movies
MovieLens with Thresholds Model (ADF), Training Time = 1 Minute

Mean Absolute Error

K

0 2 5 10 20

MetaData Off
MetaData On
Lam et al.

0.6 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7
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, \(\{x_0, x_1, \cdots, x_m\}\), by:
    \[
    x_i = 1[a_i \cdot t > 0]
    \]
- \(p(\text{all bits match}) \propto \text{cosine similarity.}\)
- 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

Hash Key Bits vs Predicted Rating

- Predicted Rating distribution across different Hash Key Bits.

Hash Key Bits vs Cost per Recommendation

- Cost per Recommendation distribution across different Hash Key Bits.

- Visual representation showing the relationship between Hash Key Bits and predicted rating/cost per recommendation.
Learning to Play Go

Joint work with Thore Graepel & David Stern
• Go is game of perfect information.
• Complexity of game tree + limited computer speed → uncertainty.
• 味 ‘aji’ = ‘taste’.
• Our Approach:
  Represent uncertainty using probabilities.
• Automatic knowledge Acquisition.
• Principled management of uncertainty.
• Applications to Go:
  – Move Prediction.
  – Tactical Search.
  – Territory Prediction.
  – Monte Carlo Go.
Move Prediction

- Learning from Expert Game Records
- Move associated with a set of patterns.
  - Exact arrangement of stones.
  - Centred on proposed move.
- Sequence of nested templates.
- Inspired by work by David Stoutamire and Frank de Groot
Patterns
• 13 Pattern Sizes
  – Smallest is vertex only.
  – Biggest is full board.
**Goal:** Pattern information stored in hash table.

**Idea:** 64 bit random numbers for each template vertex: One for each of \{black, white, empty, off\}.

**Combine with XOR (Zobrist, 1970).**

- Black at (-1,2) = 128379874091837
- Empty at (1,1) = 876542534789756
- **Goal**: Pattern information stored in hash table.
- **Idea**: 64 bit random numbers for each template vertex: One for each of \{black, white, empty, off\}.
- **Combine with XOR (Zobrist, 1970).**
• **Data Size:** 180,000 games × 250 moves × 13 pattern sizes...
  ...gives **600 million potential patterns**

• **Problem:** Need to limit number stored.

• **Idea:** Keep patterns played more than n times.

• **Bloom filter:** Approximate test for set membership with minimal memory footprint.
Relative Frequencies of Pattern Sizes

- Smaller patterns matched later in game.
- Big patterns matched at beginning of game.
Move: Biggest Pattern

Table

Skill Mu=4.5, Sigma=0.2
Bayesian Ranking Model

\[ N(u_1; \mu_1, \sigma_1^2) \quad N(x_1; u_1, \beta^2) \]

\[ N(u_2; \mu_2, \sigma_2^2) \quad N(x_2; u_2, \beta^2) \]

\[ \vdots \quad \vdots \]

\[ N(u_n; \mu_n, \sigma_n^2) \quad N(x_n; u_n, \beta^2) \]

\[ \mathbb{1}(x_1 > x_2) \]

\[ \mathbb{1}(x_1 > x_n) \]

\[ p(u|\text{move, position}) = \int p(u, x|\text{move, position}) \, dx \]
Move Prediction Performance

100 Moves / second

cumulative density function

expert move rank

- Ranking Model
- Van der Werf et al. (2002)
Rank Error vs Game Phase

![Box plot showing rank error vs phase of the game.](image-url)
Rank Error vs Pattern Size
Graphical models are a very powerful language:
  - Modeling (Bayes Nets)
  - Algorithm development (Sum-Product)
  - Highly modular (Local Factors)
  - (Relatively) easy to teach (Pictorial)

Lots of open problems in learning theory that considers system limitations (CPU, storage, etc.)
Thanks!