# Learning at Scale

**Ralf Herbrich**
**Amazon**

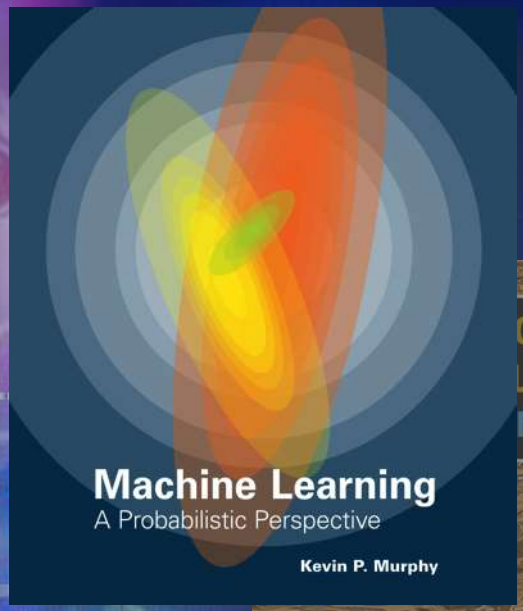
# Overview

- 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**

# Part 1: Theory

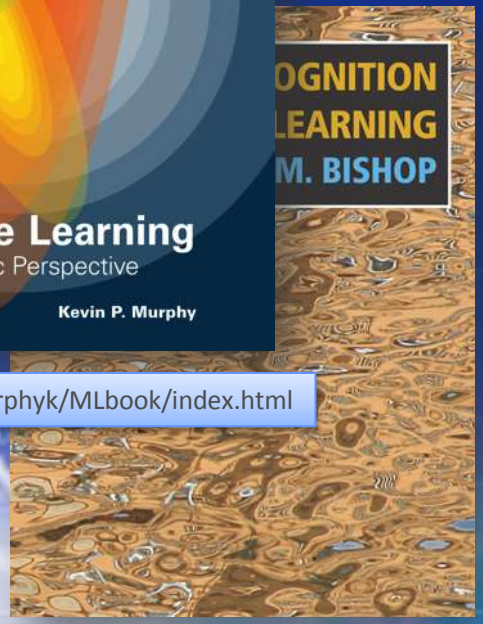
Coursera

http://www.coursera.org




http://www.cs.ubc.ca/~murphyk/MLbook/index.html

http://www.cs.ucl.ac.uk/staff/d.barber/brml/




http://research.microsoft.com/en-us/um/people/cmbishop/PRML/index.htm

# Overview

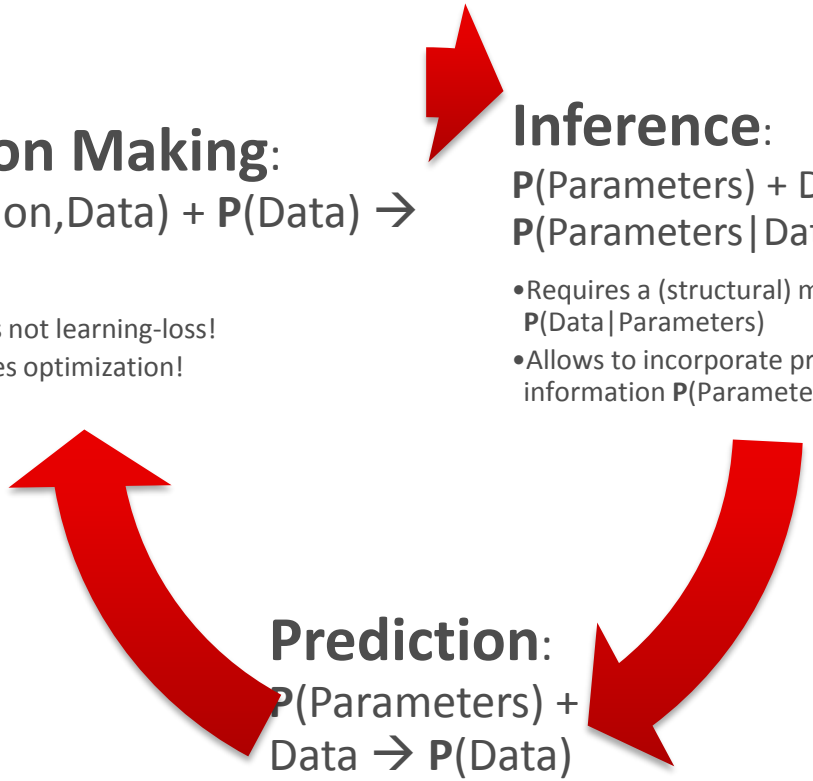
- Graphical Models
- Inference in Factor Graphs
- Approximate Message Passing
- Distributed Message Passing

# 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. $$p(A|C') > p(A|C) \quad \wedge \quad p(B|AC') = p(B|AC)$$
     $$\Rightarrow \quad p(AB|C') \geq P(AB|C)$$

**P** must be a probability measure!

# Infer-Predict-Decide Cycle



# Graphical Models

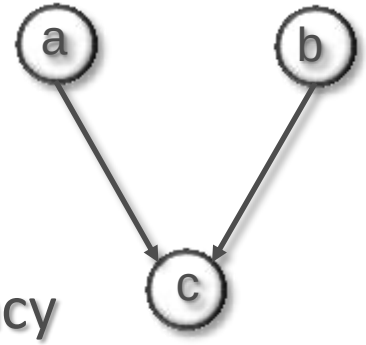
- **Definition:** Graphical representation of joint probability distribution
  - Nodes: = Variables
  - Edges: Relationship between variables
- **Variables**:
  - Observed Variables: Data
  - Unobserved Variables: 'Causes' + Temporary/Latent
- **Key Questions**:
  - (Conditional) *Dependency*: $p(a,b|c) \stackrel{?}{=} p(a|c) \cdot p(b|c)$
  - *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:**

$$p(\mathbf{x}) = \prod_i p\left(x_i | \mathbf{x}_{\text{parents}(i)}\right)$$




  - Ancestral relationship of dependency

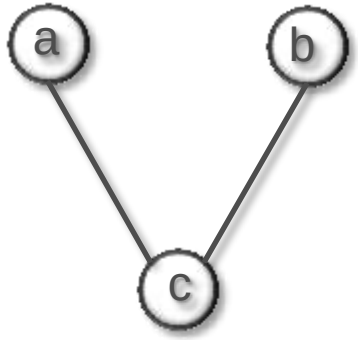
$$p(a, b, c) = p(a) \cdot p(b) \cdot p(c|a, b)$$

# Undirected Models: Markov Networks

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

- **Semantic:**

$$p(\mathbf{x}) = \frac{1}{Z} \cdot \prod_{C} \phi(x_C) \quad \phi \geq 0$$




  - Local potentials over cliques

$$p(a,b,c) = \frac{1}{Z} \cdot \phi_{ac}(a,c) \cdot \phi_{bc}(b,c)$$

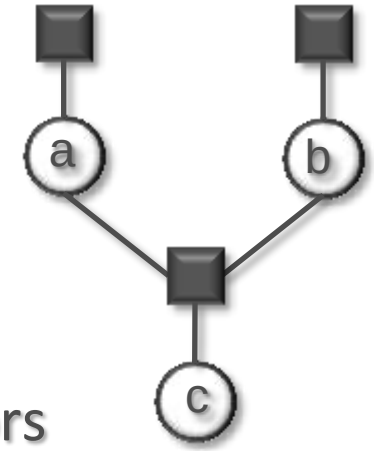
$$Z = \sum_{a} \sum_{b} \sum_{c} \phi_{ac}(a,c) \cdot \phi_{bc}(b,c)$$

# Factor Graphs

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

- **Semantic:**

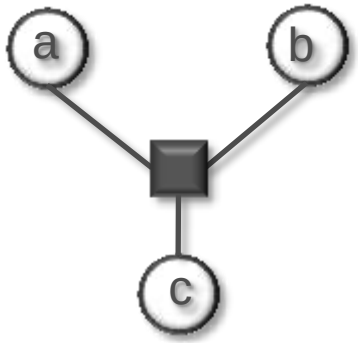
$$p(\mathbf{x}) = \prod_f f\left(\mathbf{x}_{V(f)}\right)$$



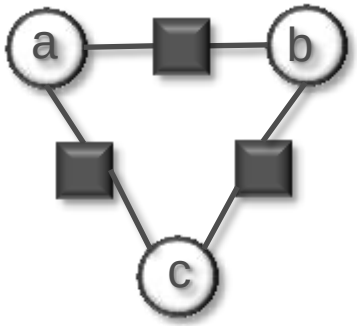

  - Local variable dependency of factors

$$p(a,b,c) = f_1(a) \cdot f_2(b) \cdot f_3(a,b,c)$$

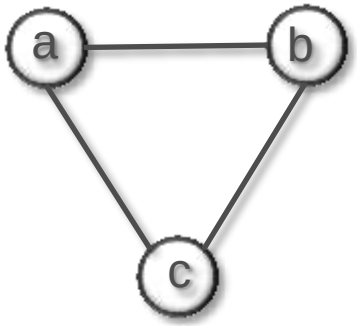
# Factor Graphs are Powerful!




$f_1(a, b, c)$




$f_1(a, b) \cdot f_2(b, c) \cdot f_3(a, c)$




$\phi(a, b, c)$

**Undirected graphical models can hide the factorisation within a clique!**

# Factor Graphs and Bayes' Law

- Bayes' law

$$p(\mathbf{s}|y) \propto p(y|\mathbf{s}) \cdot p(\mathbf{s})$$

- Factorising prior

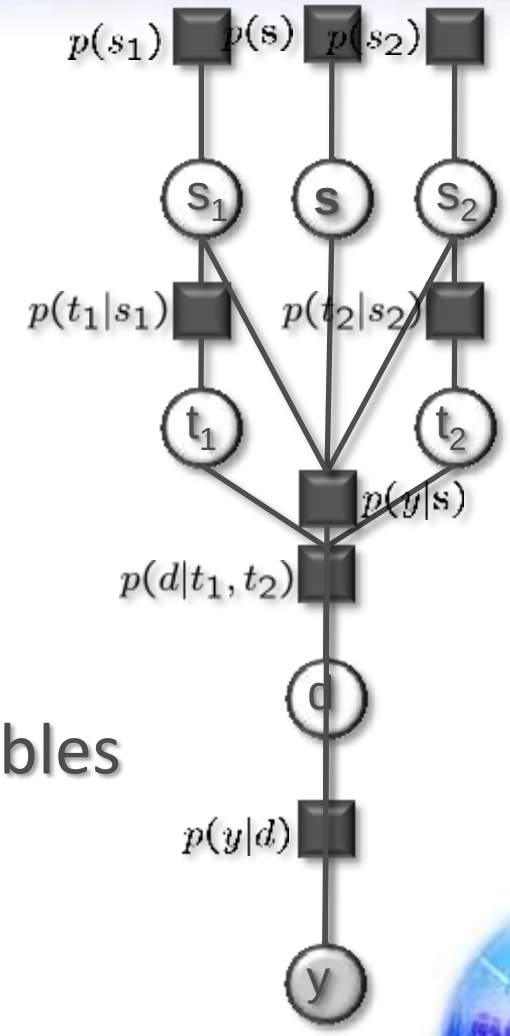
$$p(\mathbf{s}) = p(s_1) \cdot p(s_2)$$

- Factorising likelihood

$$p(y, \mathbf{t}, d|\mathbf{s}) = \prod_i p(t_i|s_i) \cdot p(d|t_1, t_2) \cdot p(y|d)$$

- Inference: Sum out latent variables

$$p(y|\mathbf{s}) = \sum_{\mathbf{t}} \sum_{d} p(y, \mathbf{t}, d|\mathbf{s})$$



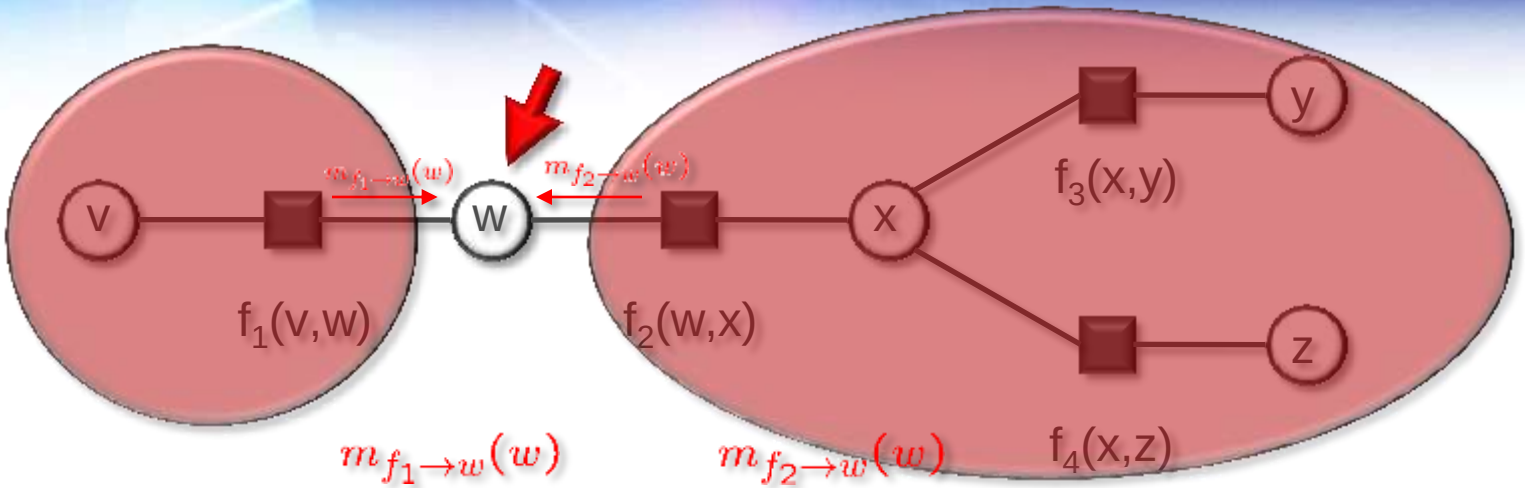
# Summary

| | Dependency | Efficient Inference | Usage |
|---|---|---|---|
| **Bayesian Networks** | **Yes** | **Somewhat** | Ancestral Generative Process |
| **Markov Networks** | **Yes** | **No** | Local Couplings and Potentials |
| **Factor Graphs** | **No** | **Yes** | Efficient, distributed inference |

# Overview

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

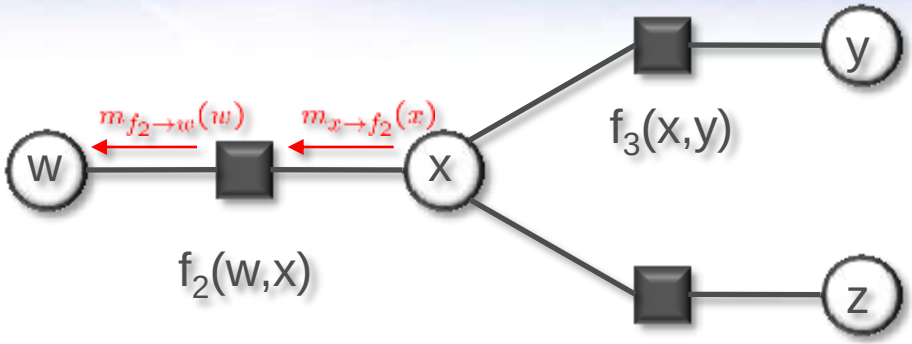
# Factor Trees: Separation




$$p(w) = \left[ \sum_v \sum_x f_1(v,w) \sum_y f_3(x,y) \sum_z f_4(x,z) \right] \left[ \sum_x \sum_y \sum_z f_2(w,x) f_3(x,y) f_4(x,z) \right]$$

**Observation:** Sum of products becomes product of sums of all messages from neighbouring factors to variable!

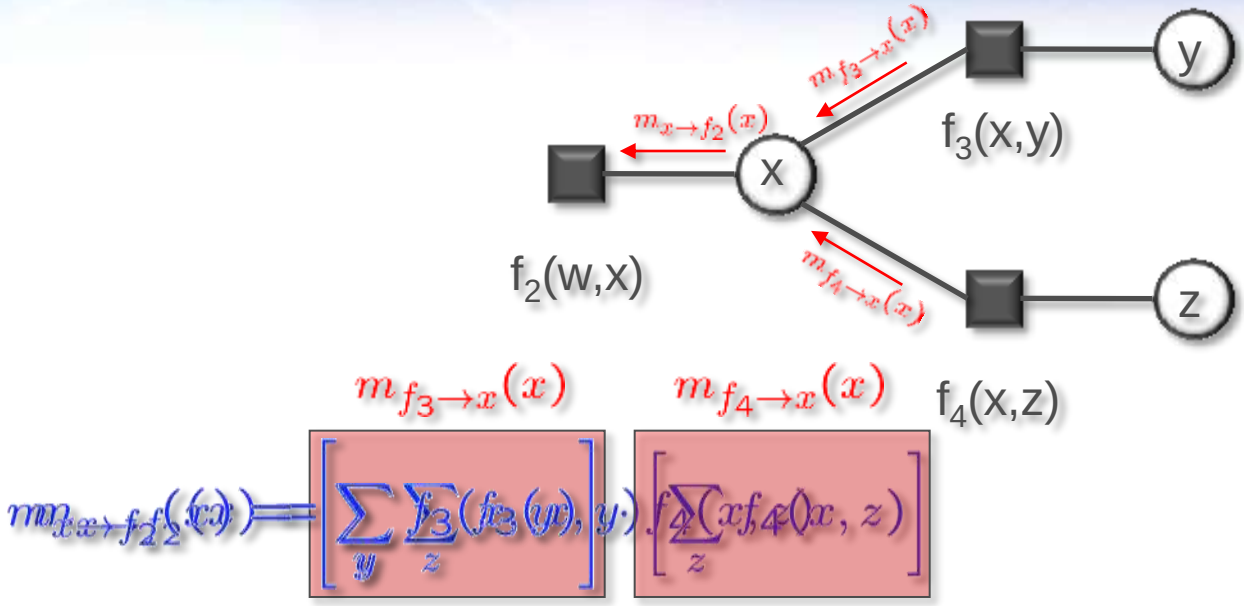
# Messages: From Factors To Variables




$$m_{f_2 \to w}(w) = \sum_x f_2(w,x) \underbrace{\left[ \sum_y f_3(x,y) \sum_z f_4(x,z) \right]}_{m_{x \to f_2}(x)}$$

**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)

$$p(t) = \prod_{f \in F_t} m_{f \to t}(t)$$

$$m_{f \to t_1}(t_1) = \sum_{t_2} \sum_{t_3} \cdots \sum_{t_n} f(t_1, t_2, t_3, \ldots) \prod_{i>1} m_{t_i \to f}(t_i)$$

$$m_{t \to f}(t) = \prod_{f_j \in F_t \setminus \{f\}} m_{f_j \to t}(t)$$

- 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:** $\lambda_{f\to t}(t) := \log\left[m_{f\to t}(t)\right]$

$$\log\left[p(t)\right] = \sum_{f\in F_t} \lambda_{f\to t}(t)$$

$$\lambda_{f\to t_1}(t_1) = \sum_{t_2}\sum_{t_3}\cdots\sum_{t_n} f(t_1,t_2,t_3,\ldots)\exp\left[\sum_{i>1}\lambda_{t_i\to f}(t_i)\right]$$

$$\lambda_{t\to f}(t) = \sum_{f_j\in F_t\setminus\{f\}} \lambda_{f_j\to t}(t)$$

- **Exponential Family Messages**:

$$m(t) \propto \exp\left(\psi(t)\cdot\boldsymbol{\theta}\right)$$

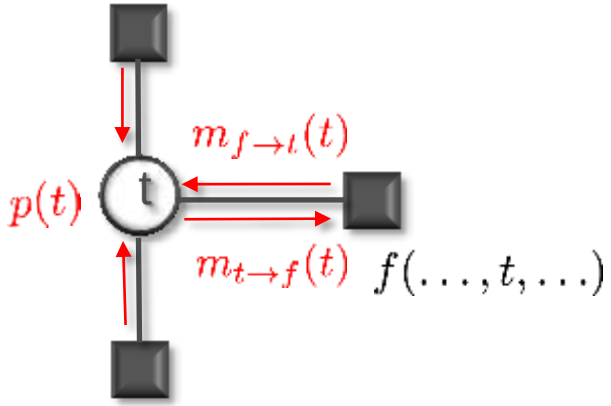
- Message updates are just additions of the parameters $\boldsymbol{\theta}$ !

# Exponential Families

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

# Practical Considerations II

- **Redundant computations:**




$$p(t) = \prod_{f\in F_t} m_{f\to t}(t)$$

$$m_{t\to f}(t) = \prod_{f_j\in F_t\setminus\{f\}} m_{f_j\to t}(t) \quad \Rightarrow \quad p(t) = m_{t\to f}(t)\cdot m_{f\to t}(t)$$

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

$$m_{t\to f}(t) = \frac{p(t)}{m_{f\to t}(t)}$$

# Overview

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

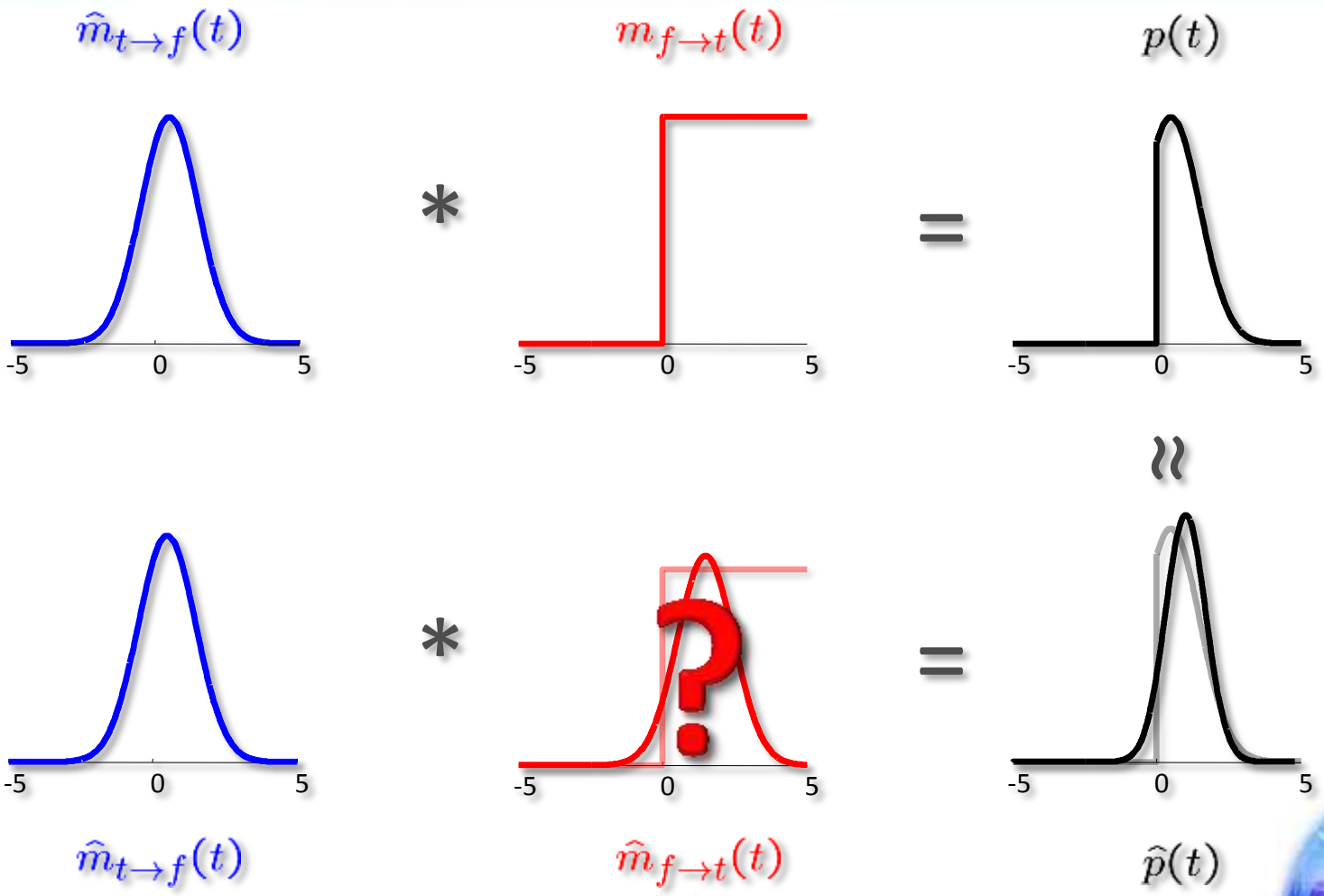
# Approximate Message Passing

- **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

$$\widehat{p}(t) = \text{argmin}_{\widehat{p}}, D\left[\overbrace{m_{f\to t}\cdot \widehat{m}_{t\to f}}^{p(t)}, \widehat{p}\right]$$

$$\widehat{m}_{f\to t}(t) = \frac{\widehat{p}(t)}{\widehat{m}_{t\to f}(t)}$$

# Approximate Message Passing



# Divergence Measures

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

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

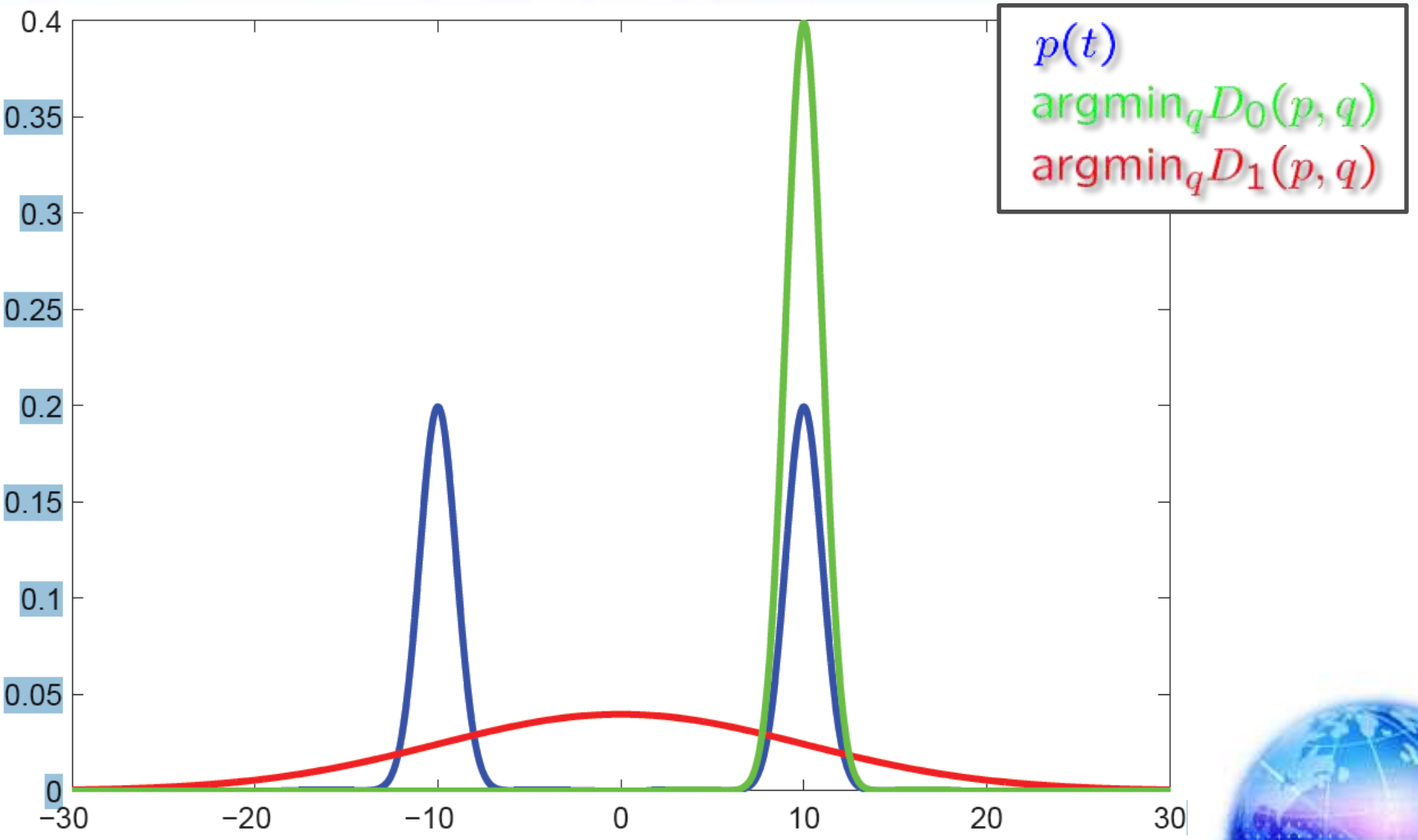
- **Minimizer for Exponential Families:** Matching the moments of the distribution $p(t)$!
- **General α-Divergence:**

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

- **Special Cases:**

$$\begin{aligned} D_0(p,q) &= \mathrm{KL}(q,p) \\ D_1(p,q) &= \mathrm{KL}(p,q) \end{aligned}$$

# α-Divergence in Pictures



# Overview

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

# Large-Data Challenge

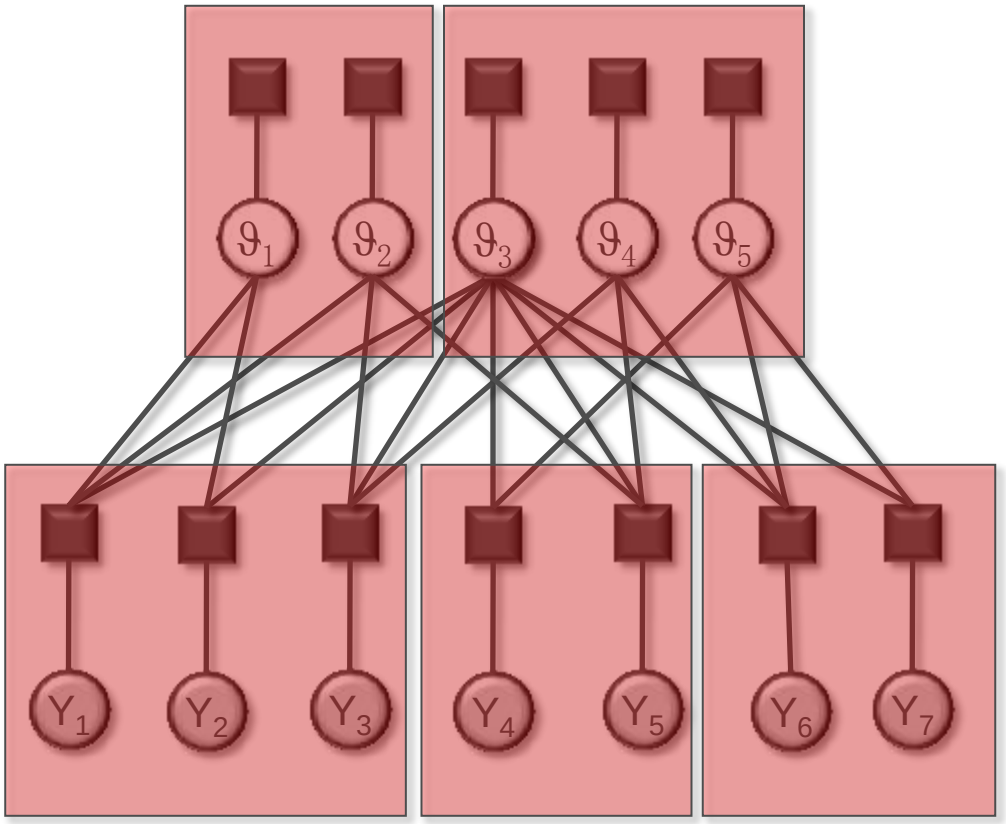
| Datasets | Number of Data Items | Number of Variables |
|---|---|---|
| Facebook News Feed | 100B news stories / day | 650M users / day |
| Facebook Social Graph | 130B friends connection | 1B users |
| Google PageRank | ~4T web links | 1T web pages |
| Amazon Forecasting | 15.6M products/ day (peak) | 20+M products |
| Xbox Gamer Ranking | >1M sessions/game (peak) | 20+M users |

## Important Constants

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

# Distributed Conditional Models

$$p(\boldsymbol{\theta}|\mathbf{X},\mathbf{Y}) \propto \prod_i p(y_i|\boldsymbol{\theta},\mathbf{x}_i) \cdot \prod_j p(\theta_j)$$




**Belief Store ("Memory")**

**Message Passing ("Communicate")**

**Data Messages ("Compute")**

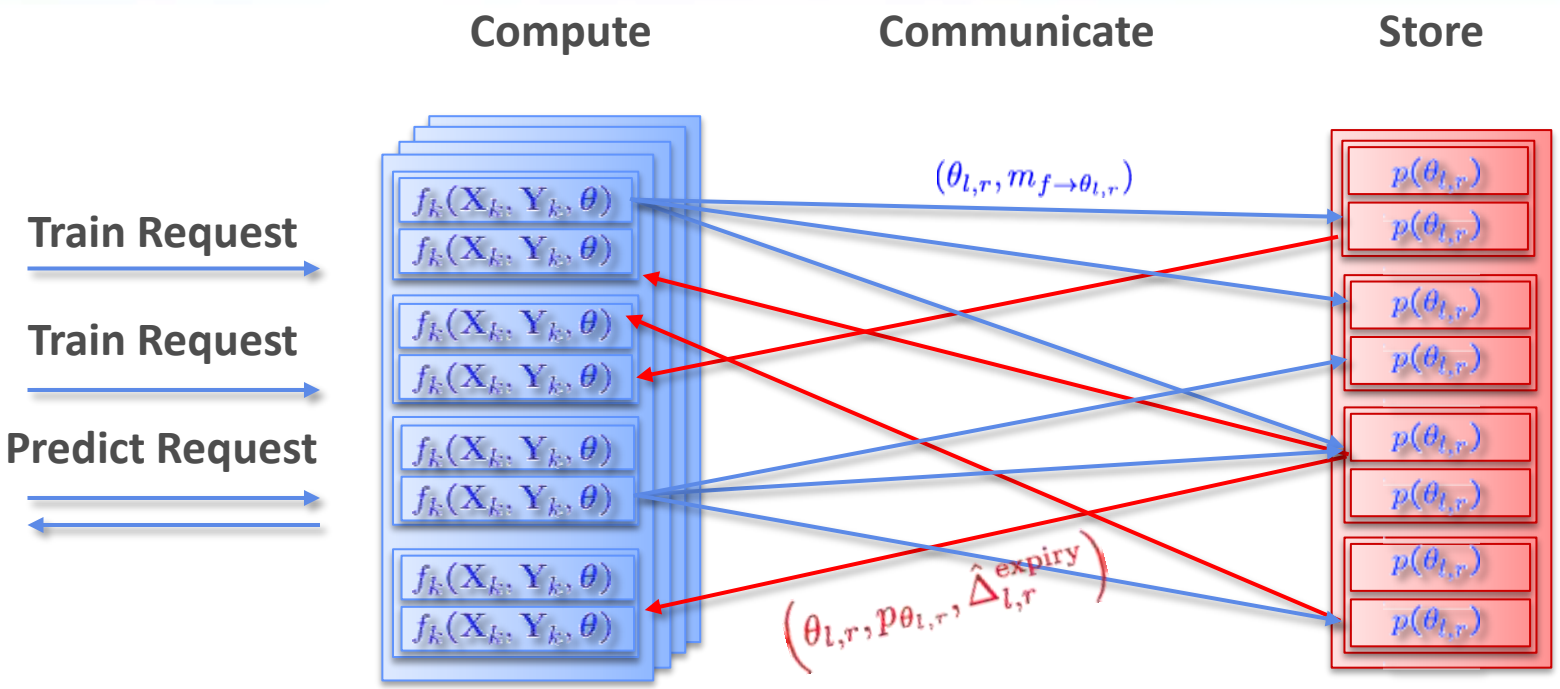
# Distributed Message Passing

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

$$\prod_i p(y_i|\boldsymbol{\theta}, \mathbf{x}_i) \cdot p(\boldsymbol{\theta}) = \prod_k \underbrace{\prod_{j=1}^{n_k} p(y_{k,j}|\boldsymbol{\theta}, \mathbf{x}_{k,j})}_{f_k(\mathbf{X}_k, \mathbf{Y}_k, \boldsymbol{\theta})} \cdot \prod_l \underbrace{\prod_{r=1}^{m_l} p(\theta_{l,r})}_{g_l(\boldsymbol{\theta}_l)}$$

- **Data factors**: $f_k(\mathbf{X}_k, \mathbf{Y}_k, \boldsymbol{\theta})$
  - Know exactly which model parameter messages get updated
- **Parameter factors**: $g_l(\boldsymbol{\theta}_l)$
  - Need to keep track of which data factors need message update

# A Systems Service View



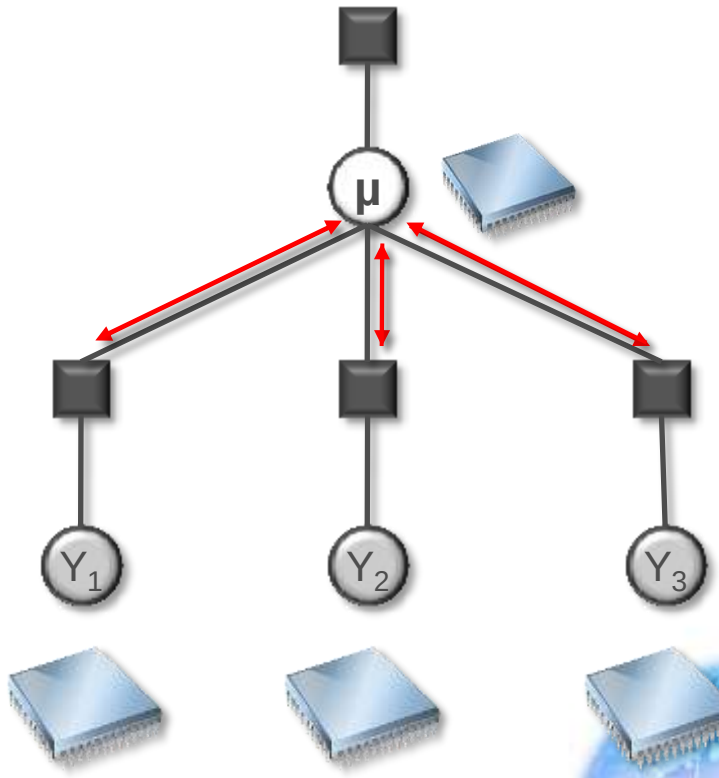
# Additional Technical Challenges

- 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**

# Relation to Map-Reduce

- **Map-Reduce**
  - **Map**: Data nodes compute messages $m_{F_k \to \mu}$ from data $y_i$ and $m_{\mu \to F_k}$
  - **Reduce**: Combine messages $m_{F_k \to \mu}$ into $p_\mu$ by multiplication
  - Vanilla MR is a single pass only!
- **Caveats**:
  - Approximate data factors need all incoming message $m_{F_k \to \mu}$!
  - Each machine needs to be able to store the belief over $\mu$

$$p(\theta|\mathbf{x},\mathbf{y}) \propto \prod_k f_k(\mathbf{Y}_k|\theta,\mathbf{X}_k)\cdot p(\theta)$$

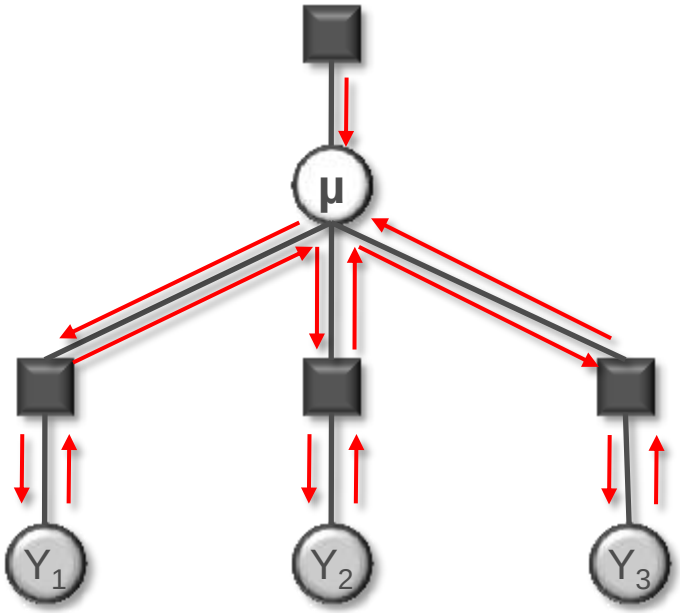


# Approximation Quality

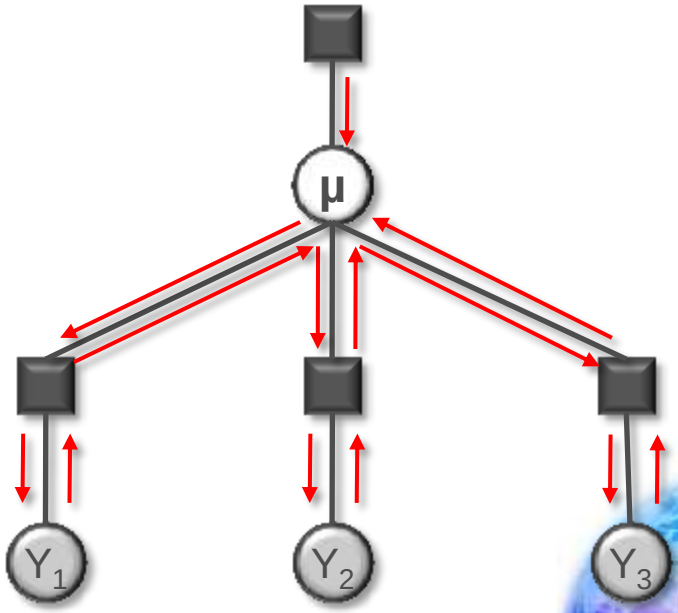
$$p(y_i|\boldsymbol{\theta}, \mathbf{x}_i) = \Phi(y_i \boldsymbol{\theta}^{\mathrm{T}} \mathbf{x}_i)$$

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

**Sequential**



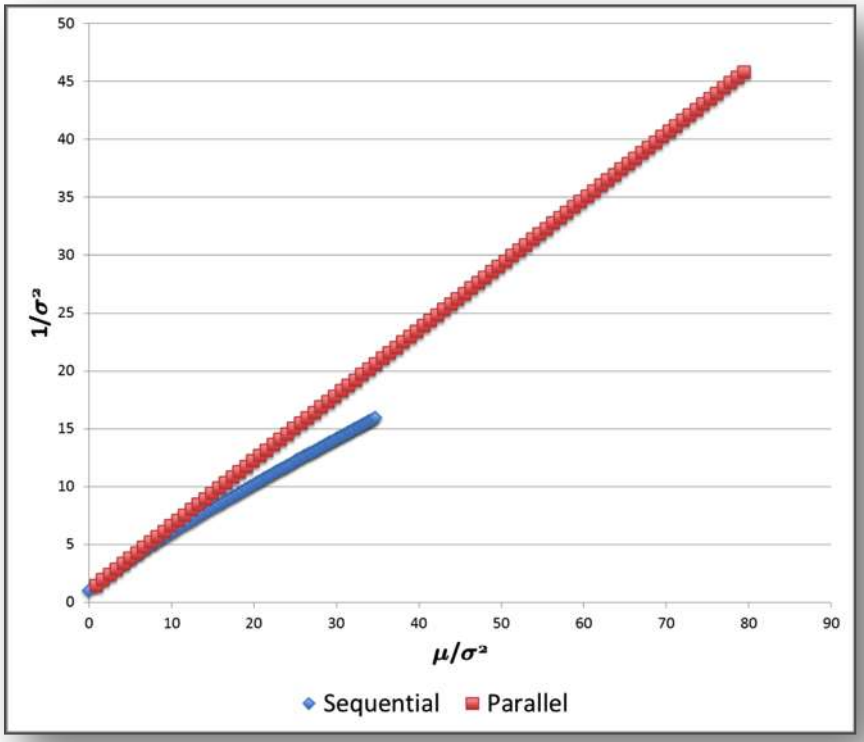

**Parallel**



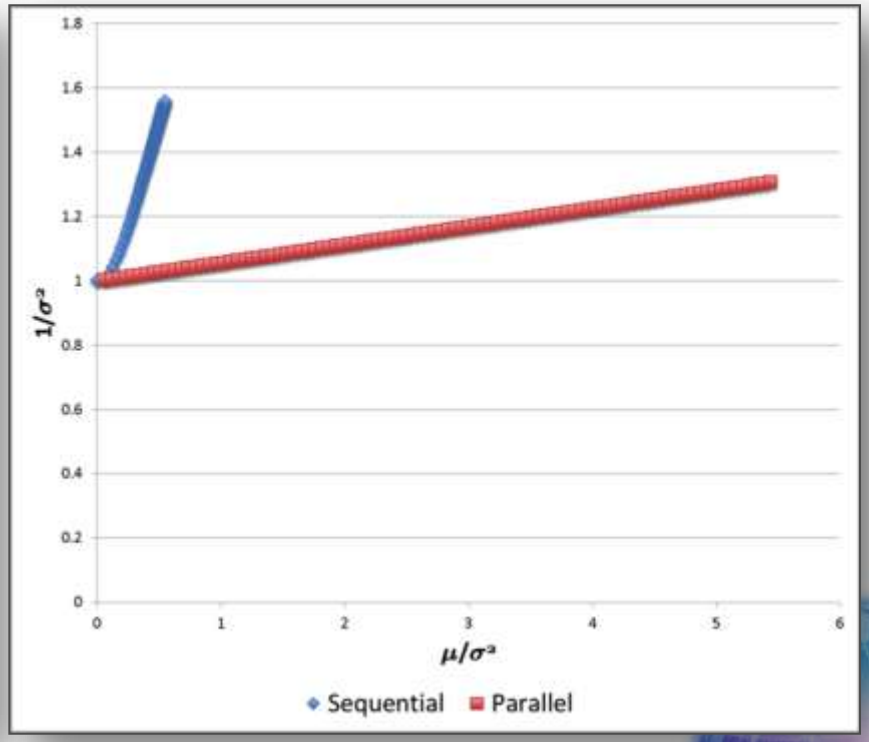
# Approximation Quality

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

## Single Bias Feature



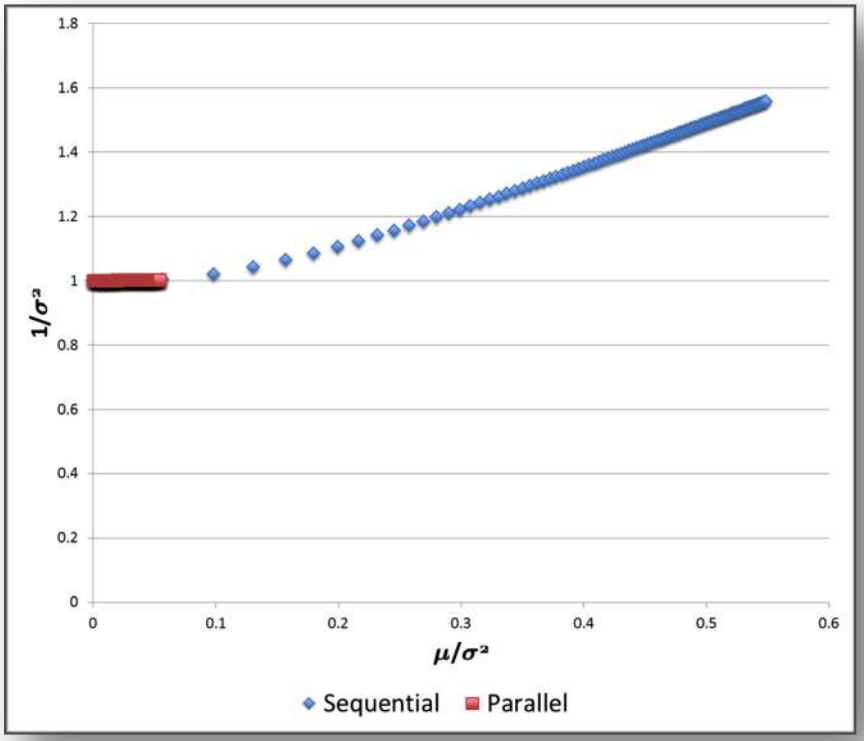

## 100 Bias Features



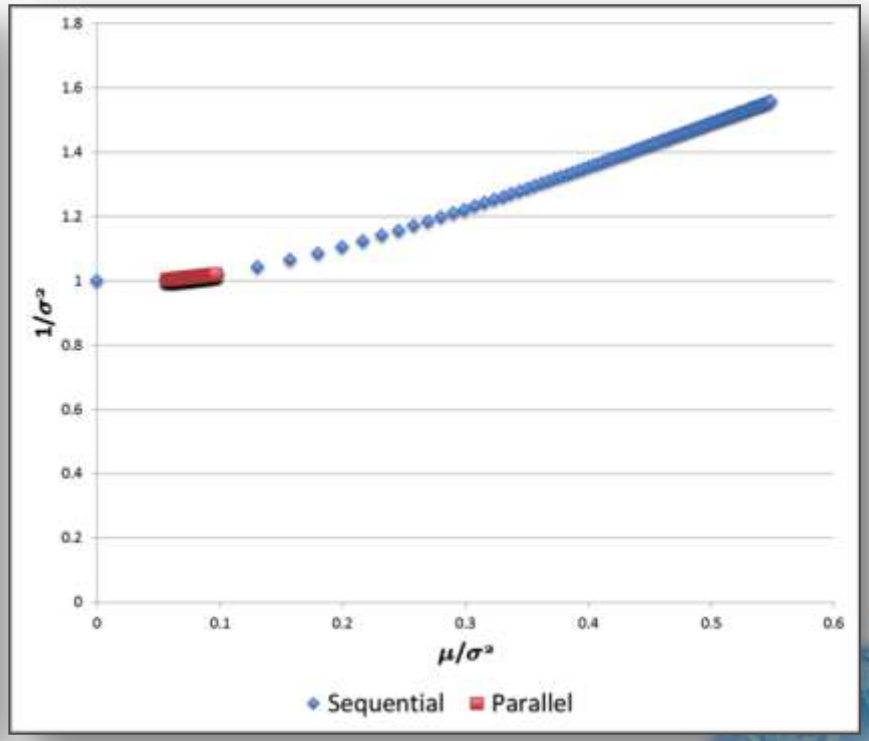
# Solution : Dampening!

$$\lambda_{f\to\theta} \Rightarrow \alpha \cdot \lambda_{f\to\theta}$$

## First Step




## Second Step



Break!

Part 2: Applications

# Overview

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

# TrueSkill™

Joint work with Thore Graepel, Tom Minka & Phillip Trelford

# Motivation

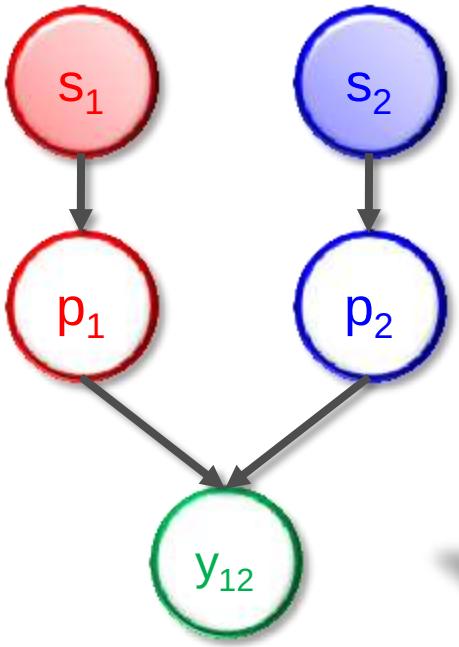
- Competition is central to our lives
  - Innate biological trait
  - Driving principle of many sports
- Chess Rating for fair competition
  - ELO: Developed in 1960 by Árpád Imre Élő
  - Matchmaking system for tournaments
- Challenges of online gaming
  - Learn from few match outcomes efficiently
  - Support multiple teams and multiple players per team

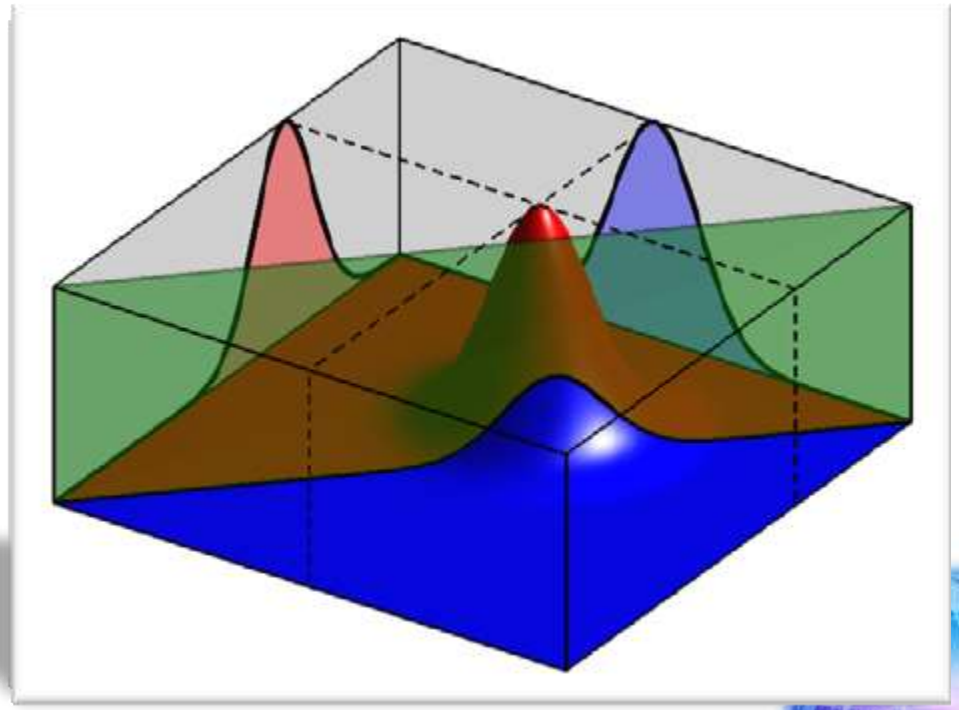
# The Skill Rating Problem

- **Given:**
  - Match outcomes: Orderings among k teams
- **Qu**
  -

# Two Player Match Outcome Model

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

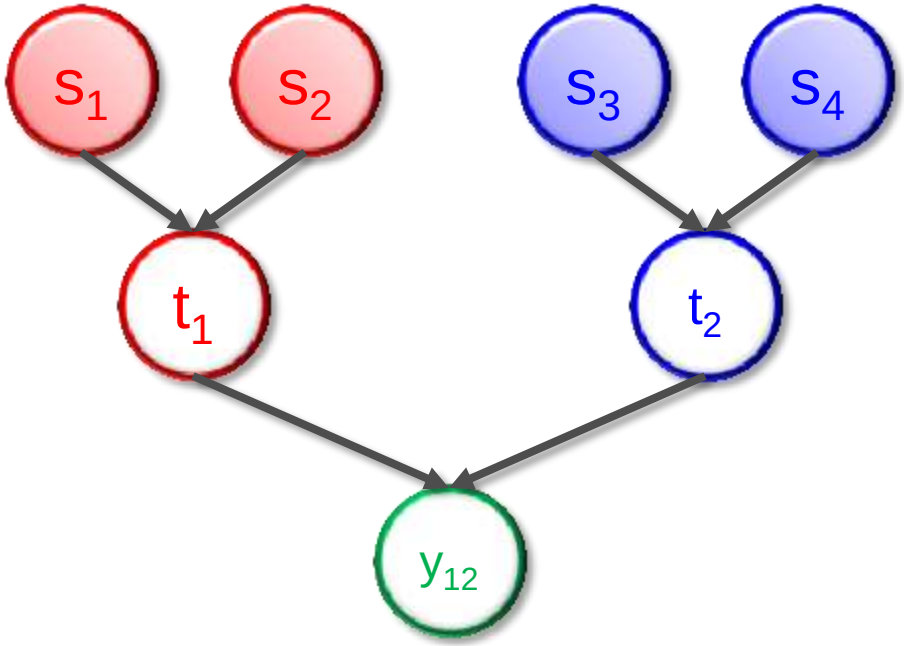


$$\mathbf{P}(y_{12} = (1,2)|p_1, p_2) = \mathbb{I}(p_1 > p_2)$$

# Two Team Match Outcome Model

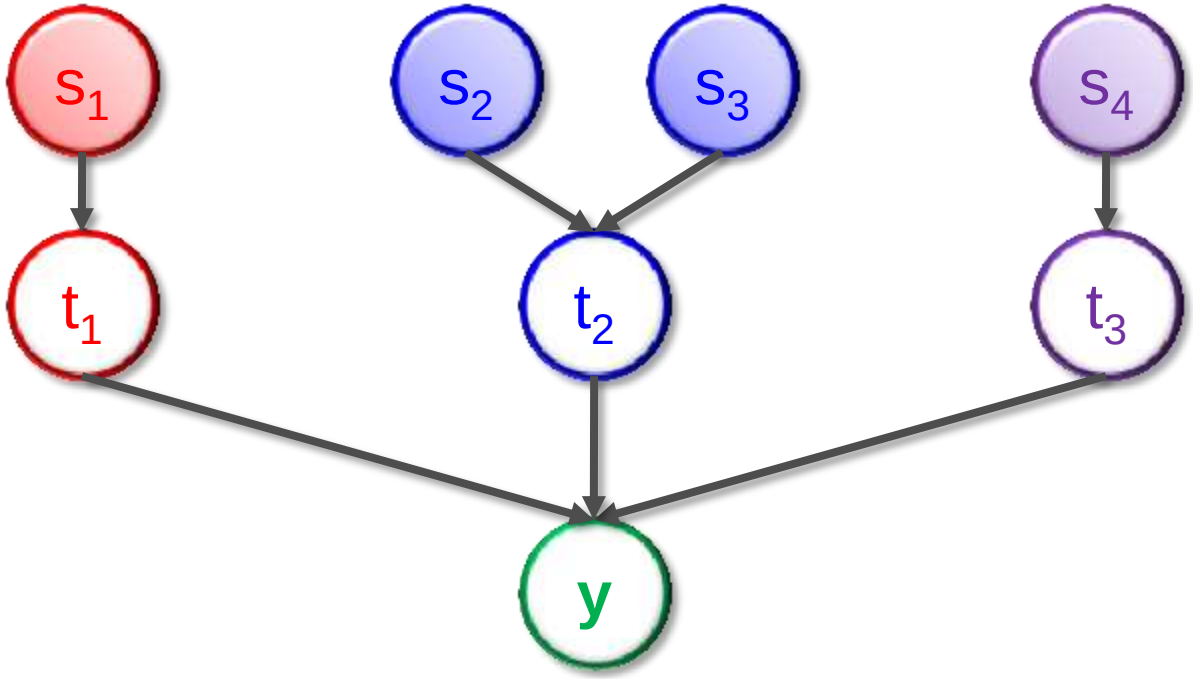
- Skill of a team is the sum of the skills of its members




$$\mathbf{P}(t_1|s_1, s_2) = \mathcal{N}\left(t_1; s_1 + s_2, 2 \cdot \beta^2\right)$$

# Multiple Team Match Outcome Model

- Possible outcomes: Permutations of the teams

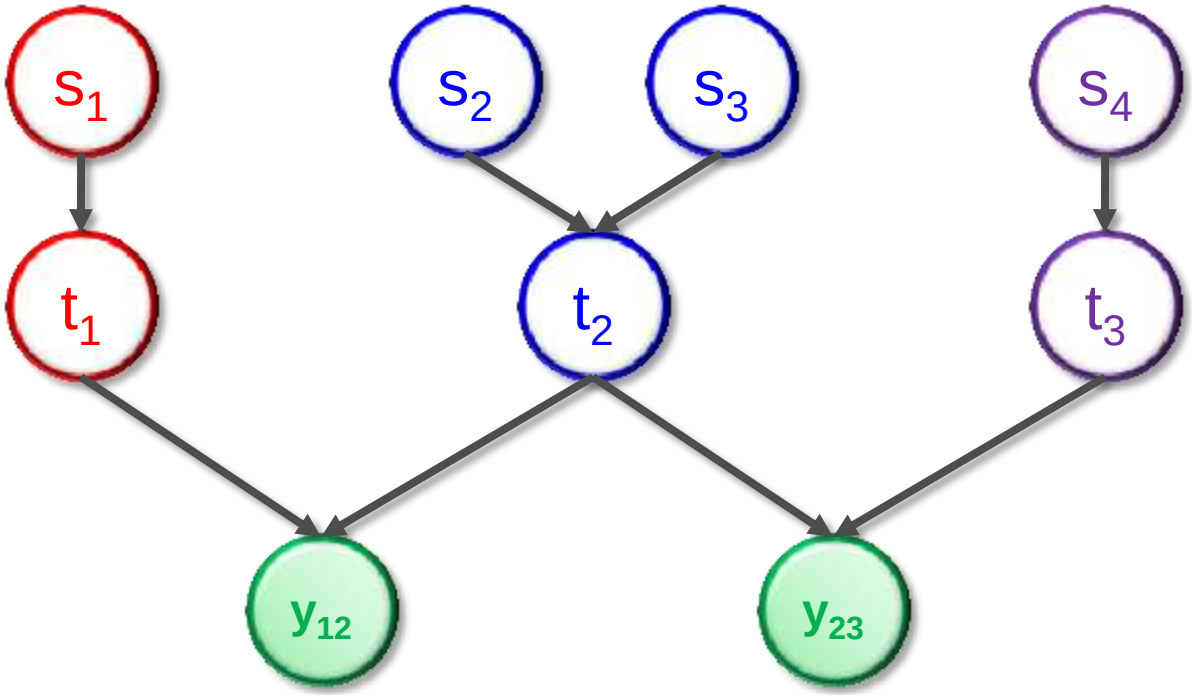



$$\mathbf{P}(\boldsymbol{y}|t_1, t_2, t_3) = \mathbb{I}(\boldsymbol{y} = (i, j, k)) \text{ where } t_i > t_j > t_k$$

# Multiple Team Match Outcome Model

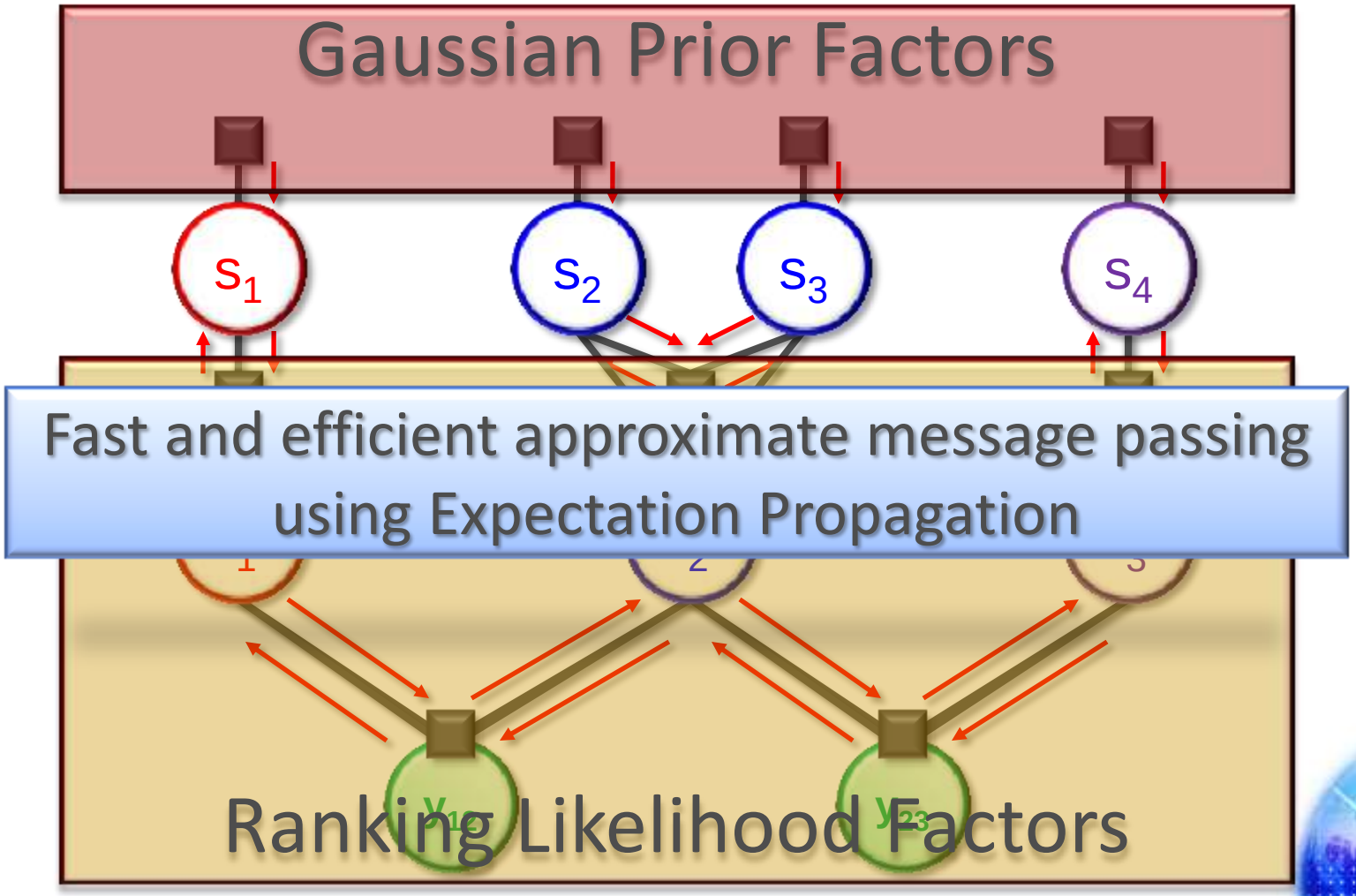
- But we are interested in the (Gaussian) posterior!

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




$y_{12} = (1, 2)$ $y_{23} = (2, 3)$

# Efficient Approximate Inference

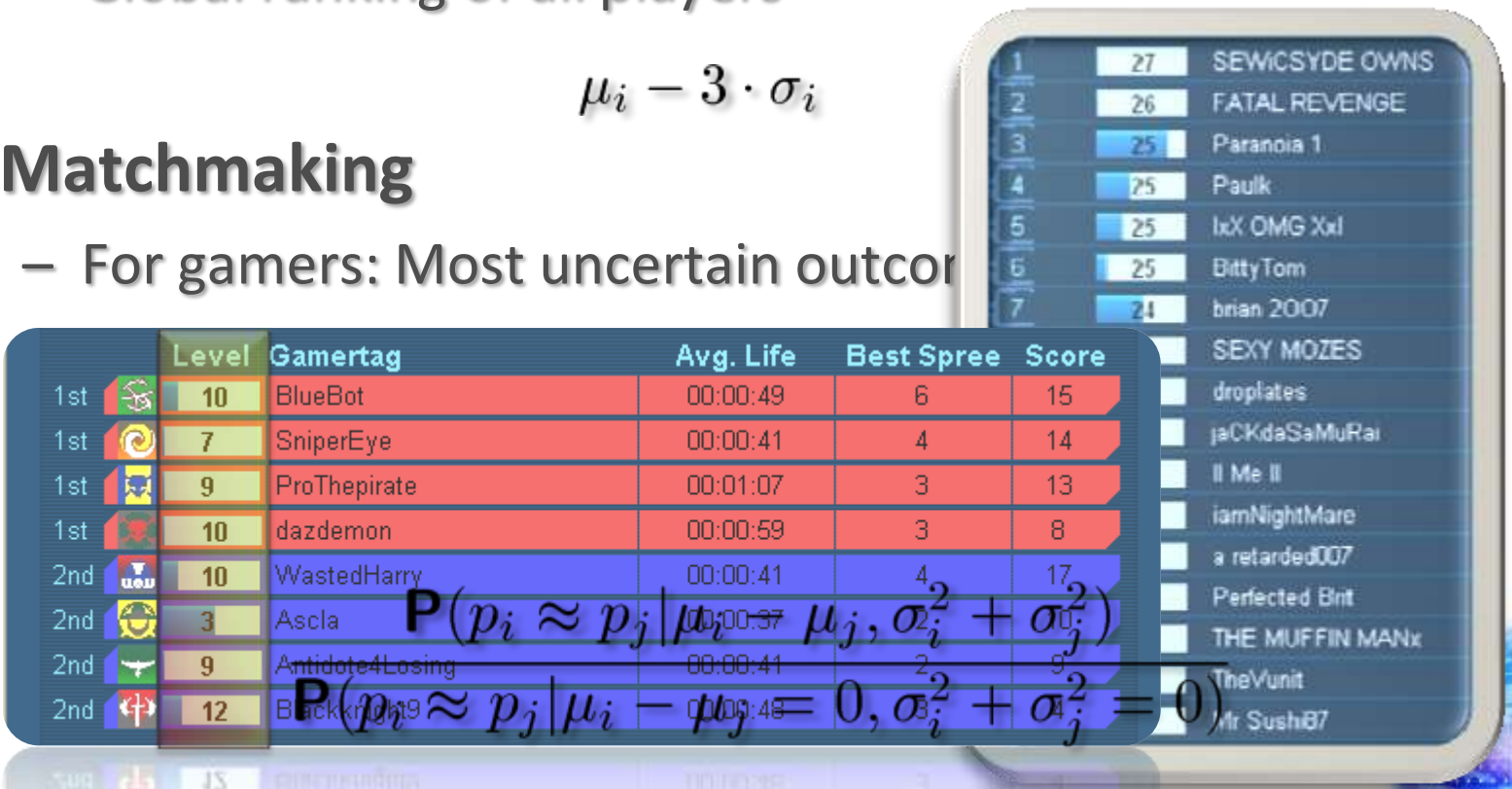


# Applications to Online Gaming

- **Leaderboard**
  - Global ranking of all players

$$\mu_i - 3 \cdot \sigma_i$$

- **Matchmaking**
  - For gamers: Most uncertain outcor



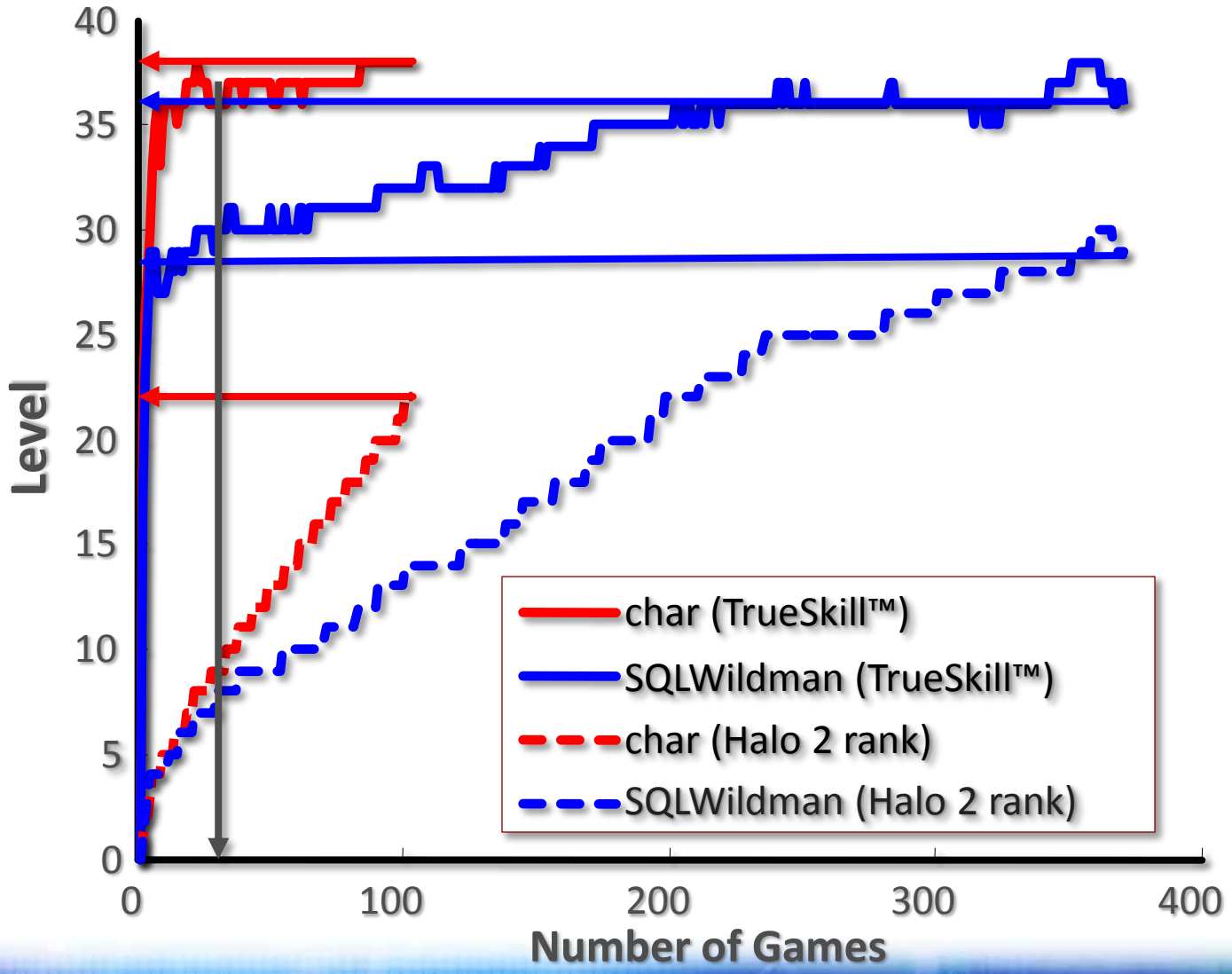

$$\mathbf{P}(p_i \approx p_j | \mu_i \approx \mu_j, \sigma_i^2 + \sigma_j^2)$$

$$\mathbf{P}(p_i \approx p_j | \mu_i - \mu_j = 0, \sigma_i^2 + \sigma_j^2 = 0)$$

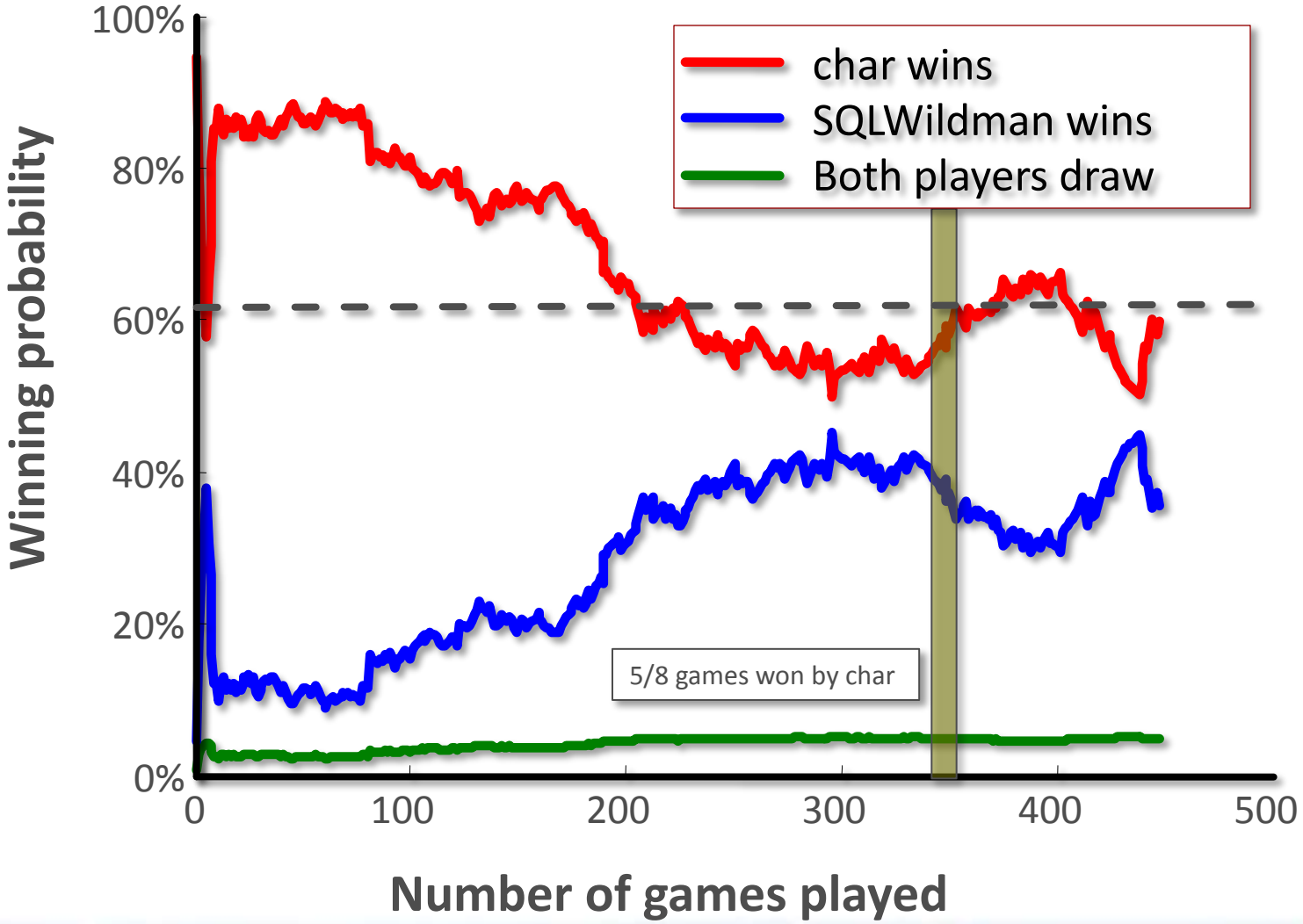
# Experimental Setup

- **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



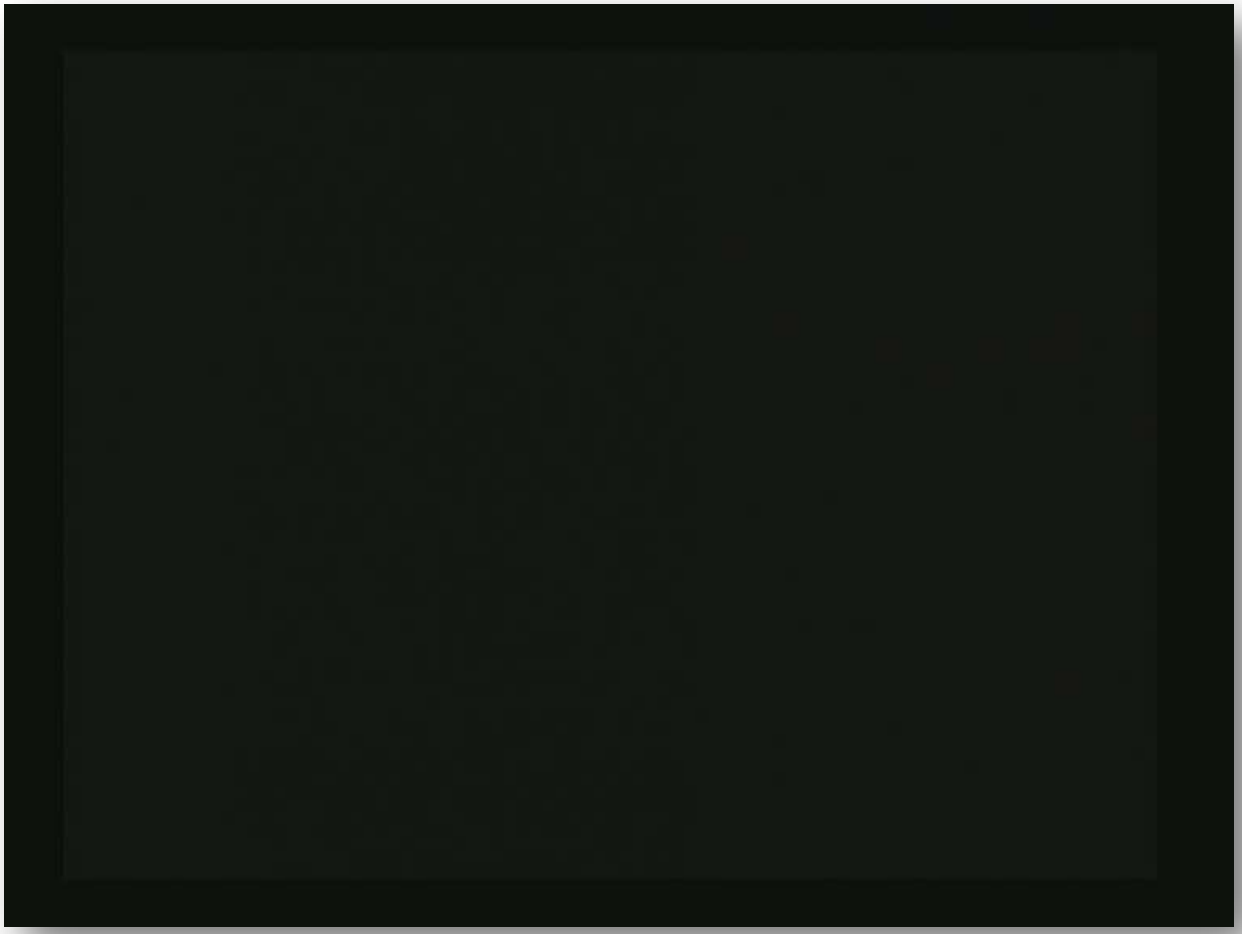
# Convergence Speed (ctd.)



# Xbox 360 & Halo 3

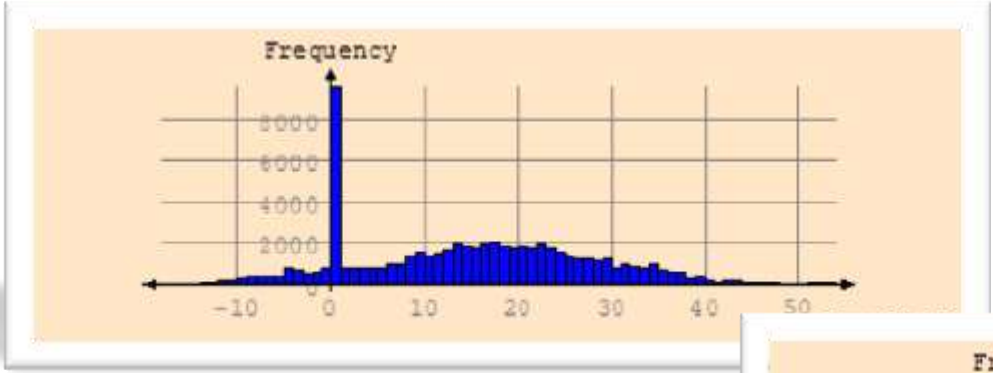
- **Xbox 360 Live**
  - Launched in September 2005
  - Every game uses TrueSkill™ to match players
  - > 10 million players
  - > 2 million matches per day
  - > 2 billion hours of gameplay
- **Halo 3**
  - Launched on 25th September 2007
  - Largest entertainment launch in history
  - > 200,000 player concurrently (peak: 1,000,000)

# Halo 3 in Action



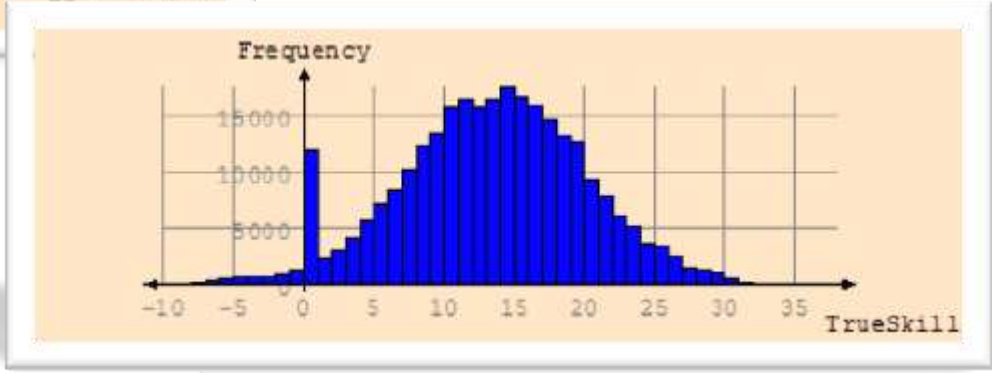
# Halo 3 Public Beta Analysis

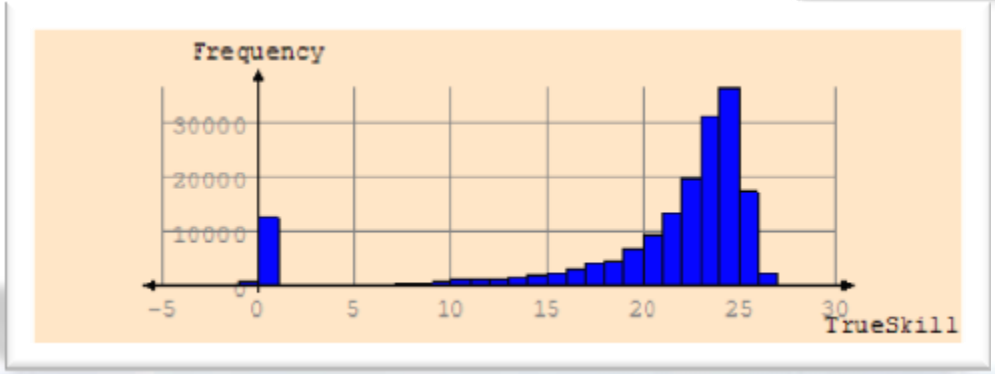
# Skill Distributions of Online Games




**Golf (18 holes)**: 60 levels

**Car racing (3-4 laps)**: 40 levels

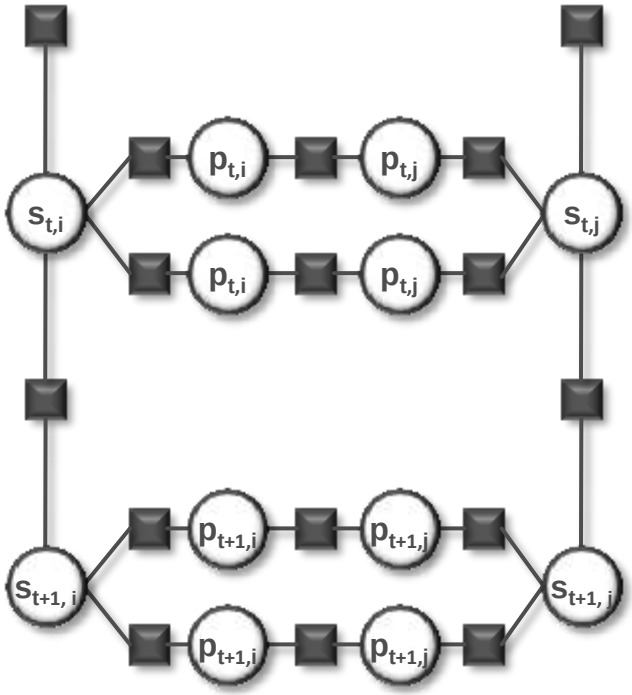


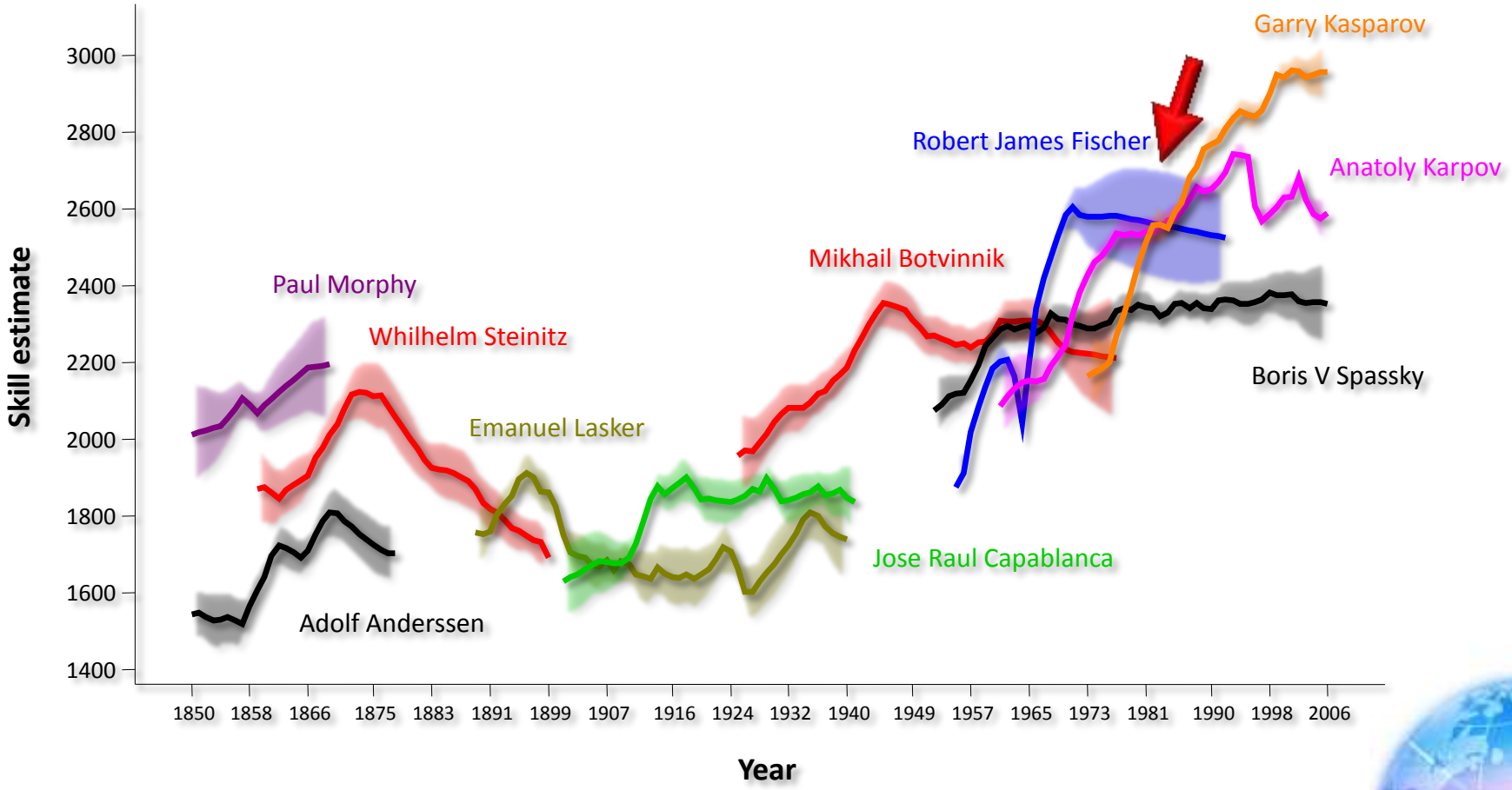

**UNO (chance game)**: 10 levels

# 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



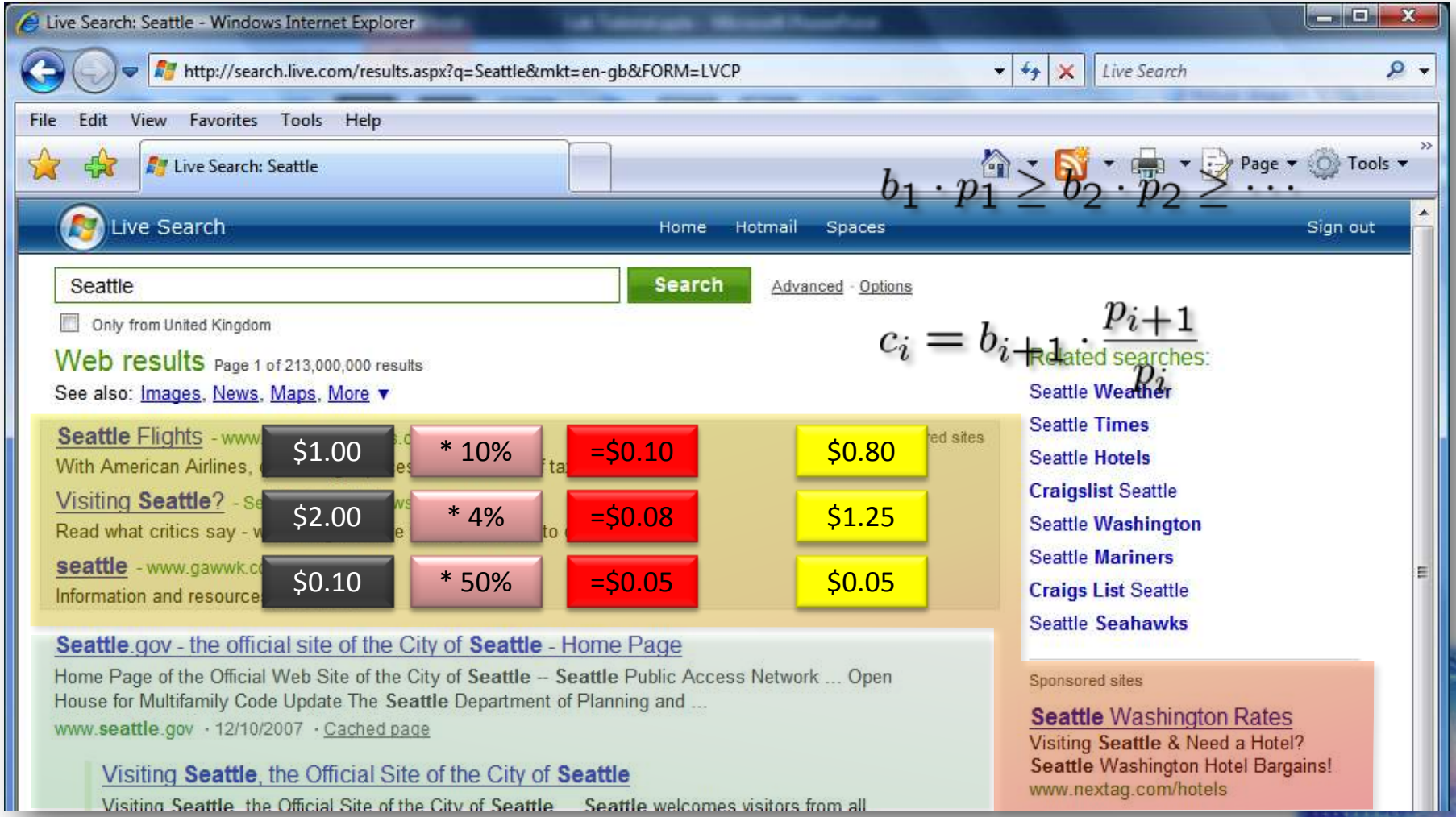
# ChessBase Analysis: 1850 - 2006



# Online Advertising

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

# 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

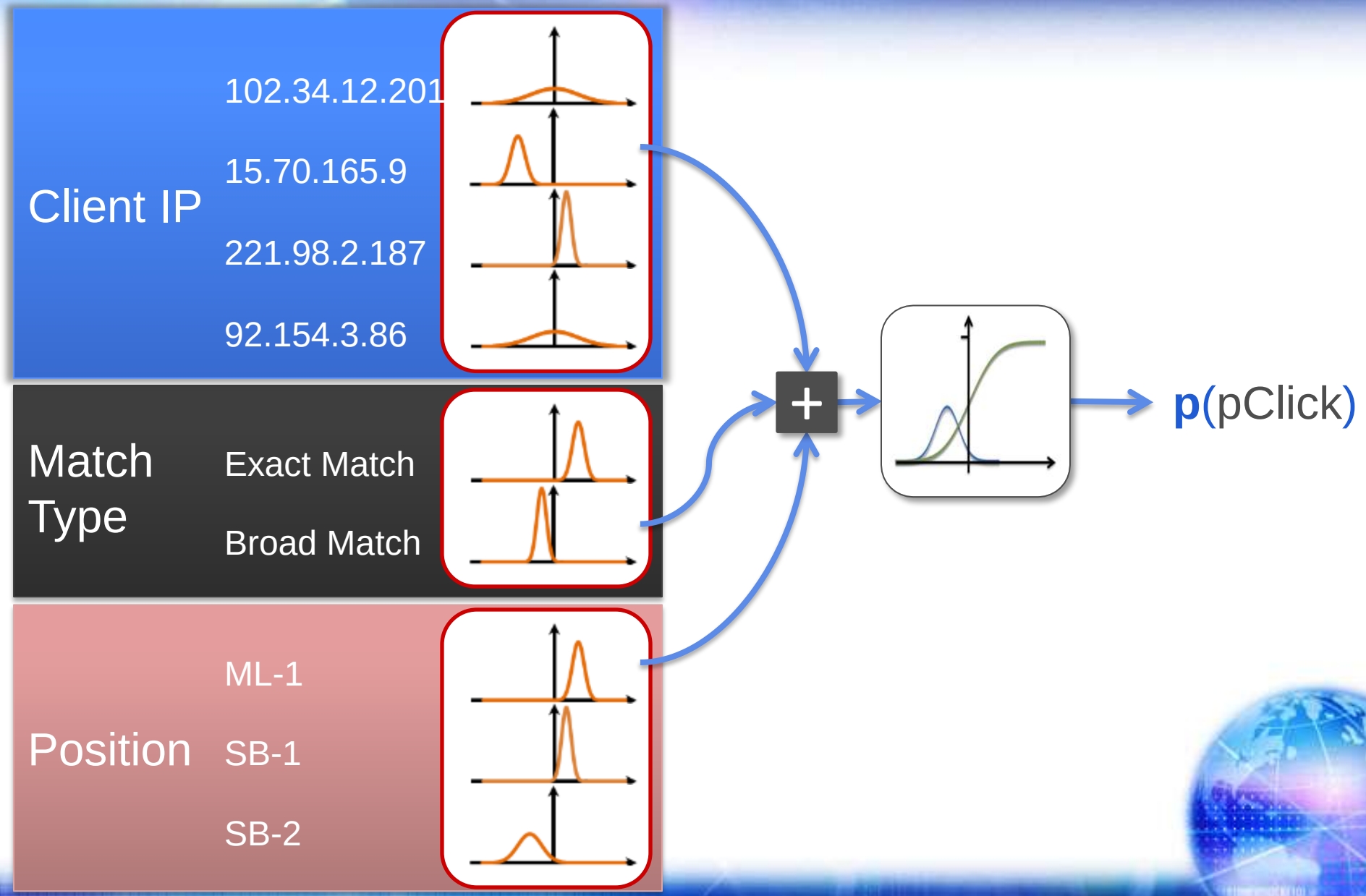
- **Learning algorithm speed requirement**:
  - 5,787 impression updates / sec
  - 172.8 μs per impression update

# The Flow of Information

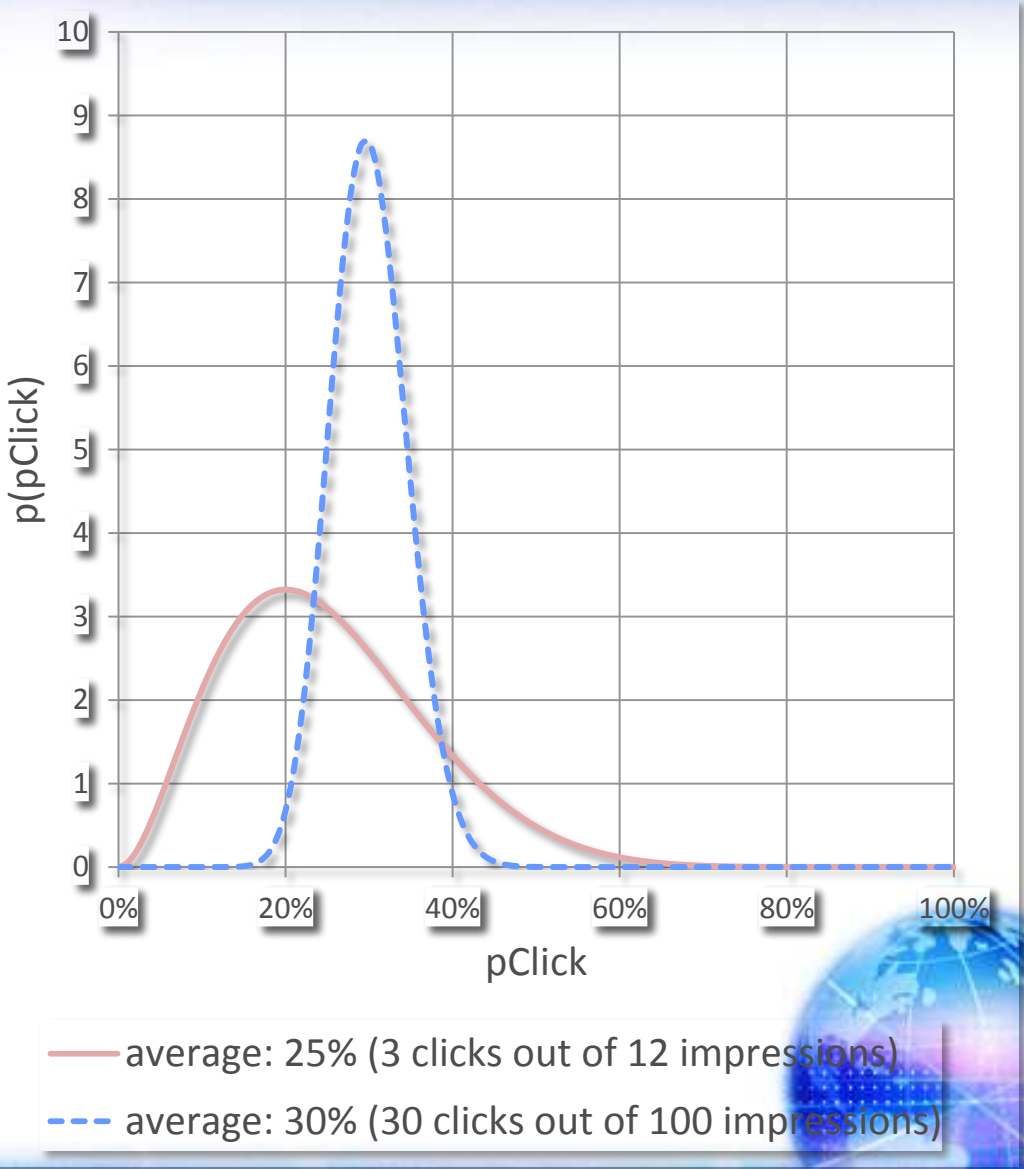
User interaction → Raw Logs → Structured Data

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

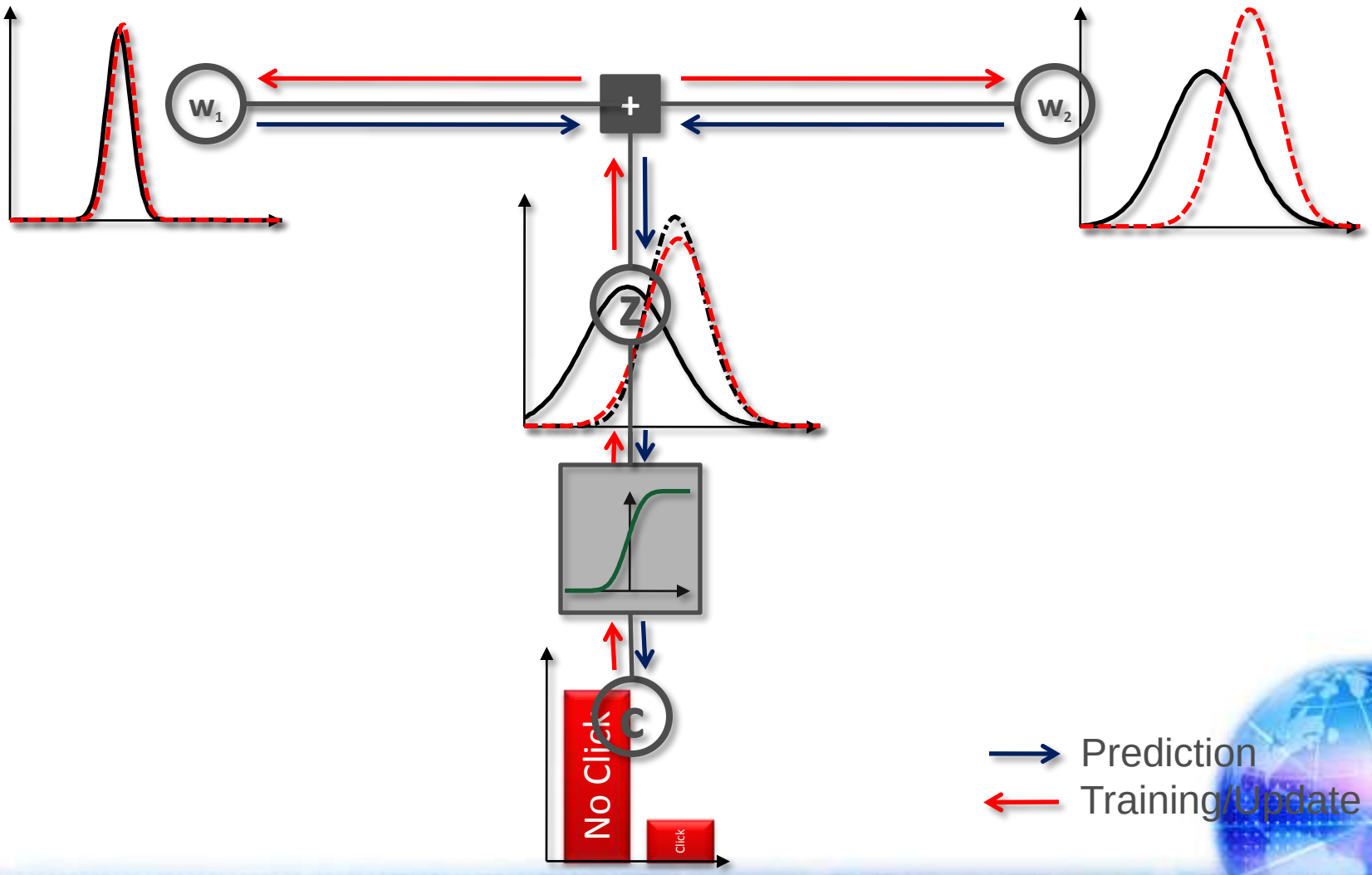
# Uncertainty: Bayesian Probabilities



# Principled Exploration



# Training Algorithm in Action

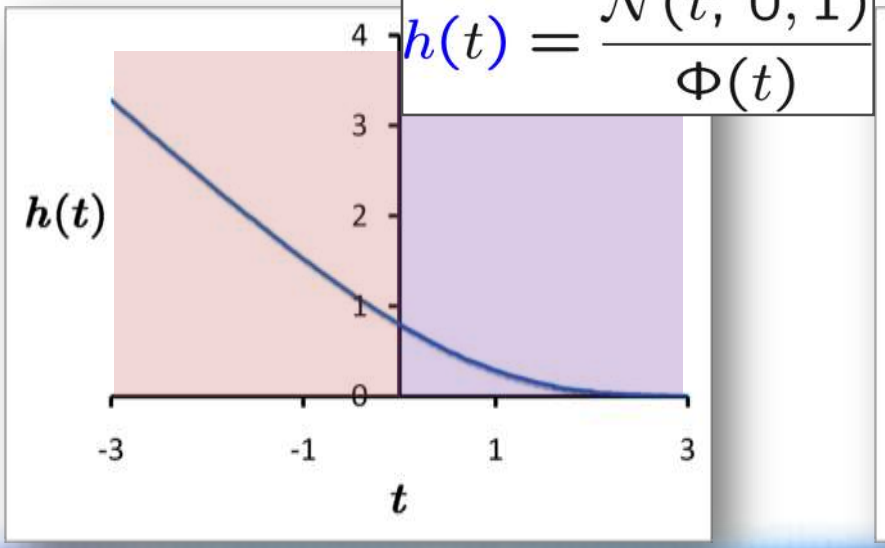


# Inference: An Optimization View

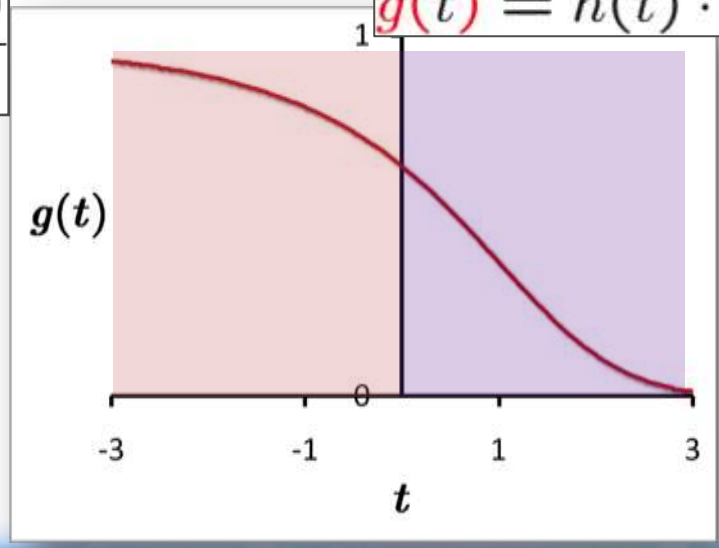
$$\mu_i \leftarrow \mu_i + \frac{\sigma_i^2}{s} \cdot h\left[\frac{\sum_{j=1}^{d} \mu_j}{s}\right] \qquad \sigma_i^2 \leftarrow \sigma_i^2 \left(1 - \frac{\sigma_i^2}{s^2} \cdot g\left[\frac{\sum_{j=1}^{d} \mu_j}{s}\right]\right)$$

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

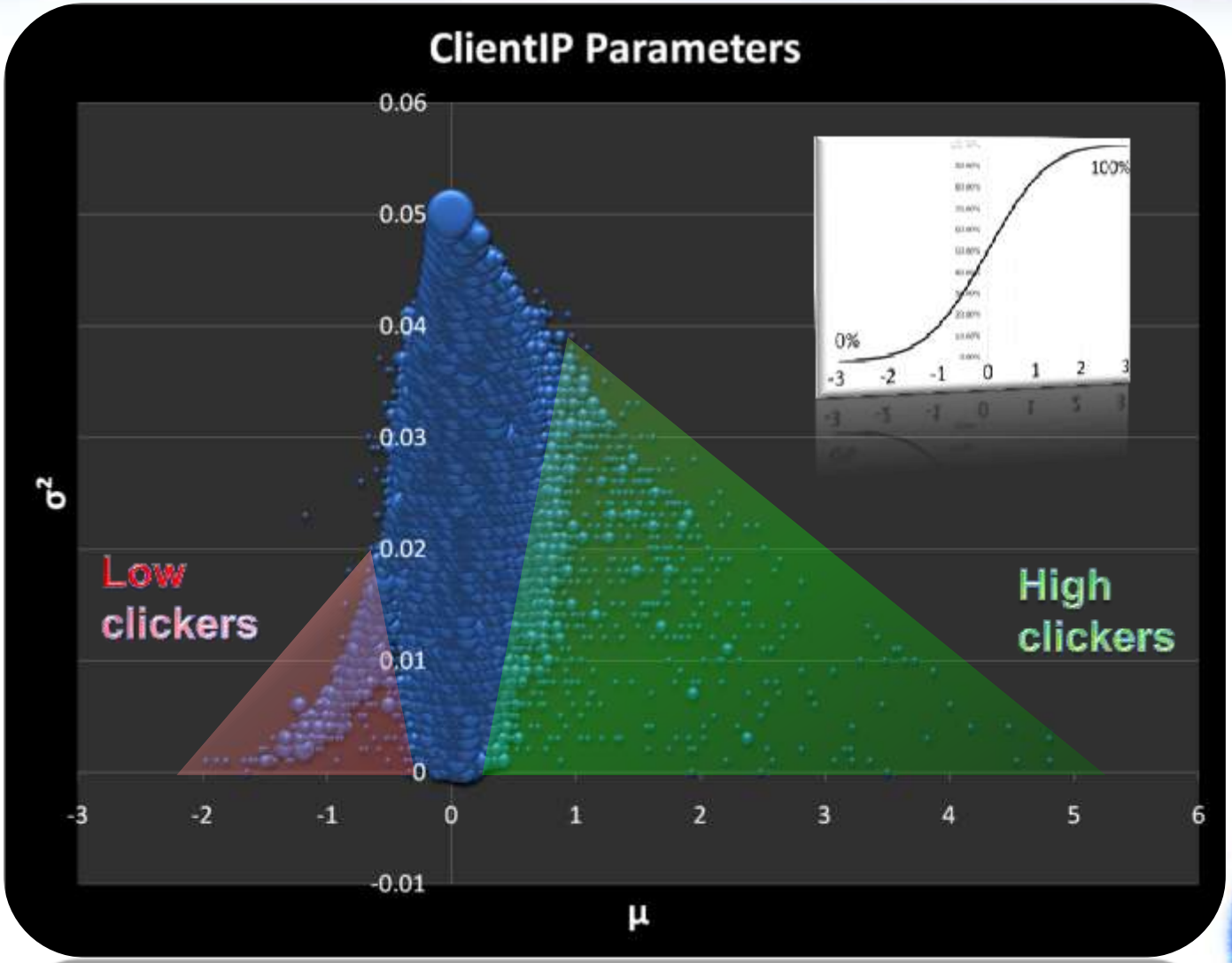
$$h(t) = \frac{\mathcal{N}(t;\, 0, 1)}{\Phi(t)}$$



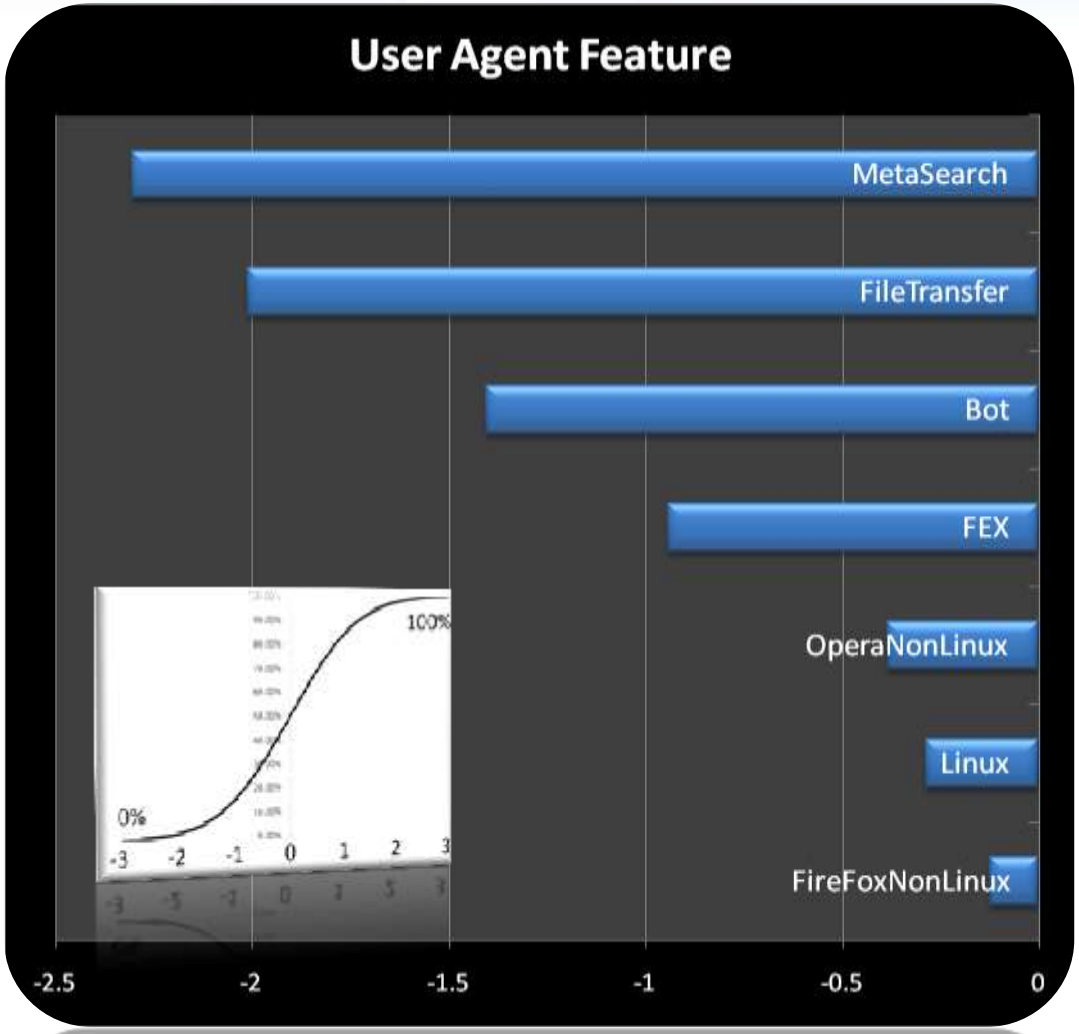

$$g(t) = h(t) \cdot [h(t) + t]$$



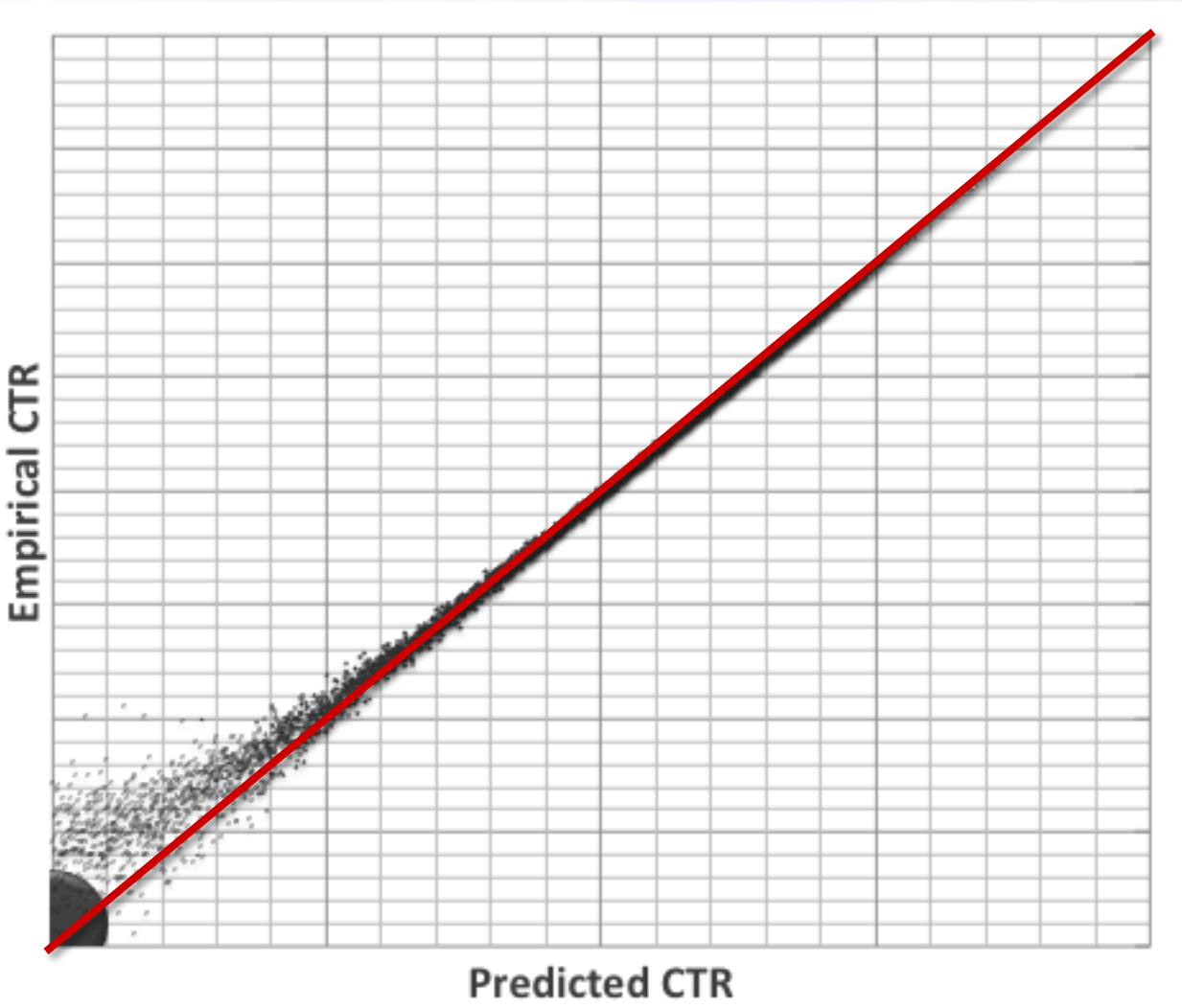
# Client IP: Mean & Variance



# UserAgent: Mean Posterior Effects

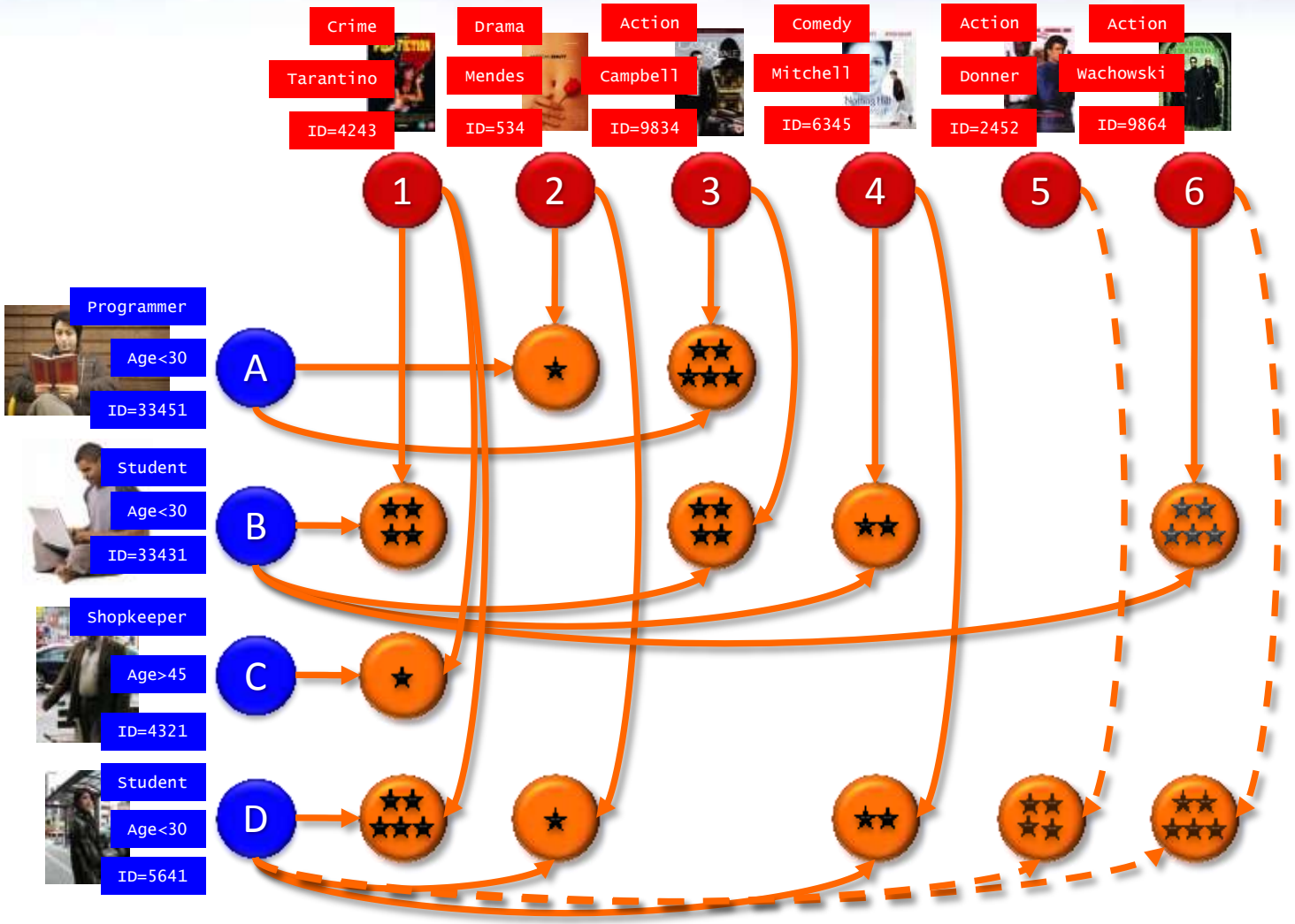


# Accuracy



# MatchBox

Joint work with Thore Graepel, Joaquin Quiñonero Candela, David Stern


Crime
Tarantino
ID=4243
Drama
Mendes
ID=534
Action
Campbell
ID=9834
Comedy
Mitchell
ID=6345
Action
Donner
ID=2452
Action
Wachowski
ID=9864
1
2
3
4
5
6
Programmer
Age<30
ID=33451
Student
Age<30
ID=33431
Shopkeeper
Age>45
ID=4321
Student
Age<30
ID=5641
A
B
C
D

# Matchbox With Metadata

User Metadata

Item Metadata

ID=234

Mal

British

Camera

SLR

$u_{01}$

$u_{02}$

$v_{11}$

$v_{12}$

21

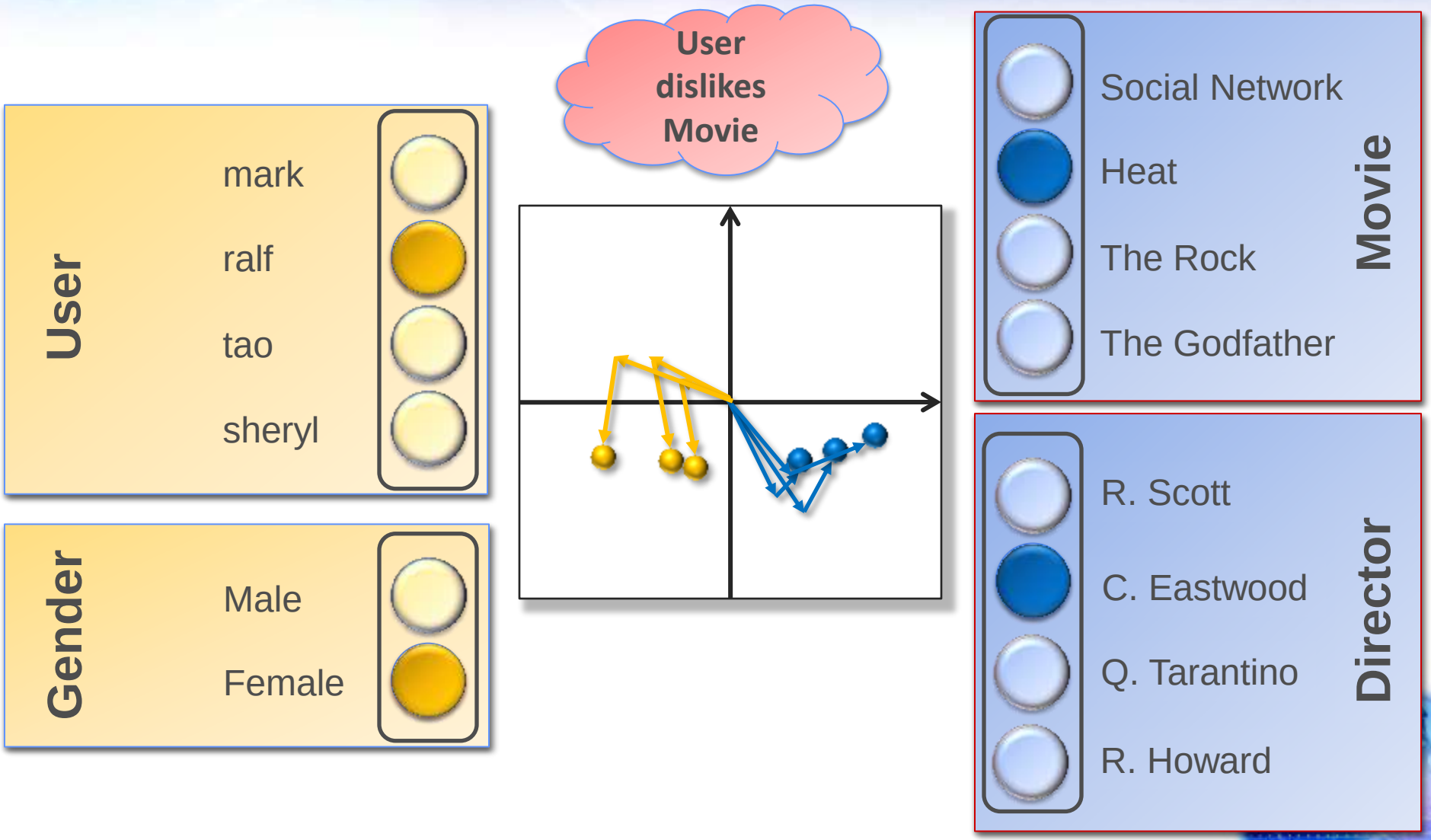
22

User $\mathbf{s} = \mathbf{U}\mathbf{x}$

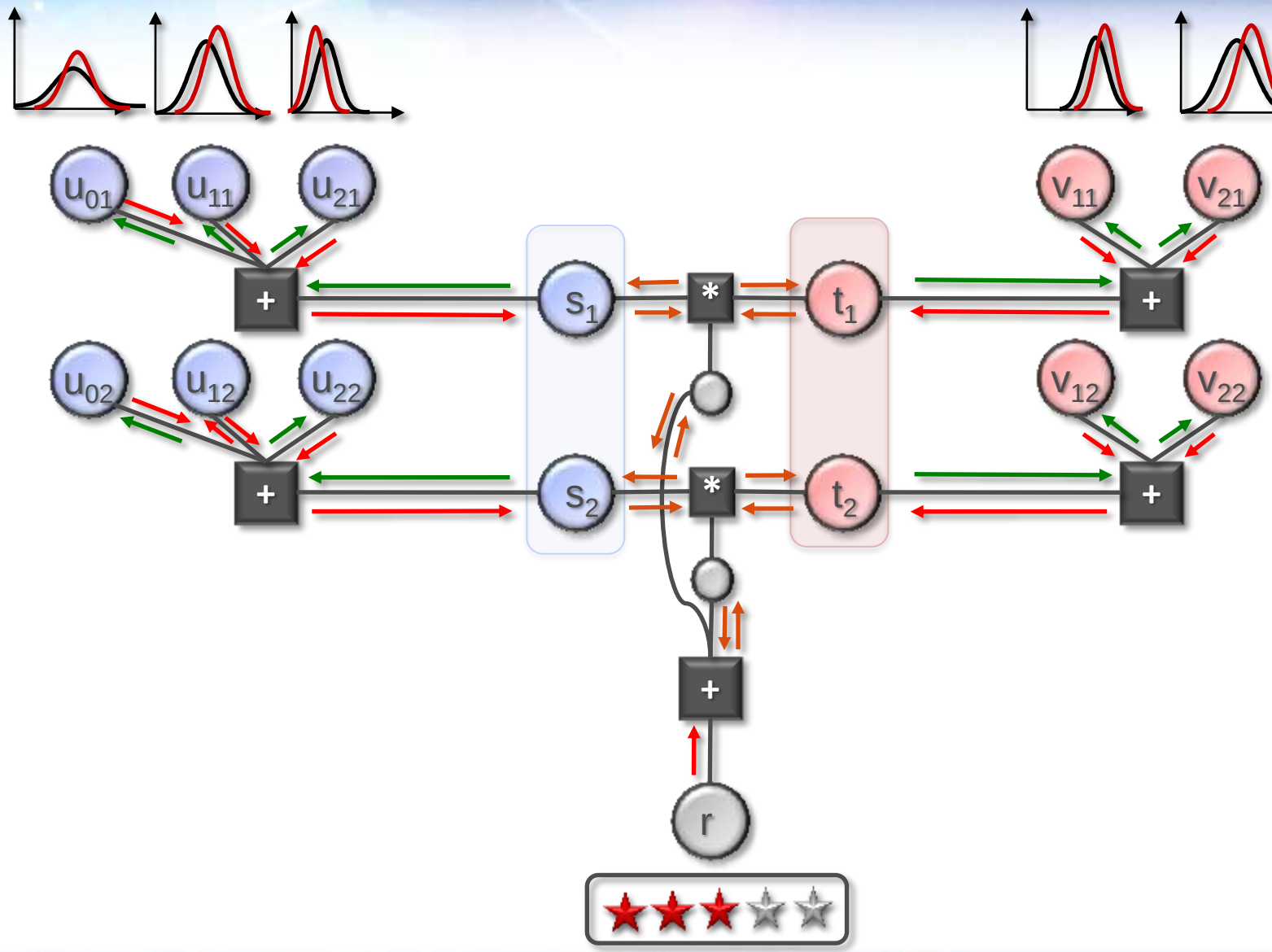
Item $\mathbf{t} = \mathbf{V}\mathbf{y}$

Rating potential ~ $\mathcal{N}(\mathbf{s}^\top \mathbf{t}, \beta^2)$

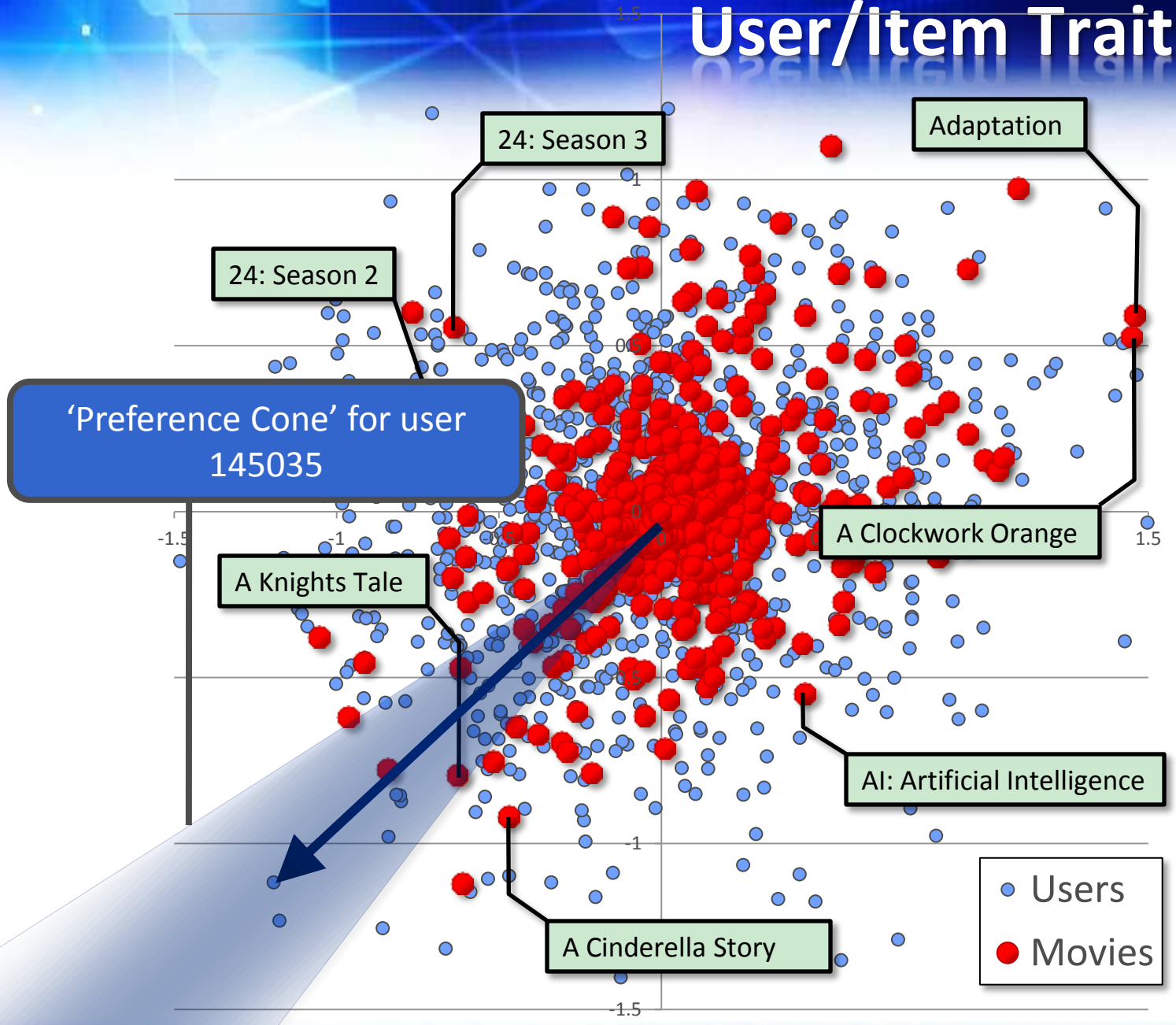
# Recommender System: MatchBox



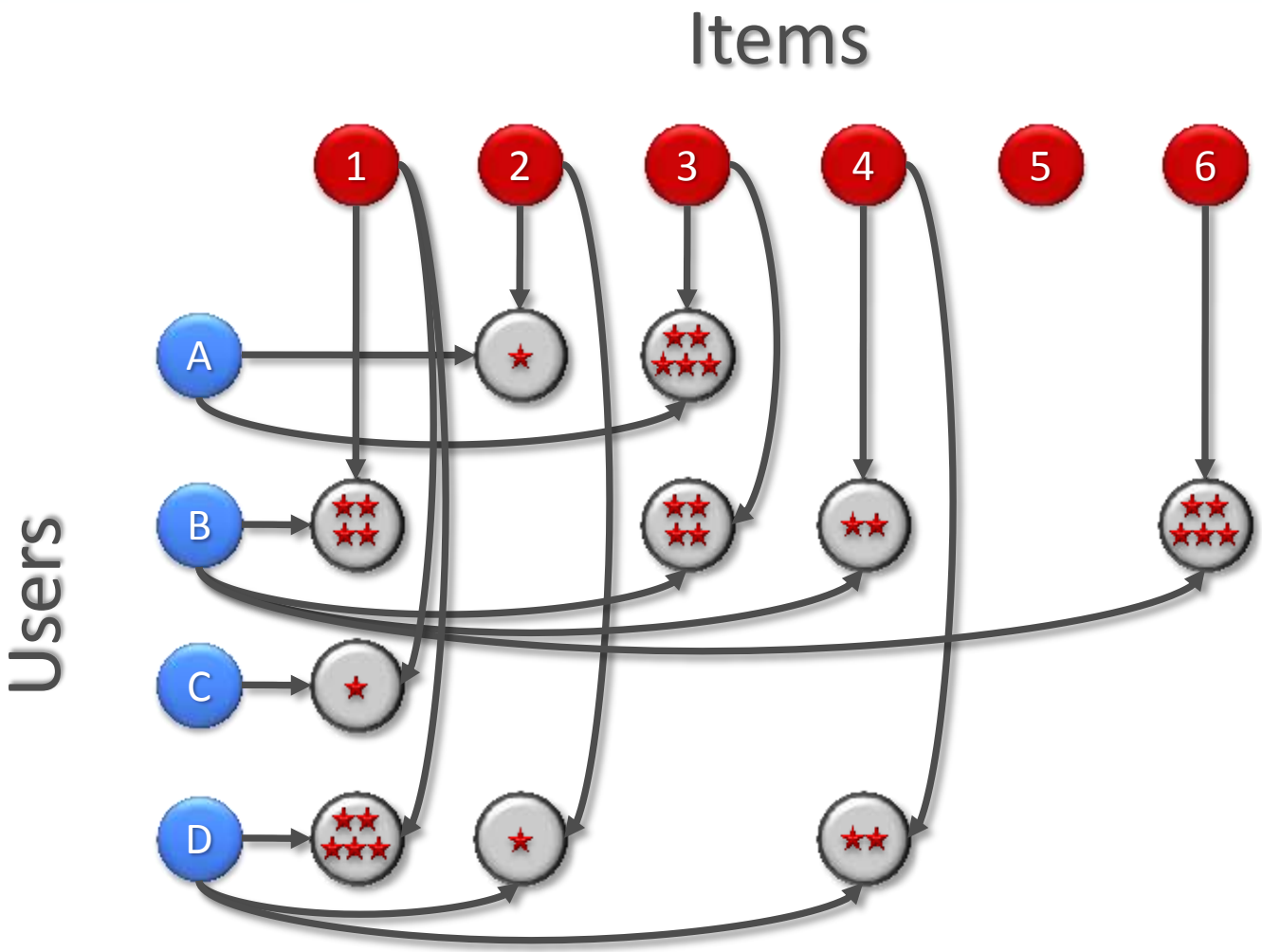
# Message Passing For Matchbox



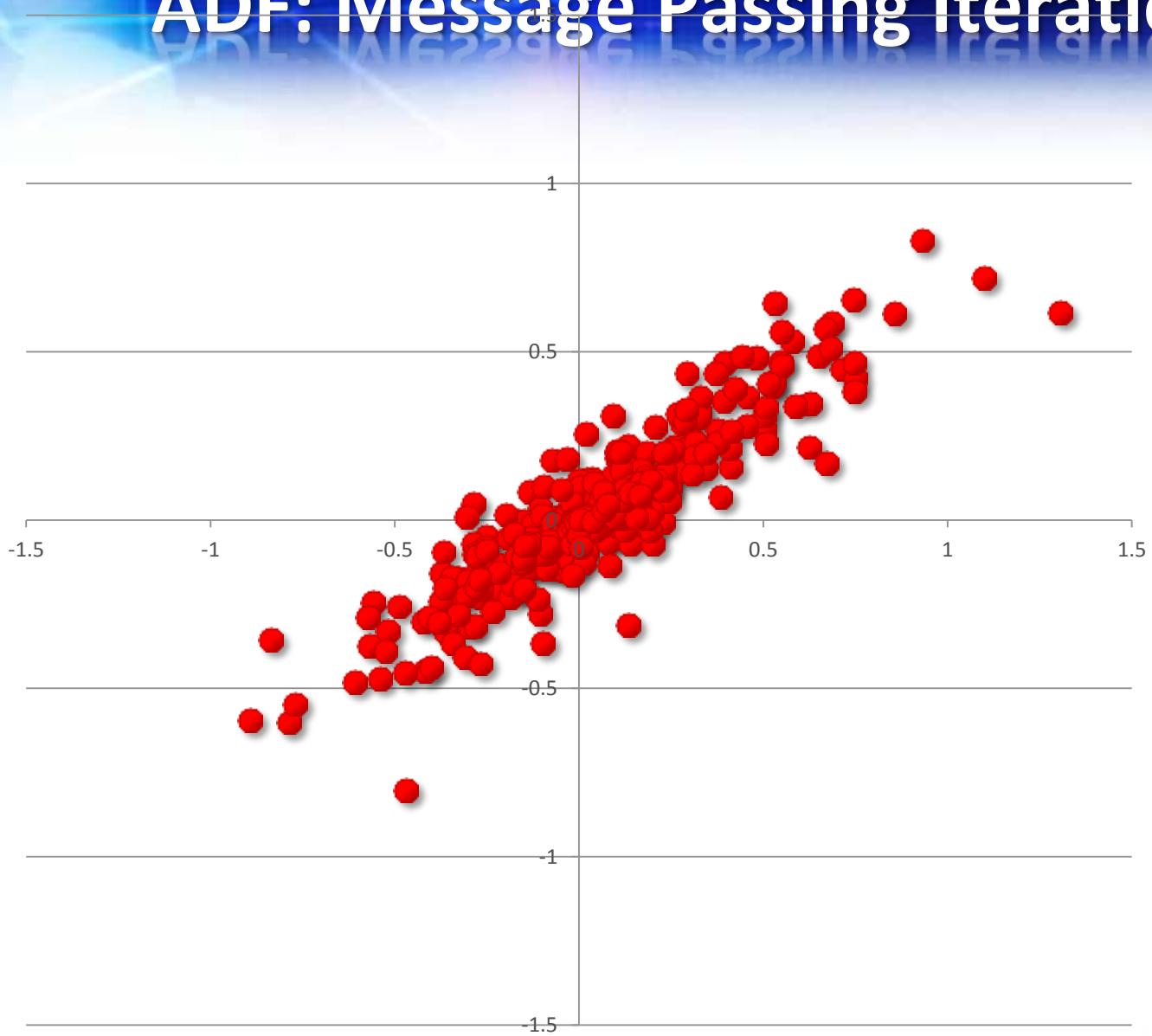
# User/Item Trait Space



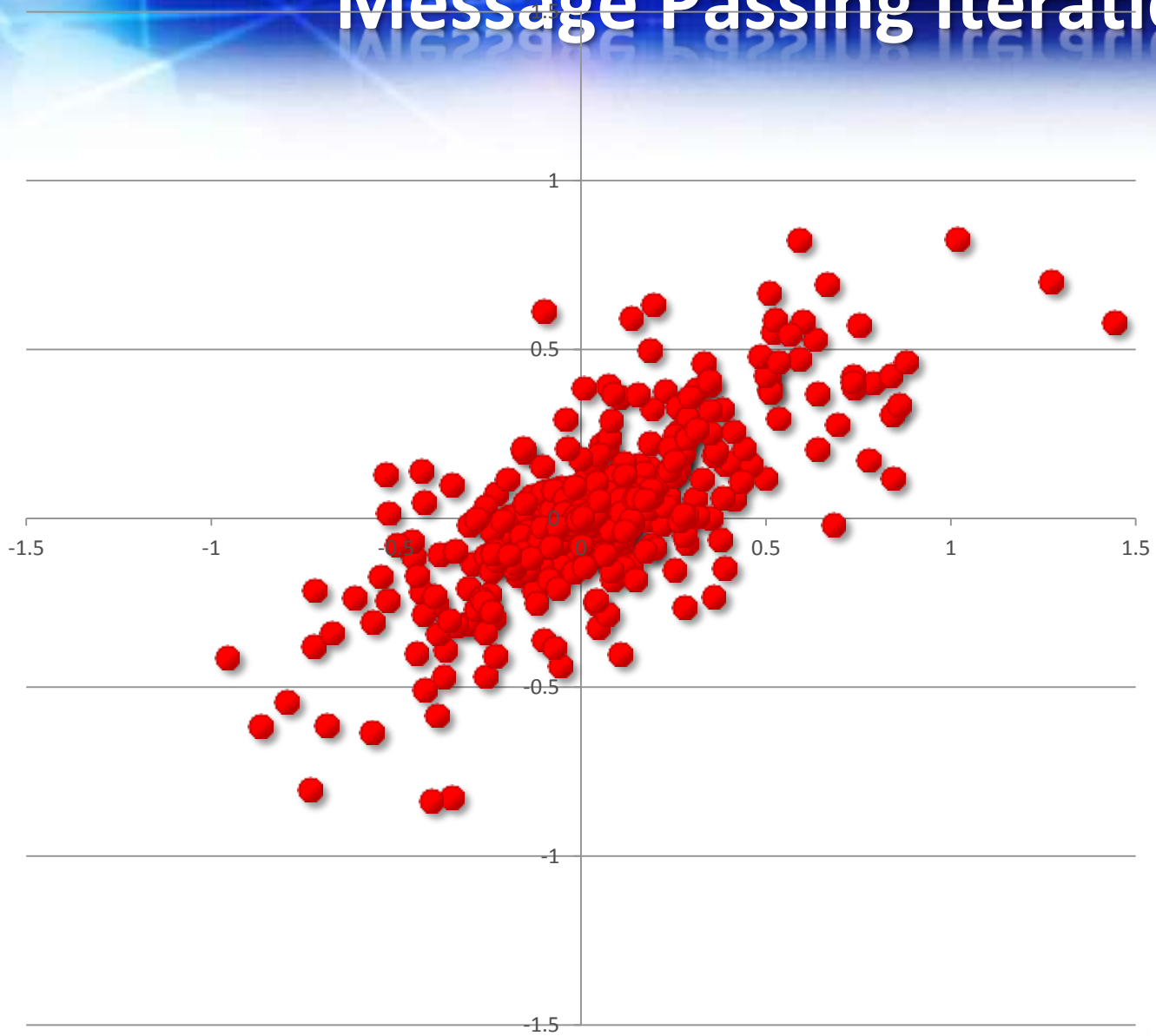
# Incremental Training with ADF



# ADF: Message Passing Iteration 1



# Message Passing Iteration 2

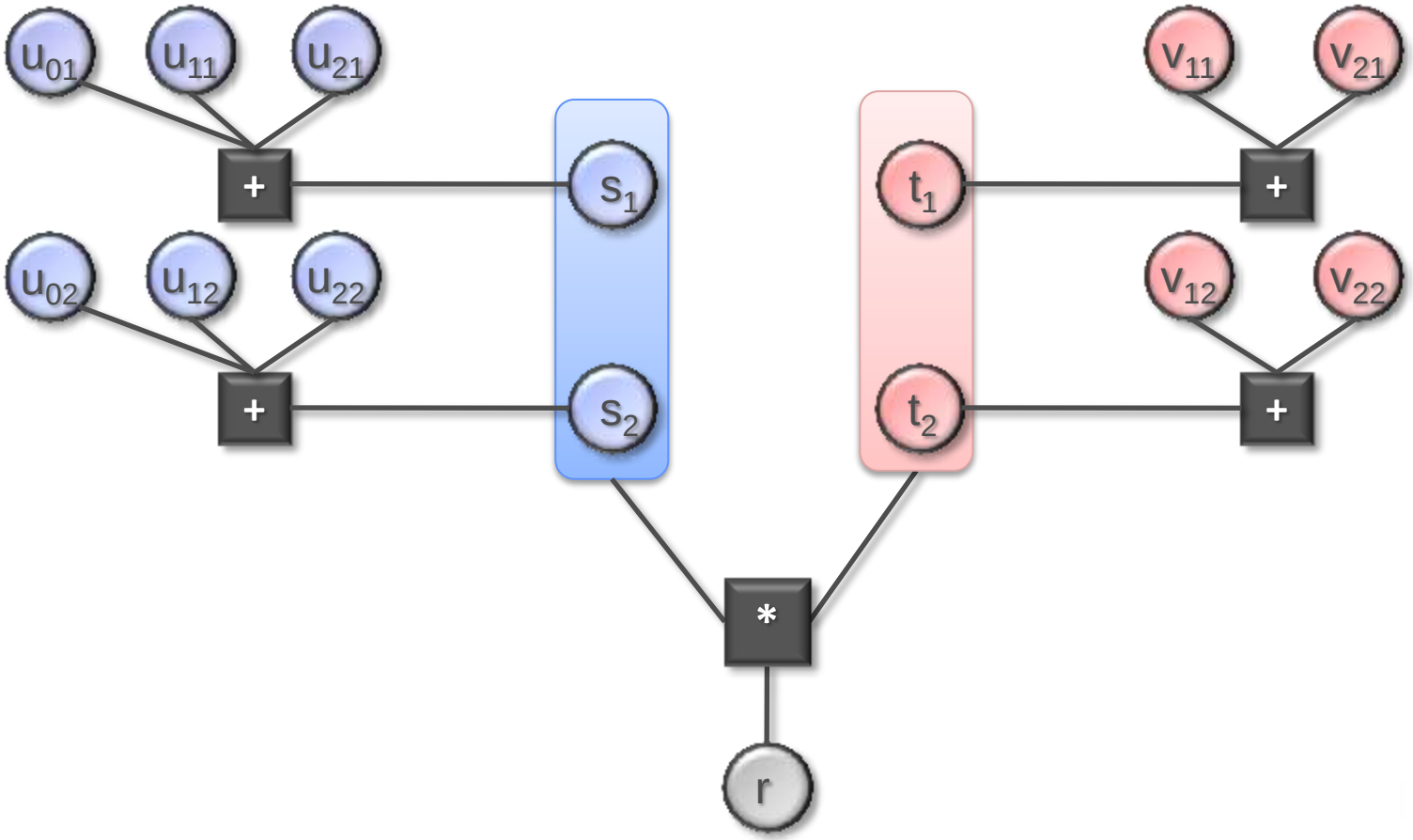


# Message Passing Iteration 3

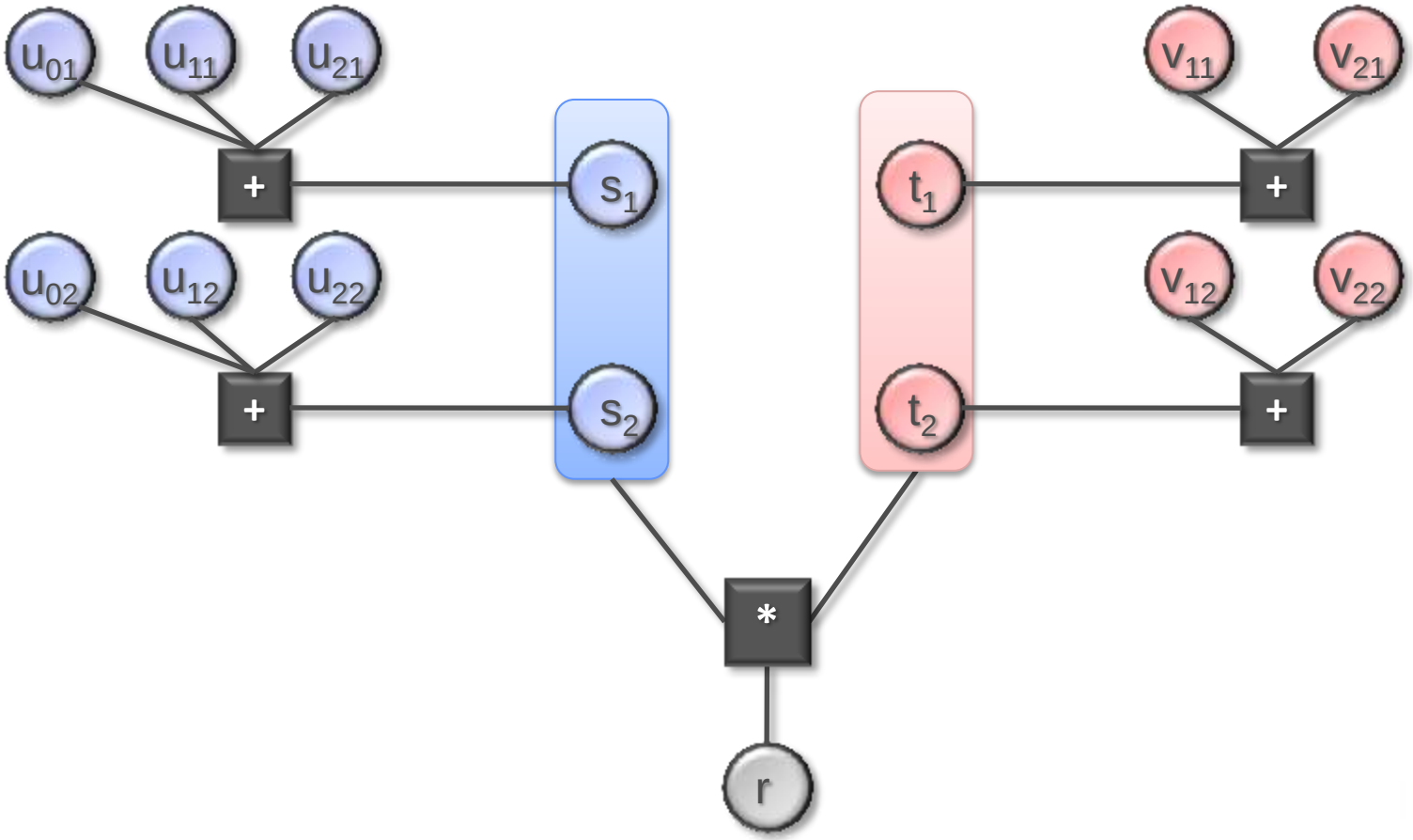
# Message Passing Iteration 4

# feedback models

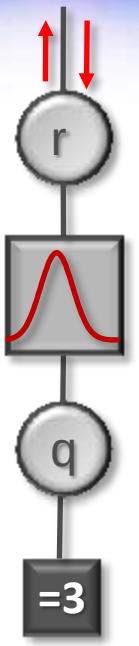
# Feedback Models



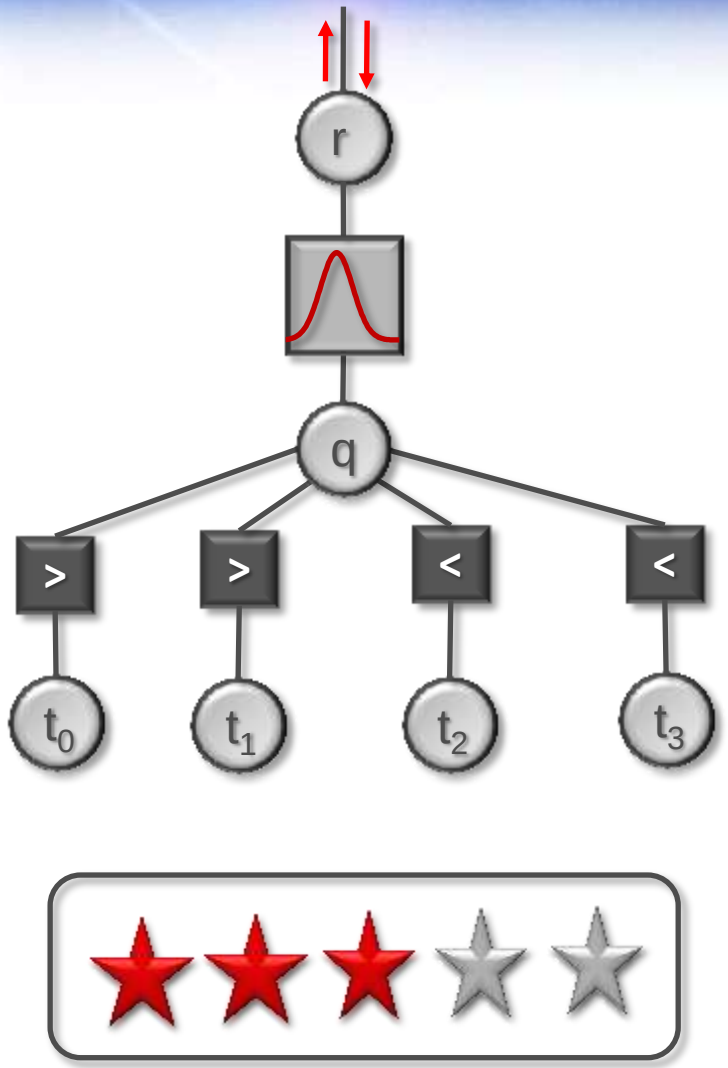
# Feedback Models



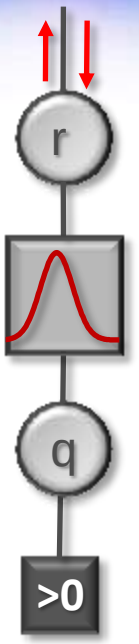
# Feedback Models



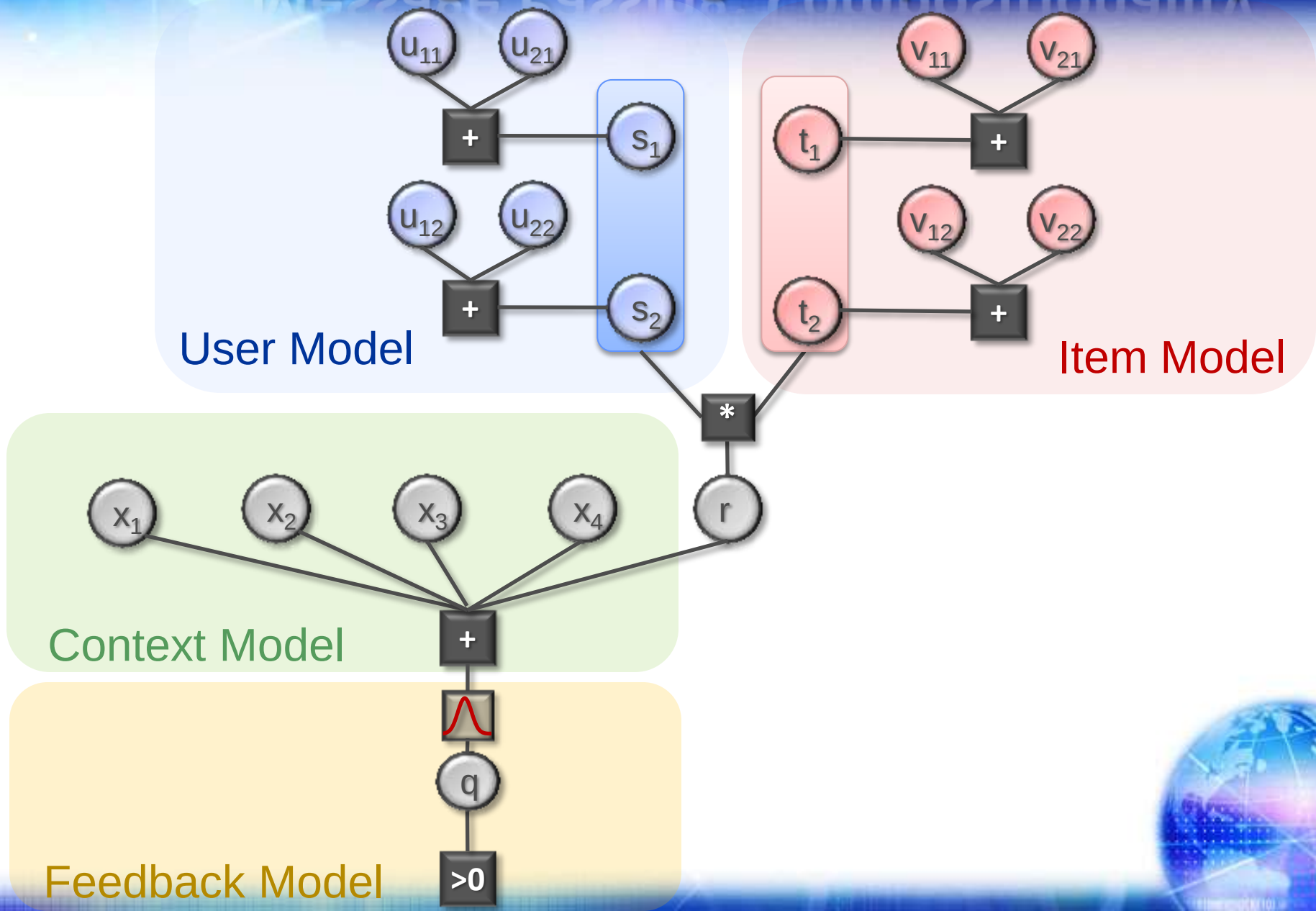
# Feedback Models



# Feedback Models



# Message Passing: Compositionality



# 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

| User ID |
|---|

| User Job | |
|---|---|
| Other | Lawyer |
| Academic | Programmer |
| Artist | Retired |
| Admin | Sales |
| Student | Scientist |
| Customer Service | Self-Employed |
| Health Care | Technician |
| Managerial | Craftsman |
| Farmer | Unemployed |
| Homemaker | Writer |

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

| User Gender |
|---|
| Male |
| Female |

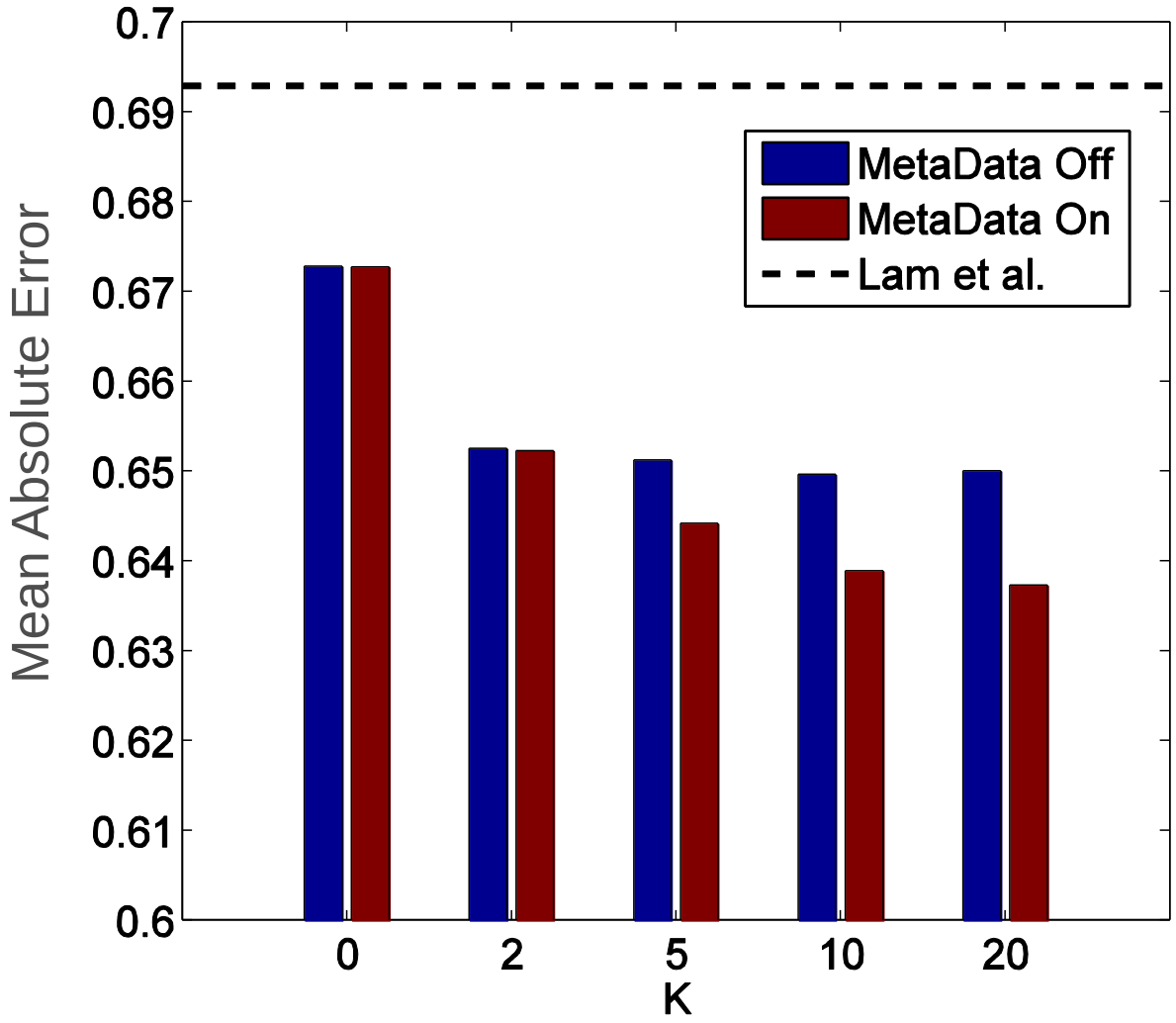
## 3,900 movies

| Movie ID |
|---|

| Movie Genre | |
|---|---|
| Action | Horror |
| Adventure | Musical |
| Animation | Mystery |
| Children's | Romance |
| Comedy | Thriller |
| Crime | Sci-Fi |
| Documentary | War |
| Drama | Western |
| Fantasy | Film Noir |

# MovieLens with Thresholds Model

(ADF), Training Time= 1 Minute



# 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:

$$P(h(x) = h(y)) = \text{sim}(x, y)$$

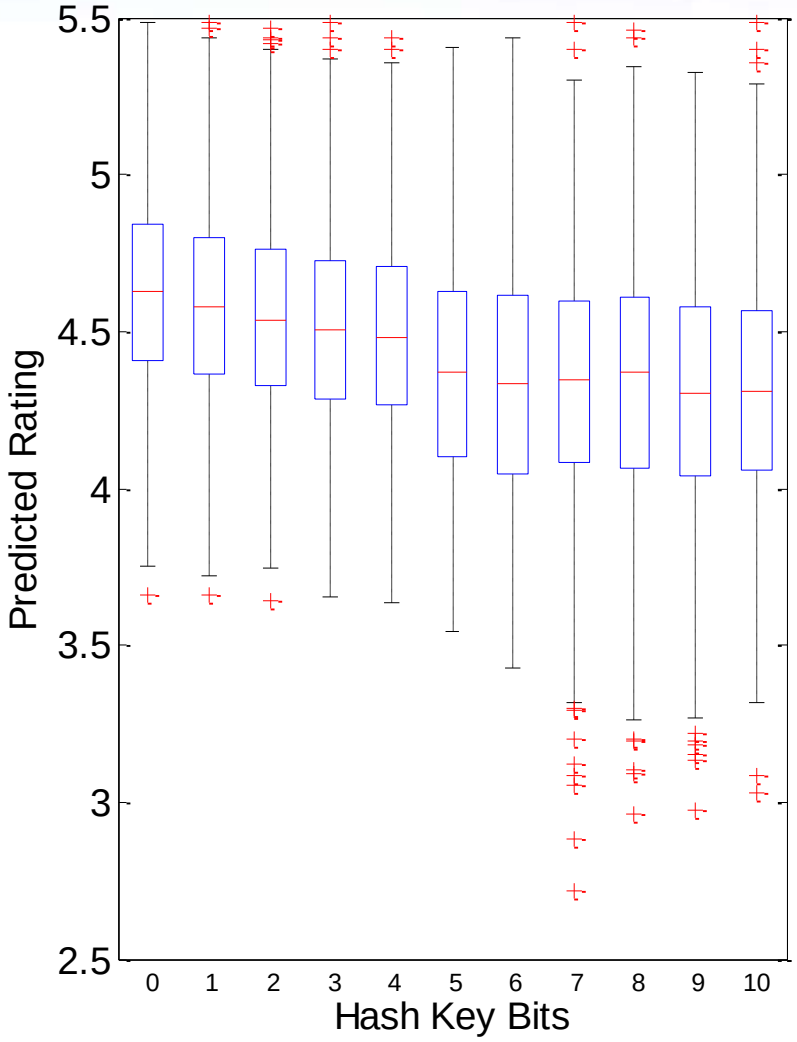
# Random Projection Hashing

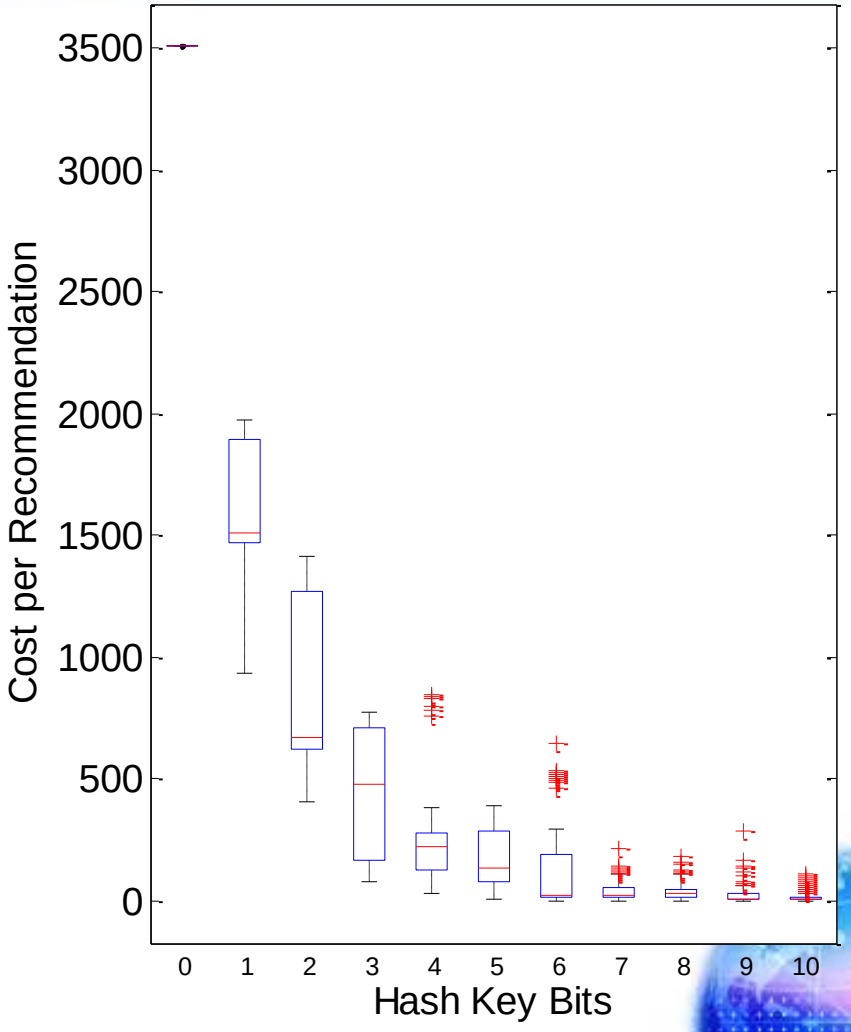
- Random Projections:
  - Generate random hyper planes (m random vectors, $a_i$).
  - Gives m bit hash, $\{x_0, x_1, \cdots, x_m\}$ , by:

$$x_i = 1[\mathbf{a}_i \cdot \mathbf{t} > 0]$$

- p(all bits match) ∝ 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



# Learning to Play Go

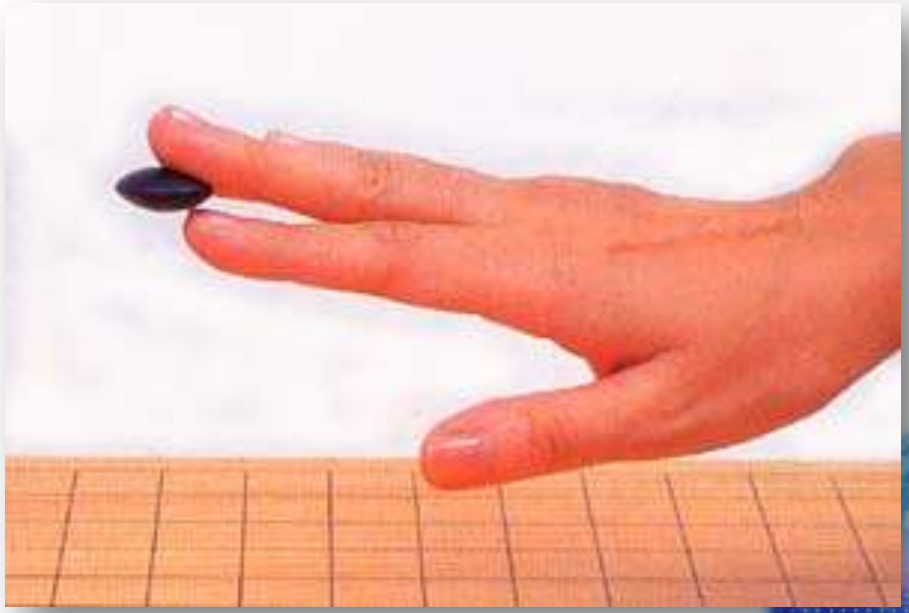
Joint work with Thore Graepel & David Stern

# Uncertainty in Go

- Go is game of perfect information.
- Complexity of game tree +
  limited computer speed → uncertainty.
- 味 ‘aji’ = ‘taste’.
- Our Approach:
  Represent uncertainty using probabilities.

# Machine Learning

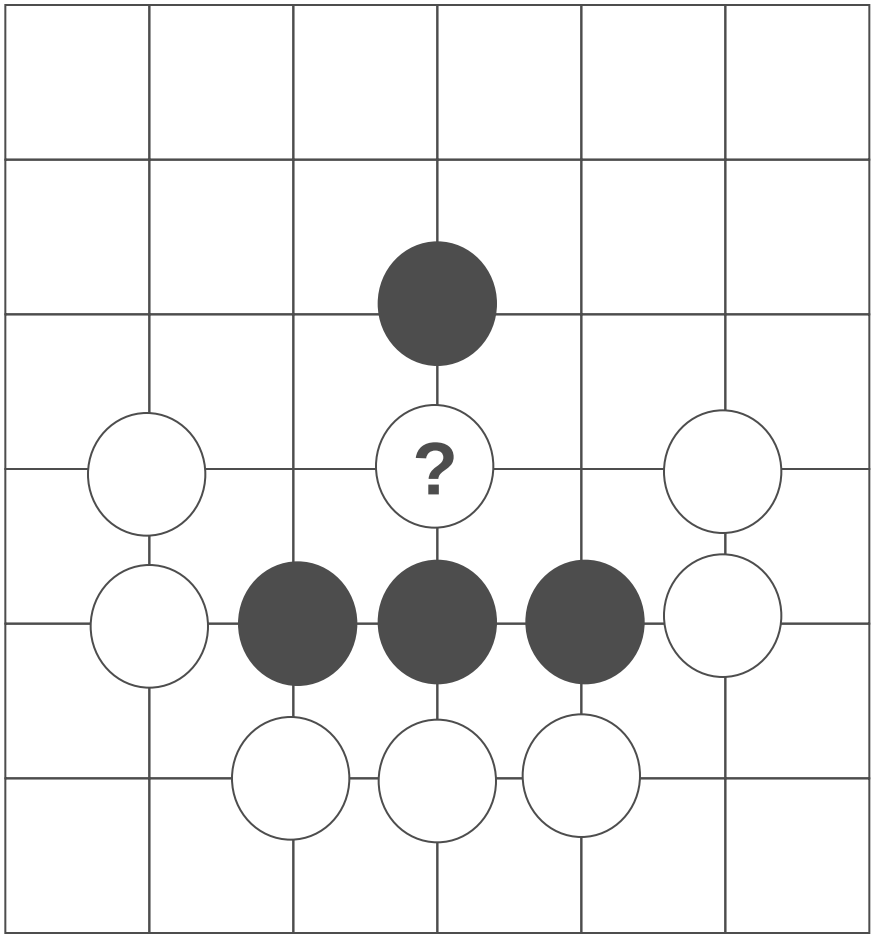
- Automatic knowledge Acquisition.
- Principled management of uncertainty.
- Applications to Go:
  - **Move Prediction.**
  - **Tactical Search.**
  - **Territory Prediction.**
  - **Monte Carlo Go.**



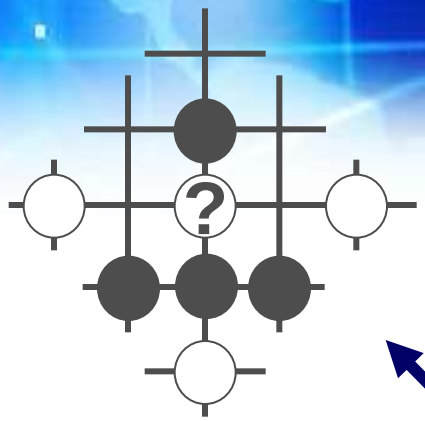
# Move Prediction

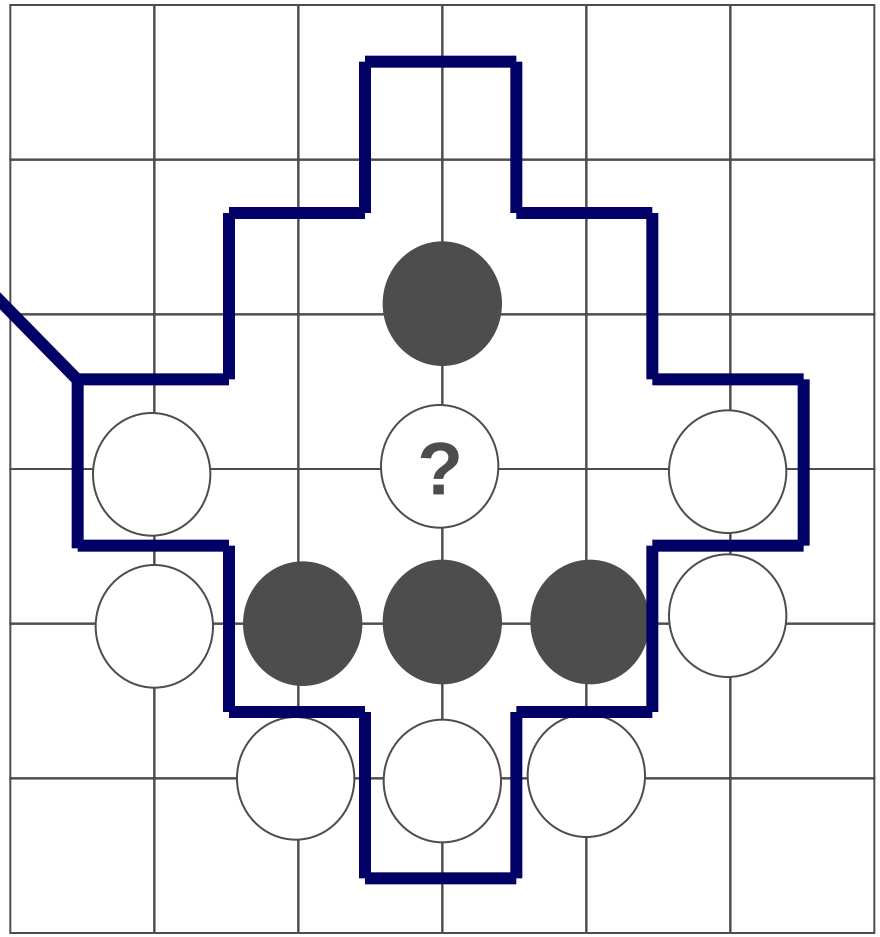
- Learning from Expert Game Records
- Move associated with a set of patterns.
  - **Exact arrangement of stones.**
  - **Centred on proposed move.**
- Sequence of nested templates.
- Inspired by work by David Stoutamire and Frank de Groot

# Patterns

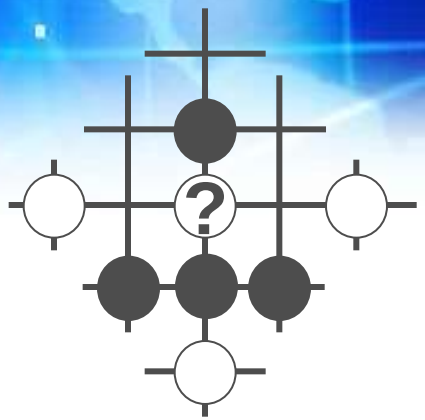


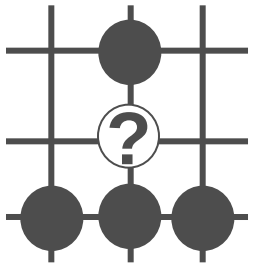
# Patterns

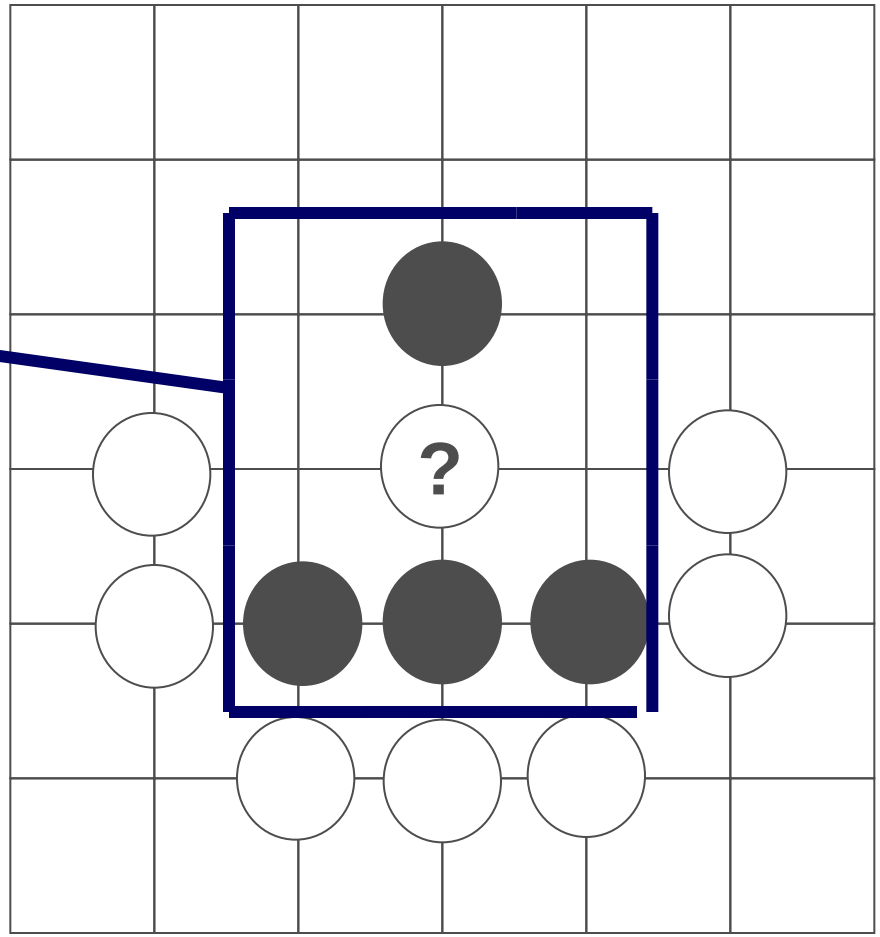


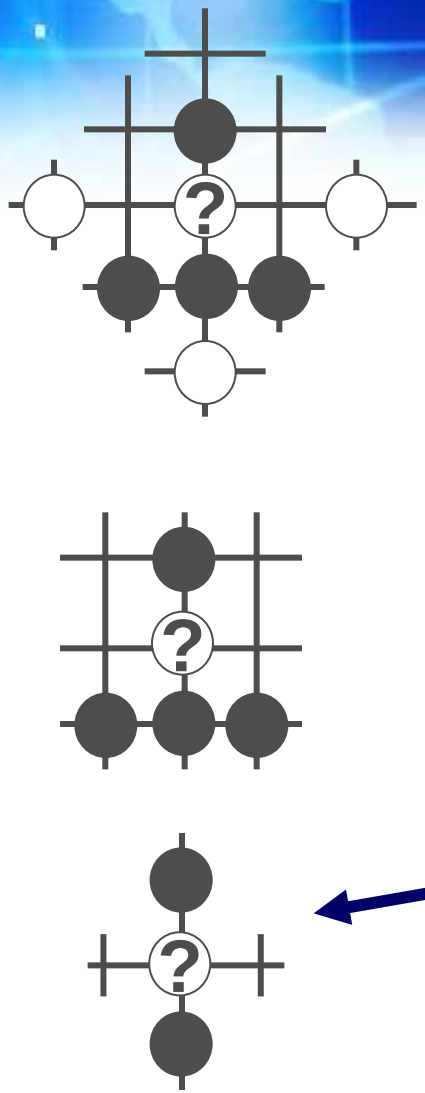
# Patterns



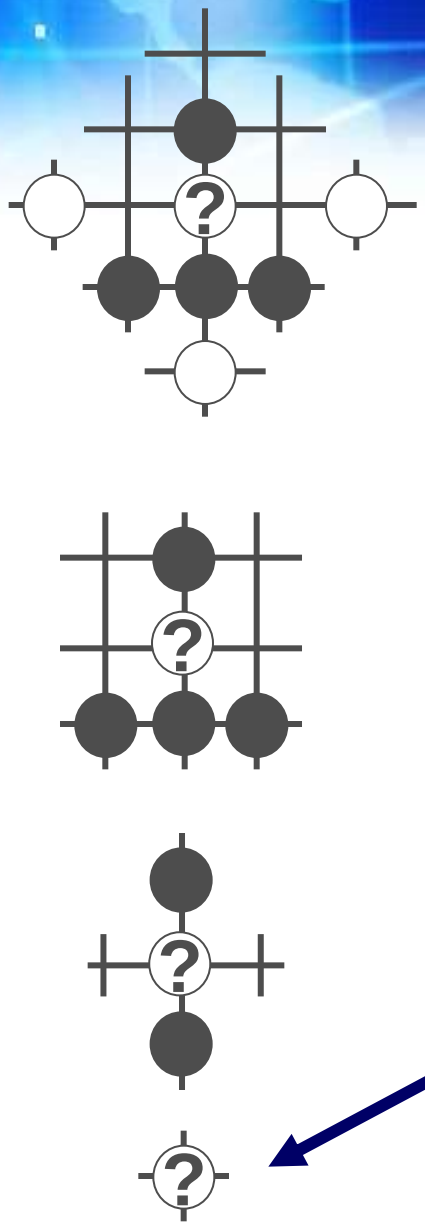



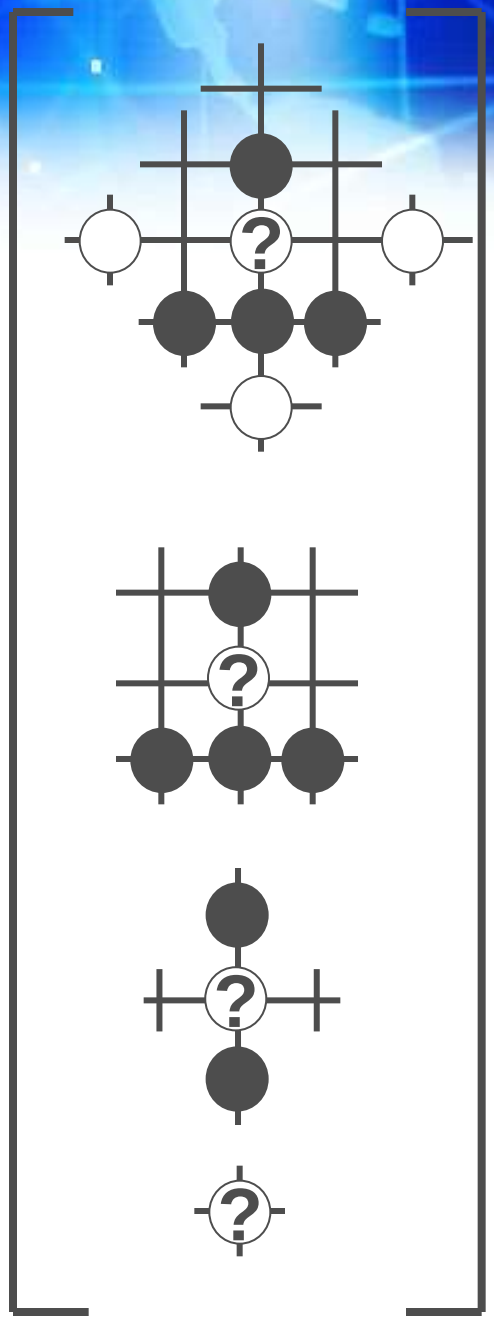
# Patterns

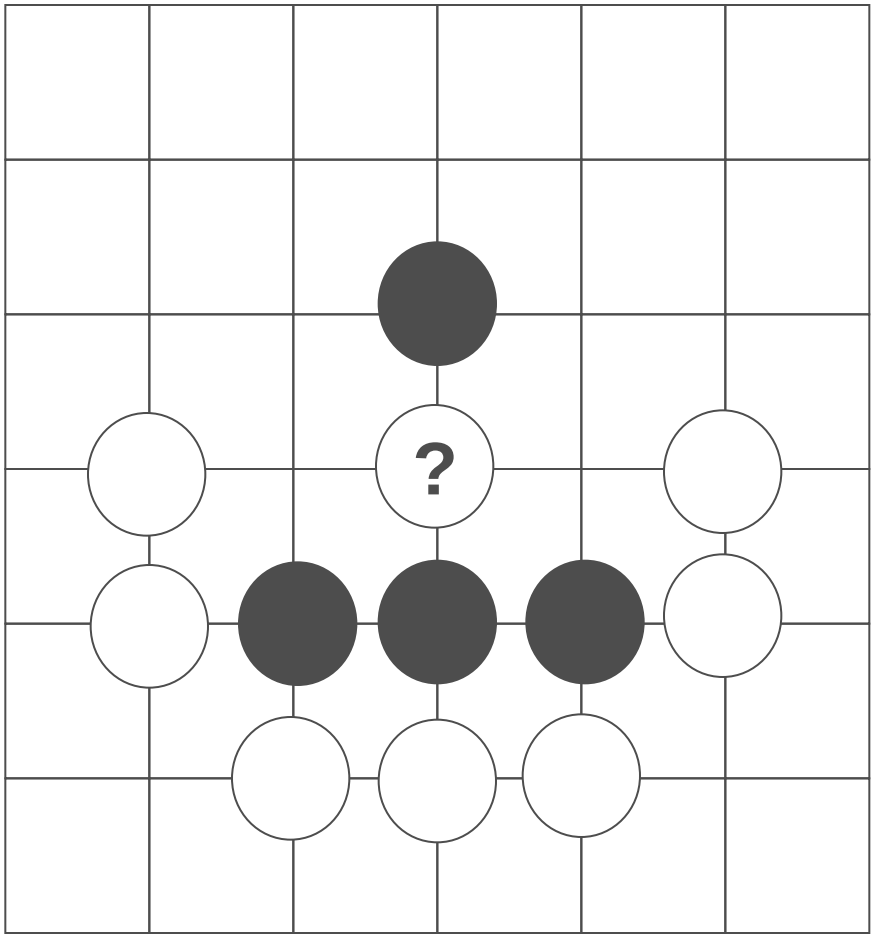


# Patterns

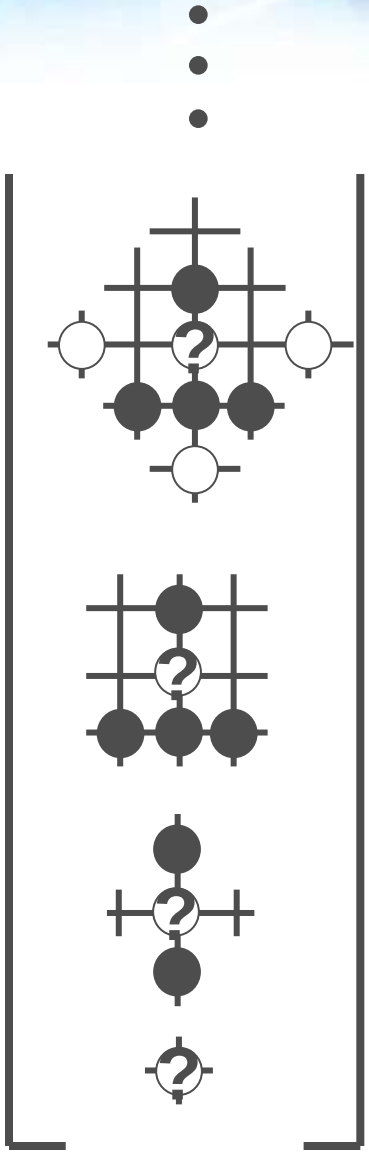


# Patterns



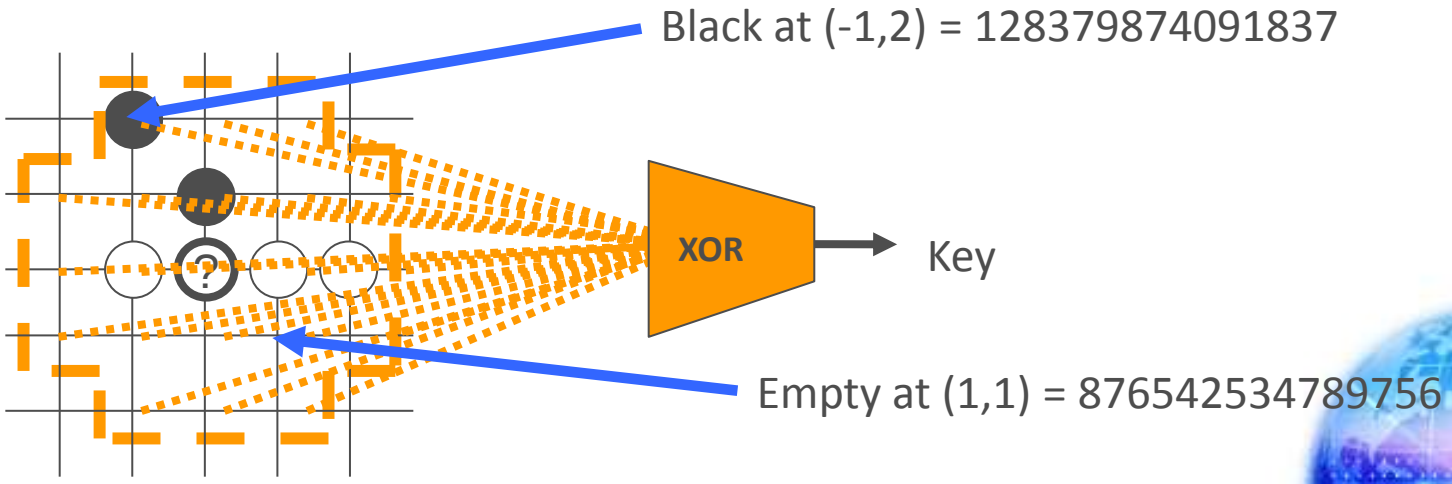
# Patterns



- 13 Pattern Sizes
  - **Smallest is vertex only.**
  - **Biggest is full board.**

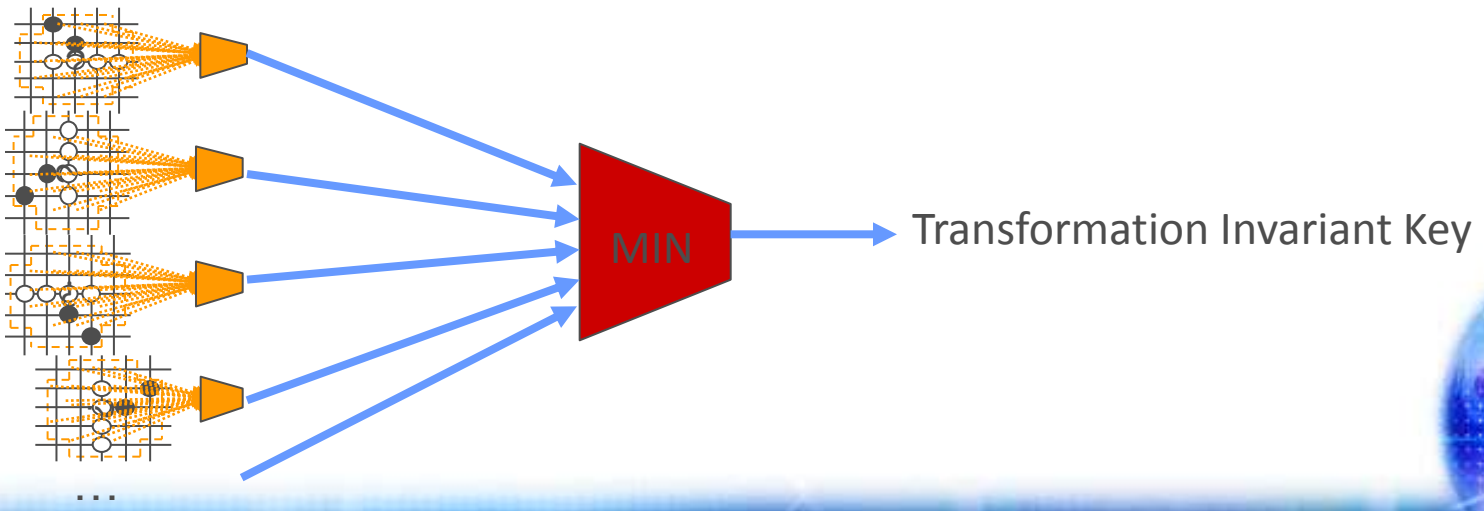
# Pattern Matching

- **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).



# Pattern Hash Key

- **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).



# Harvesting

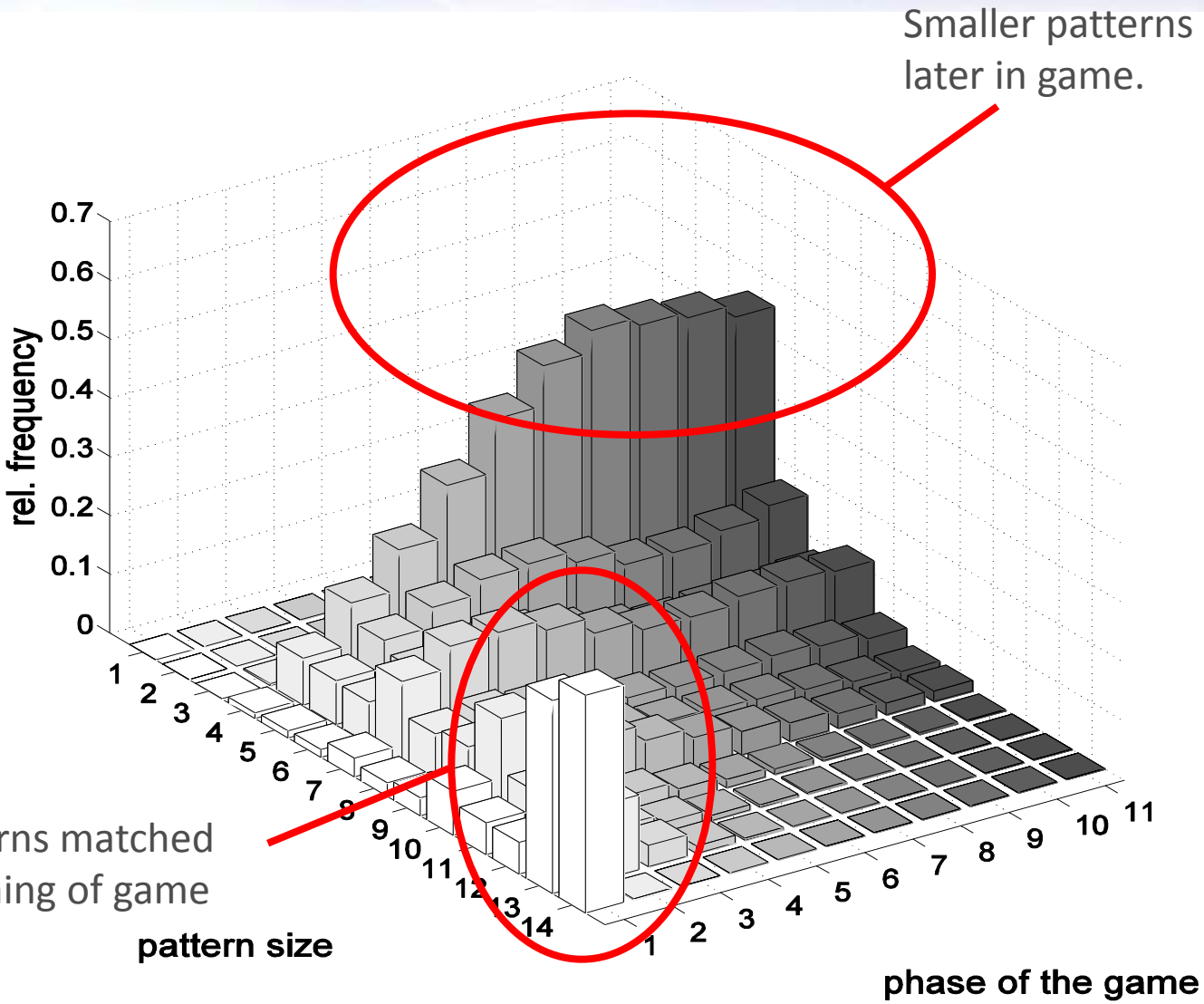
- **Data Size**: 180,000 games × 250 moves × 13 pattern sizes…

  …gives **600 million potential patterns**

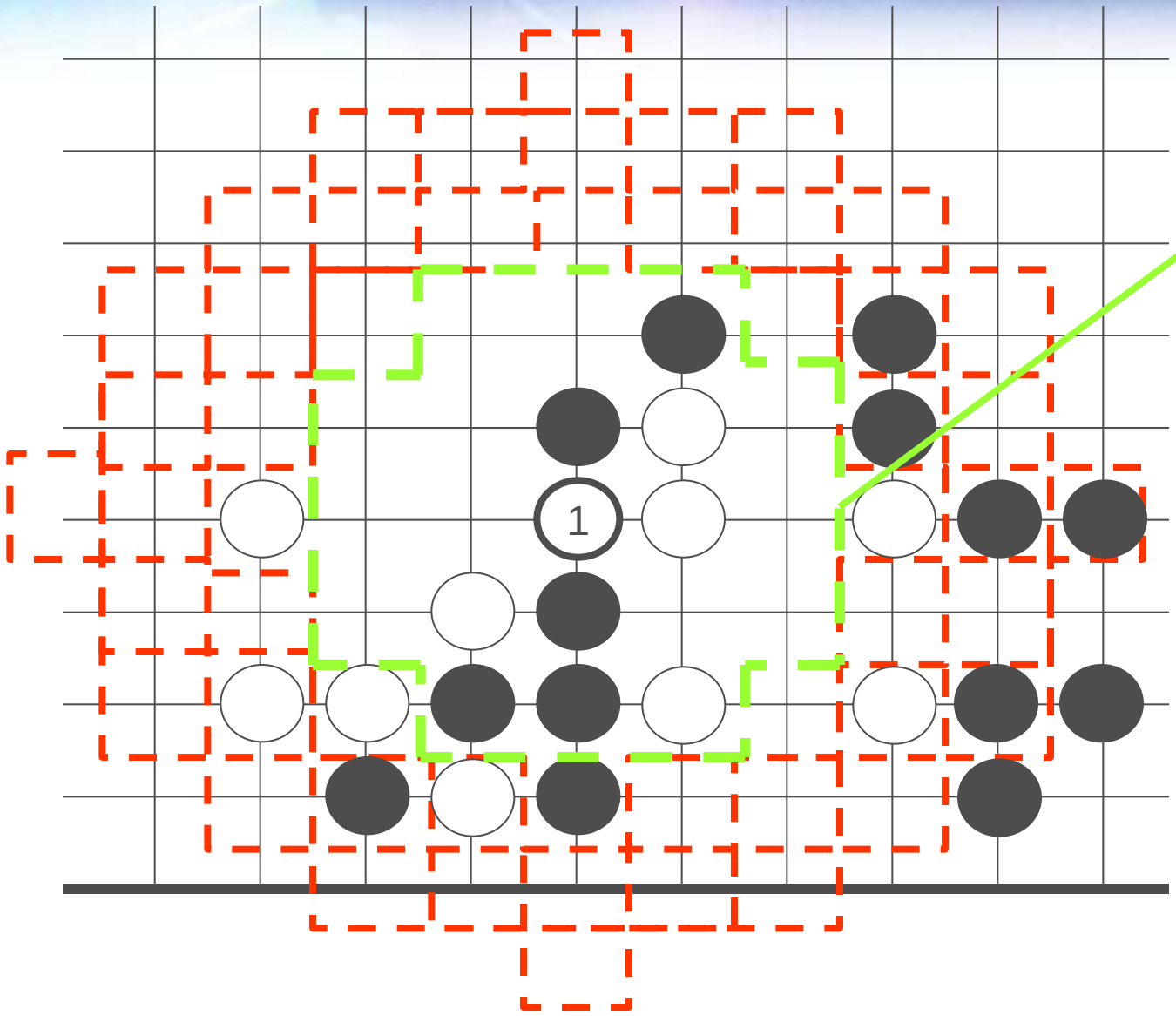
- **Problem**: Need to limit number stored.

- **Idea**: Keep patterns played more than n times.
- **Bloom filter**: Approximate test for set membership with minimal memory footprint.

# Relative Frequencies of Pattern Sizes



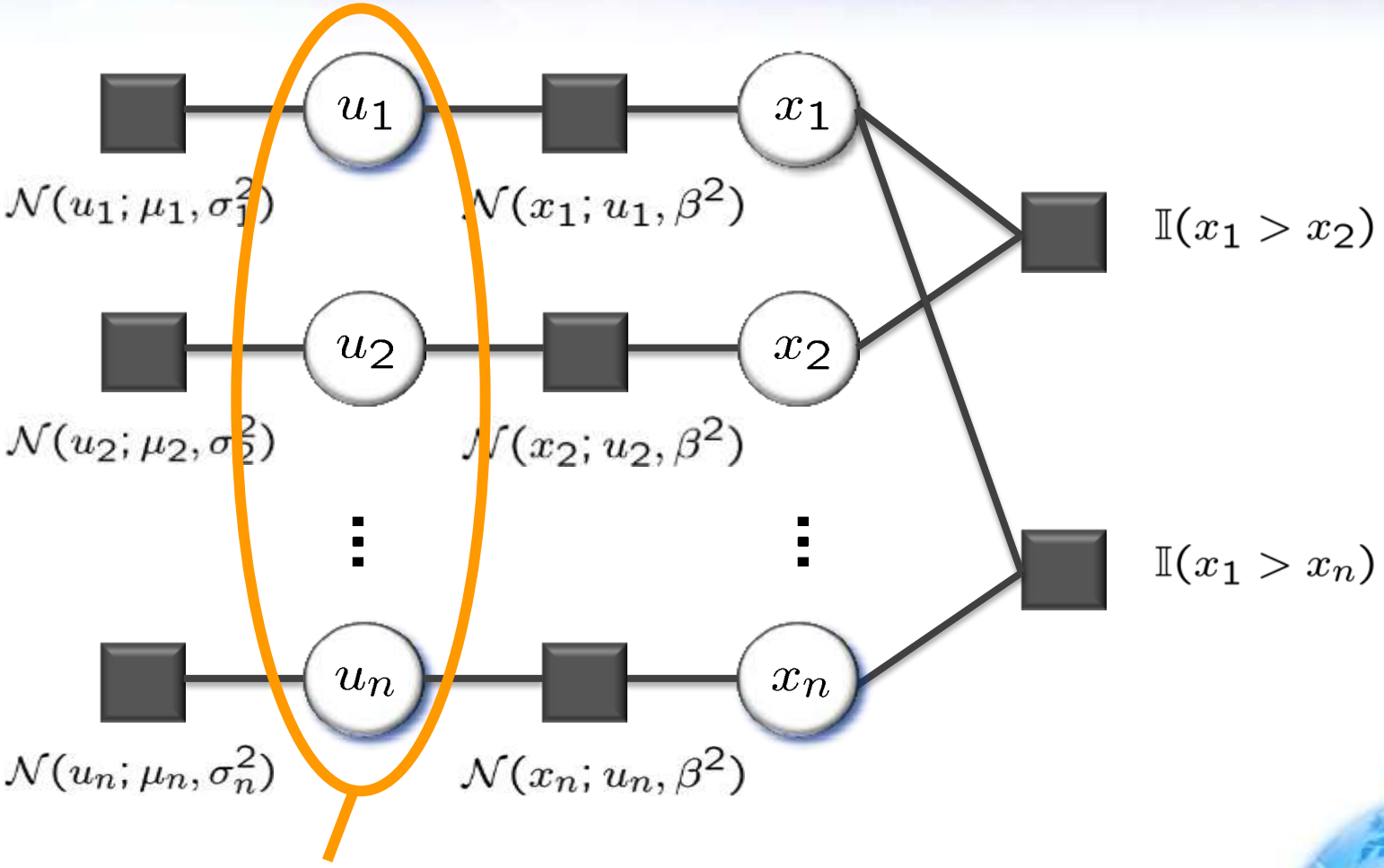
# Move: Biggest Pattern




Table

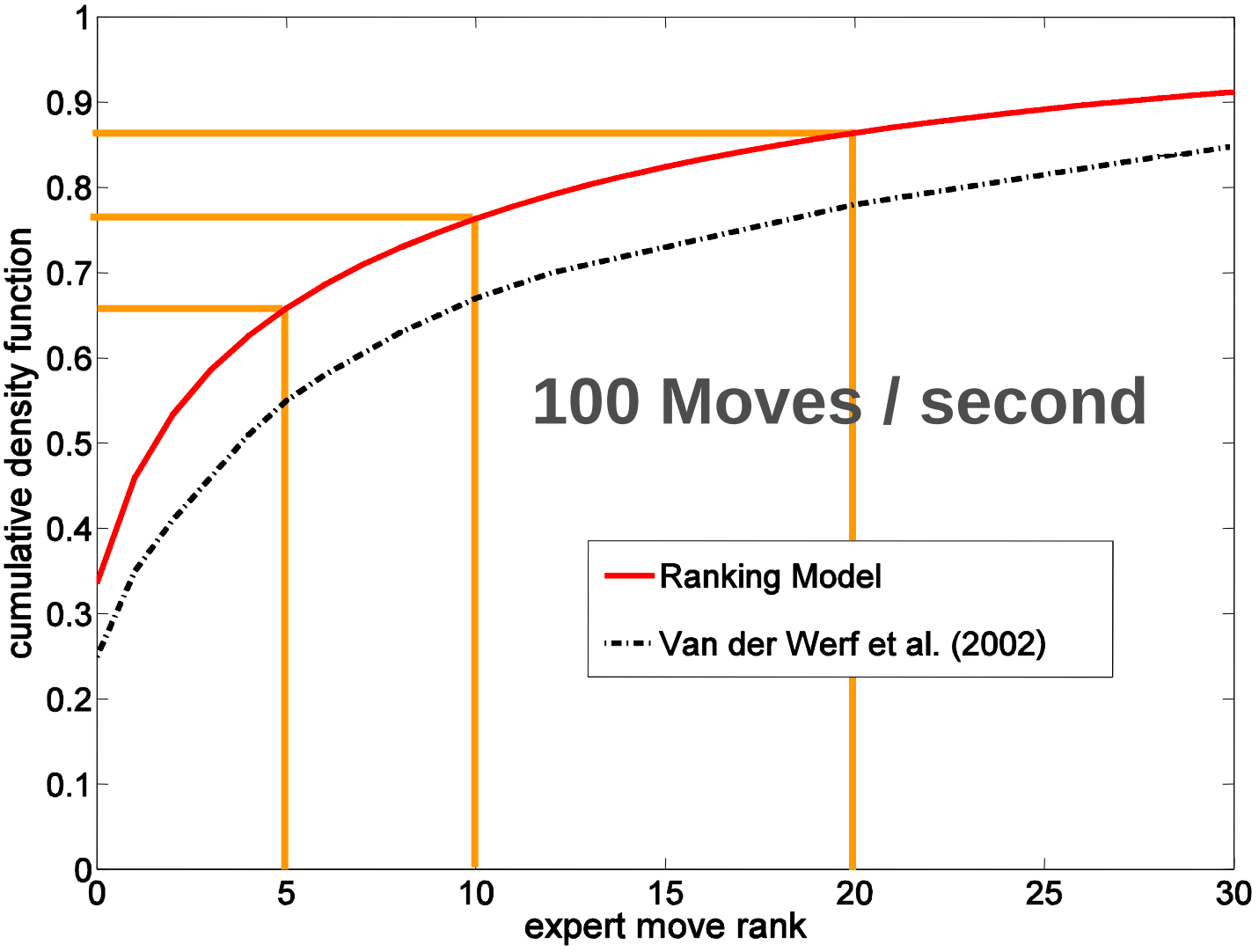
Skill Mu=4.5, Sigma=0.2

# Bayesian Ranking Model

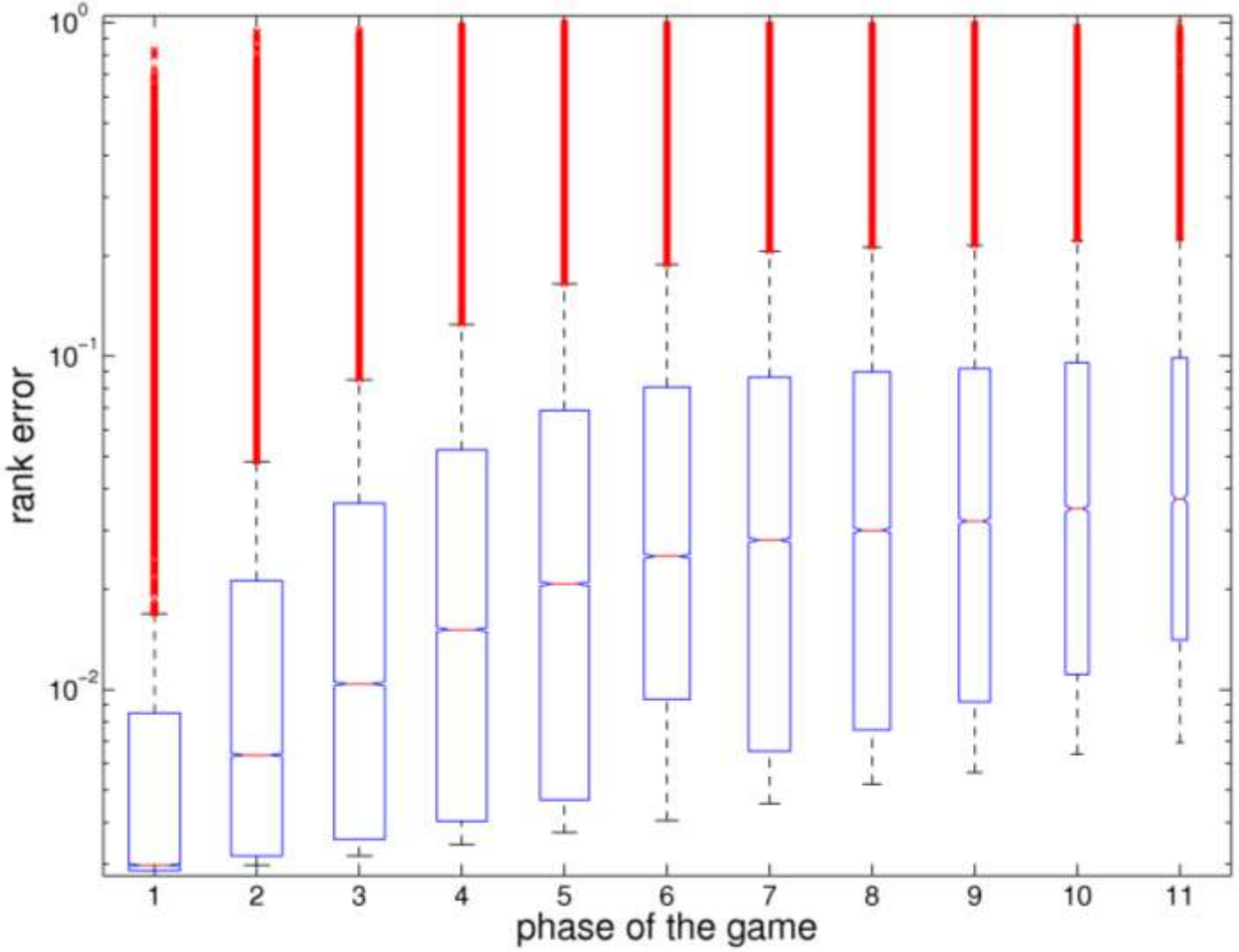



$$p(\mathbf{u}|\text{move}, \text{position}) = \int p(\mathbf{u}, \mathbf{x}|\text{move}, \text{position}) d\mathbf{x}$$

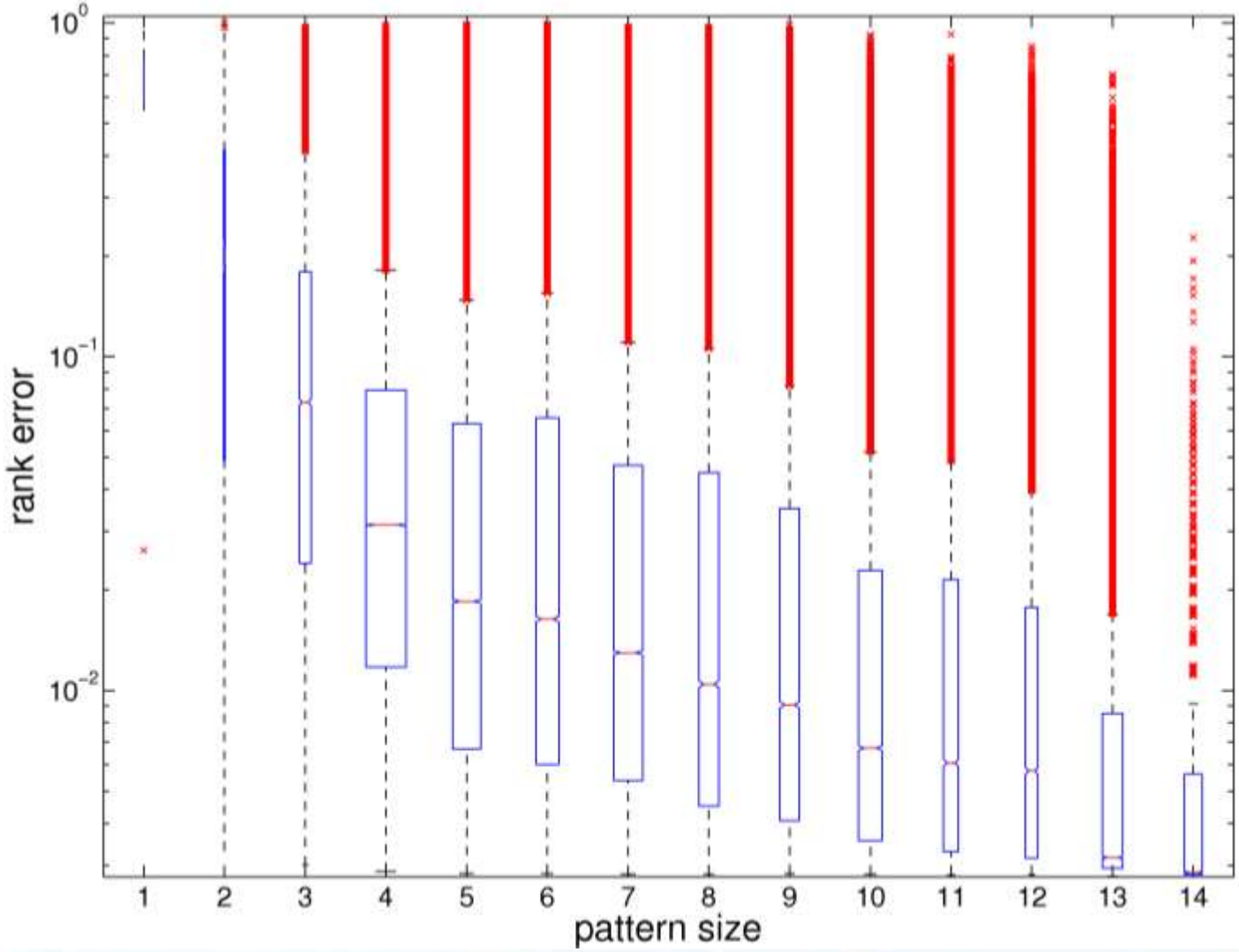
# Move Prediction Performance



# Rank Error vs Game Phase



# Rank Error vs Pattern Size



# Conclusion

- 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!