Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Probabilistic Machine Learning

Graphical Models: Bayesian Networks

Ralf Herbrich

Graphical Models

- **Challenge**: How to formulate complex likelihoods/data models & priors for *actual* data?
  - **Example 1**: Match outcomes $y \in \{-1,1\}$ for a head-to-head match between two players
    - **Prior**: $p(\boldsymbol{s}) = \mathcal{N}(s_1; \mu_1, \sigma_1^2) \cdot \mathcal{N}(s_2; \mu_2, \sigma_2^2)$ skill belief
    - **Likelihood**: $p(y|\boldsymbol{s}) = \int \mathcal{N}(p_1; s_1, \beta^2) \cdot \mathcal{N}(p_2; s_2, \beta^2) \cdot \mathbb{I}(y(p_1 - p_2) > 0) \, dp_1 \, dp_2$ marginalization

Match outcome

Player performance

  - **Example 2**: Time series $\boldsymbol{y}$ of temperatures given
    - **Prior**: $p(w) = \mathcal{N}(w; \mu, \sigma^2)$ External state mapping parameter belief
    - **Likelihood**: $p(\boldsymbol{y}|w, X) = \int \mathcal{N}(z_1;\ w \cdot x_1, \tau^2) \cdot \mathcal{N}(y_1; z_1, \beta^2) \cdot \mathcal{N}(z_2; z_1 + w \cdot x_2, \tau^2) \cdots \, d\boldsymbol{z}$ marginalization

Dynamics model

Observed temperature model

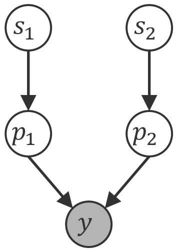
Conditional hidden state model

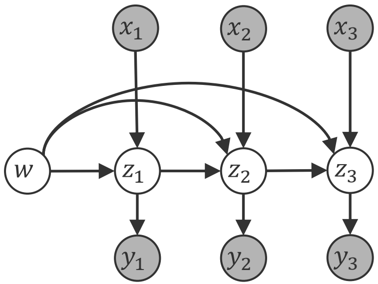
Graphical Models

- **Observation**: The product structure of the probabilities seems crucial!
- **Idea**: Define a graph where each of the variables are nodes and edges indicate relationships between variables

$$\mathcal{N}(s_1;\mu_1,\sigma_1^2)\cdot\mathcal{N}(s_2;\mu_2,\sigma_2^2)\cdot\mathcal{N}(p_1;s_1,\beta^2)\cdot\mathcal{N}(p_2;s_2,\beta^2)\cdot\mathbb{I}(y(p_1-p_2)>0)$$

$$\mathcal{N}(w;\mu,\sigma^2)\cdot\mathcal{N}(z_1;\,w\cdot x_1,\tau^2)\cdot\mathcal{N}(y_1;z_1,\beta^2)\cdot\mathcal{N}(z_2;z_1+w\cdot x_2,\tau^2)\cdot\mathcal{N}(y_2;z_2,\beta^2)\cdots$$





- **Advantages**: Simple way to visualize factor structure of the joint probability
  - **Bayesian Networks**: Insights into (conditional) independence based on graph properties
  - **Factor Graphs**: Insights into efficient inference and approximation algorithms

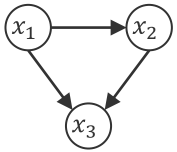
Bayesian Networks

- **Observation**. Any joint distribution $p(x_1, \ldots, x_n)$ can be written as

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

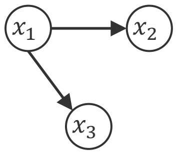
- **Bayesian Network**. *Given a joint distribution as a product of conditional distributions,* $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | \text{parents}_i)$, *a* Bayesian network *is a graph with a node for every variable* x_i, *and a directed from every variable* $x \in \text{parent}_i$ to x_i. *If the variable is independent of all other variables, it has no incoming edges.*

- **Examples**: For 3 variables, we have these four generic Bayesian networks



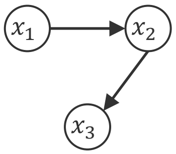
$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2)$

full mesh



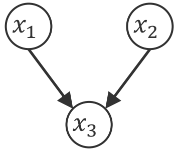
$p(x_1, x_2, x_3) = p(x_1) \cdot \prod_{i=2}^{3} p(x_i|x_1)$

star



$p(x_1, x_2, x_3) = p(x_1) \cdot \prod_{i=2}^{3} p(x_i|x_{i-1})$

chain



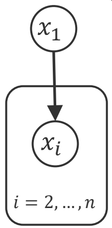
$p(x_1, x_2, x_3) = p(x_3|x_1, x_2) \cdot \prod_{i=1}^{2} p(x_i)$

sink

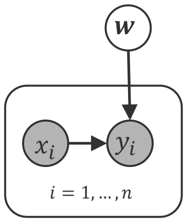
Bayesian Network Models

- **Plates**. If a subset of variables has the same relation only differing in their index, we use a "plate" to collapse them into a single graphical element.
  - Increase readability of models for large amounts of parameters and data
- A Bayesian network must always be a **directed acyclic graph** because only those have a topological order corresponding to a variable order.
- **Observed Variables**. If a subset of variables has been observed ("data"), the variable nodes are usually shaded ("clamped")
  - **Examples**:

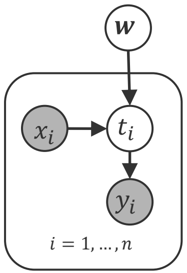
$$p(x_1, x_2, x_3) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i|x_1)$$



Linear Regression



Linear Classification



Conditional Independence

- In modelling specific data, domain experts often know whether or not two (latent) measurements can affect each other or not (i.e., are independent)
  - **Examples**:
    - Skills of two players in a video game are not dependent if they never met before
    - Skills of two players in a video game *are* dependent if they have played many times!

- Bayesian networks are useful to determine conditional independence.
- **Conditional Independence**. *A random variable x_i is conditionally independent of a random variable x_j given the variable x_k if for a values a of x_k*

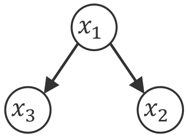
$$p\left(x_i | x_j, x_k = a\right) = p(x_i | x_k = a)$$

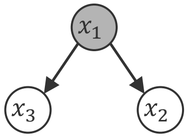
  - Equivalent definition: $p\left(x_i, x_j | x_k = a\right) = p(x_i | x_k = a) \cdot p\left(x_j | x_k = a\right)$
  - Shorthand notation (Dawid, 1979): $x_i \perp x_j | x_k$

Philip Dawid
(1946–)

Conditional Independence: Warm-Up I

- **Tail-to-Tail Node (x_1)**: $p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1)$





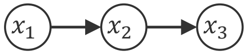
$$p(x_2, x_3) = \sum_{x_1} p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1) \neq p(x_2) \cdot p(x_3)$$

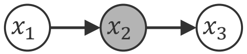
not (always) conditionally independent

$$p(x_2, x_3|x_1) = \frac{p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1)}{p(x_1)} = p(x_2|x_1) \cdot p(x_3|x_1)$$

conditionally independent

- **Head-to-Tail Node (x_2)**: $p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_2)$





$$p(x_1, x_3) = p(x_1) \cdot \sum_{x_2} p(x_2|x_1) \cdot p(x_3|x_2) = p(x_1) \cdot p(x_3|x_1) \neq p(x_1) \cdot p(x_3)$$

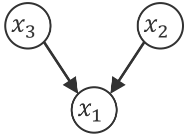
not (always) conditionally independent

$$p(x_1, x_3|x_2) = \frac{p(x_2|x_1) \cdot p(x_1)}{p(x_2)} \cdot p(x_3|x_2) = p(x_1|x_2) \cdot p(x_3|x_2)$$

conditionally independent

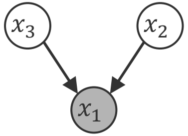
Conditional Independence: Warm-Up II

- **Head-to-Head Node (x_1)**: $p(x_1, x_2, x_3) = p(x_2) \cdot p(x_3) \cdot p(x_1|x_2, x_3)$



$$p(x_2, x_3) = \sum_{x_1} p(x_1|x_2, x_3) \cdot p(x_2) \cdot p(x_3) = p(x_2) \cdot p(x_3)$$

(conditionally) independent

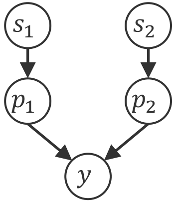


$$p(x_2, x_3|x_1) = \frac{p(x_1|x_2, x_3) \cdot p(x_2) \cdot p(x_3)}{p(x_1)} \neq p(x_2|x_1) \cdot p(x_3|x_1)$$

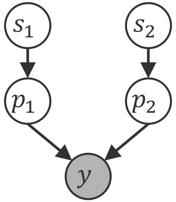
not (always) conditionally independent

  - It can be shown that the path between x_2 and x_3 become are only independent if *none* of the *descendant* node from x_1 (that can be reached in the directed graph) is observed!

- **Skill Example (ctd)**: Consider the skills of two players



Before match: s_1 and s_2 are independent

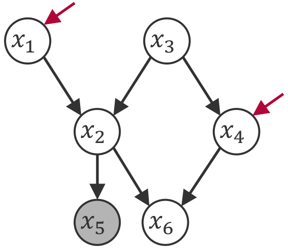


After macht: s_1 and s_2 are **not** independent

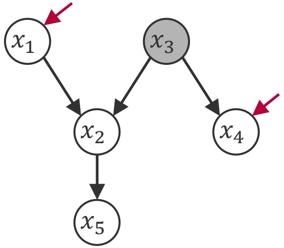
Conditional Independence: d-separation

Judea Pearl
(1936–)

- **Blocked Node**. *A node in a Bayesian network is said to be blocked if*
  1. *It's a head-to-tail or tail-to-tail node and the node is observed.*
  2. *It's a head-to-head node and neither the node or any of its descendants are observed.*
- **d-separation**. *Given a Bayesian network and a subset of observed variables, two non-observed variables x_i and x_j are* conditionally independent *(that is, d-separated) if every undirected path between x_i and x_j contains at least one blocked node.*
- **Examples**.



x_1 and x_4 are not independent



x_1 and x_4 are independent

Thank You!

Probabilistic Machine Learning

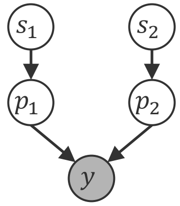
Graphical Models: Factor Graphs

Ralf Herbrich

Inference in Probabilistic Models

- **Learning**: In order to learn from data for most data models, we need to marginalize ("sum-out") all non-observed variables given the observed variables (i.e., data).
  - **Example**: Two player game with one winner

$$p(\boldsymbol{s}|y) \propto p(\boldsymbol{s}) \cdot \int \mathcal{N}(p_1; s_1, \beta^2) \cdot \mathcal{N}(p_2; s_2, \beta^2) \cdot \mathbb{I}(y(p_1 - p_2) > 0)\ dp_1\, dp_2$$



- **Problem**: Naïve summation scales exponentially because we have a sum of products (i.e., product of conditional disitrubtions of all latent variables)!
  - **Example**: Consider an example of n Bernoulli variables $x_1, \ldots, x_n$

$$p(x_1) = \underbrace{\boxed{\sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots \sum_{x_n=0}^{1}}}_{2^{n-1}\ \text{summations}} p(x_1, x_2, \ldots, x_n)$$

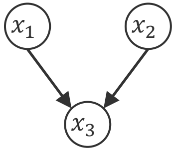
- **Idea**: We exploit the product structure of the probabilisitic model of our data because not every variable depends on all variables before them
  - **Example (ctd)**. Consider $p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i)$: then there are only $O(n)$ sums!

Factor Graphs

- **Factor Graph (Frey, 1998)**. *Given a product of m functions $f_1, f_2, \dots, f_m$, each over a subset of n variables $x_1, x_2, \dots, x_n$, a factor graph if a bipartite graphical model with m factor nodes and n variable nodes where an undirected edge connects f_i and x_j if and only if the function f_i depends on x_j.*
- Factor graphs are more expressive than a Bayesian network!

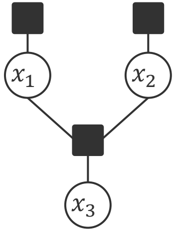
Brendan Frey
(1968 –)

Bayesian network



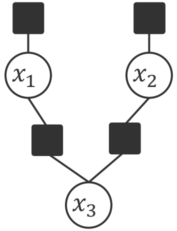
$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3|x_1, x_2)$

Corresponding factor graph



$p(x_1, x_2, x_3) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3, x_1, x_2)$

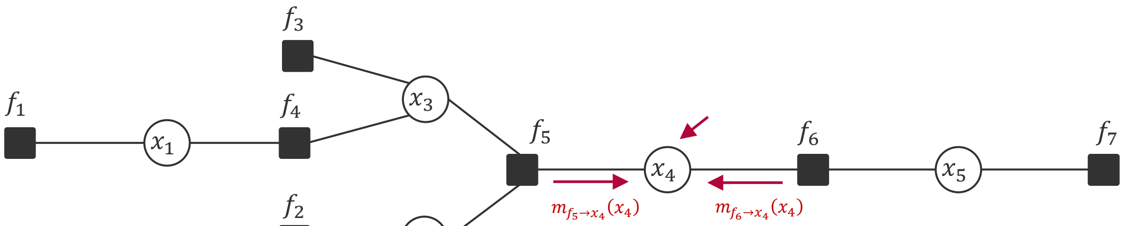
Factor graph with more structure



$p(x_1, x_2, x_3) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_1, x_3) \cdot f_4(x_2, x_3)$

Structure in $p(x_3|x_1, x_2)$

Sum-Product Algorithm: Marginals



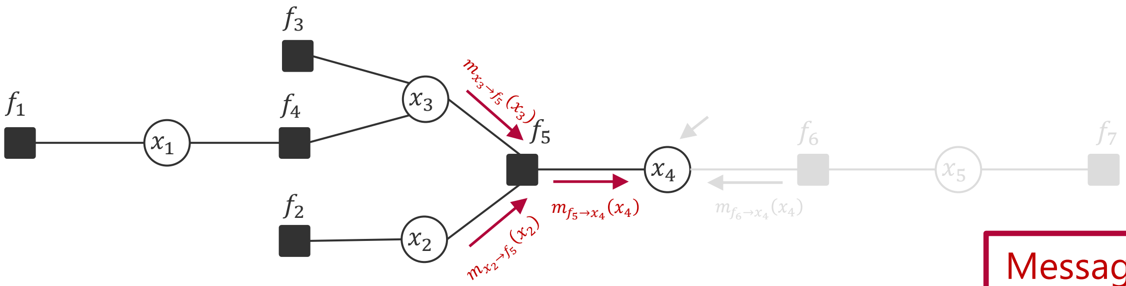
Message $m_{f_j \to x_i}(x_i)$ is the sum over all variables in the subtree rooted at f_j

$$p(x_4) = \sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} \sum_{\{x_5\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_2) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4) \cdot f_6(x_4, x_5) \cdot f_7(x_5)$$

$$= \underbrace{\left[\sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4) \right]}_{m_{f_5 \to x_4}(x_4)} \cdot \underbrace{\left[\sum_{\{x_5\}} f_6(x_4, x_5) \cdot f_7(x_5) \right]}_{m_{f_6 \to x_4}(x_4)}$$

Marginals are the product of all incoming messages from neighbouring factors!

Sum-Product Algorithm: Message from Factor to Variable



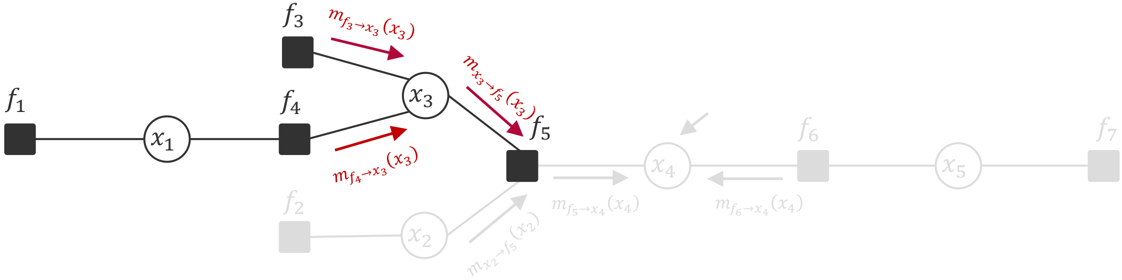
Message $m_{x_i \to f_j}(x_i)$ is the sum over all variables in the subtree rooted at x_i

$$m_{f_5 \to x_4}(x_4) = \sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4)$$

$$= \sum_{\{x_2\}} \sum_{\{x_3\}} f_5(x_2, x_3, x_4) \cdot \underbrace{[f_2(x_2)]}_{m_{x_2 \to f_5}(x_2)} \cdot \underbrace{\left[\sum_{\{x_1\}} f_1(x_1) \cdot f_3(x_3) \cdot f_4(x_1, x_3)\right]}_{m_{x_3 \to f_5}(x_3)}$$

Messages from a factor to a variable sum out all neighboring variables weighted by their incoming message

Sum-Product Algorithm: Message from Variable to Factor

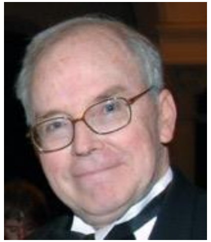


$$m_{x_3 \to f_5}(x_3) = \sum_{\{x_1\}} f_1(x_1) \cdot f_3(x_3) \cdot f_4(x_1, x_3)$$

$$= \underbrace{[f_3(x_3)]}_{m_{f_3 \to x_3}(x_3)} \cdot \underbrace{\left[\sum_{\{x_1\}} f_1(x_1) \cdot f_4(x_1, x_3)\right]}_{m_{f_4 \to x_3}(x_3)}$$

Messages from a variable to a factor multiply incoming message from neighboring factors

Sum-Product Algorithm



Robert McEliece
(1942 – 2019)

- **Sum-Product Algorithm (Aji-McEliece, 1997)**. Putting it all together, we have

$$p(x) = \prod_{f \in \mathrm{ne}(x)} m_{f \to x}(x)$$

$$m_{f \to x}(x) = \sum_{\{x' \in \mathrm{ne}(f) \setminus \{x\}\}} \cdots \sum_{\{x'' \in \mathrm{ne}(f) \setminus \{x\}\}} f(x, x', \dots, x'') \prod_{x' \in \mathrm{ne}(f) \setminus \{x\}} m_{x' \to f}(x')$$

$$m_{x \to f}(x) = \prod_{f' \in \mathrm{ne}(x) \setminus \{f\}} m_{f' \to x}(x)$$

- **Basis**: Generalized distributive law (which also holds for max-product)
- **Efficiency**: By storing messages, we
  - Only have to compute local summations in $O(2^T)$ where degree $T = \max_f |\mathrm{ne}(f)|$!
  - All marginals can be computed recursively in $O(E \cdot 2^T)$ vs $O(2^n)$!

Even more efficiency

- **Redundancies**. By the very definition of messages and marginals

$$p(x) = \prod_{f \in \mathrm{ne}(x)} m_{f\to x}(x) = m_{f'\to x}(x) \cdot \underbrace{\prod_{f \in \mathrm{ne}(x) \setminus \{f'\}} m_{f\to x}(x)}_{m_{x\to f'}(x)}$$

  - **Interpretation**. Application of Bayes' rule at a variable x at factor f

$$\underbrace{p(x)}_{\text{posterior}} = \underbrace{m_{f\to x}(x)}_{\text{Likelihood} \times \text{normalization}} \cdot \underbrace{m_{x\to f}(x)}_{\text{prior}}$$

- **Storage Efficiency**. We only store the marginals $p(x)$ and $m_{f\to x}(x)$ because

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

- **Exponential Family**. If all the messages from factors to variables are in the exponential family, then the marginals and messages from the variable to factors are simply additions and subtraction of natural parameters!
  - If $p(x) = \mathcal{G}(x; \tau_1, \rho_1)$ and $m_{f\to x}(x) = \mathcal{G}(x; \tau_2, \rho_2)$ then $m_{x\to f}(x) \propto \mathcal{G}(x; \tau_1 - \tau_2, \rho_1 - \rho_2)$

Approximate Message Passing

- **Message update from factors to variables**. For general factors f, the sum-product algorithm is not closed under the application of

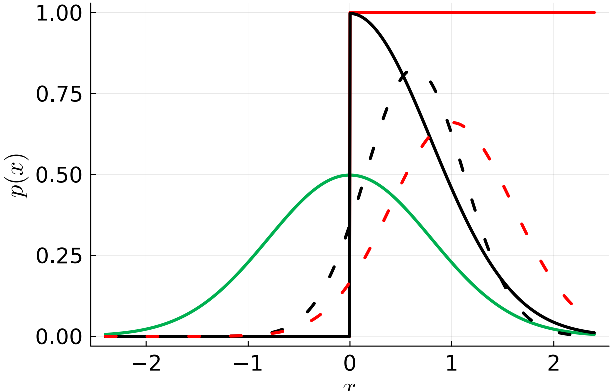
$$m_{f\to x}(x) = \sum_{\{x'\in \mathrm{ne}(f)\setminus\{x\}\}} \cdots \sum_{\{x''\in \mathrm{ne}(f)\setminus\{x\}\}} f(x, x', \dots, x'') \prod_{x'\in \mathrm{ne}(f)\setminus\{x\}} m_{x'\to f}(x')$$

  - **Example**: Truncating a 1D-Gaussian distribution

- **Idea**: Find the “best” approximation $\hat{p}(x)$ for the marginal $p(x)$ and approximate $m_{f\to x}(x)$ by

$$\widehat{m}_{f\to x}(x) = \frac{\hat{p}(x)}{m_{x\to f}(x)}$$

  - **Example**: Truncating a 1D-Gaussian distribution



Information Theoretic Approximation: KL Divergence

- **Problem**. We have a non-Gaussian posterior distribution $p(x)$ and would like to approximate it by a Gaussian $q(x) = \mathcal{N}(x; \mu, \sigma^2)$.
- **Idea**. The best approximation $\mu^*, {\sigma^2}^*$ minimizes the Kullback-Leibler divergence

$$\mathrm{KL}\big(p(\cdot)|\mathcal{N}(\cdot; \mu, \sigma^2)\big) = \int p(x) \cdot \log_2\left(\frac{p(x)}{\mathcal{N}(x; \mu, \sigma^2)}\right) dx$$

- **Theorem (Moment Matching)**. *Given any distribution* $p(x)$ *the minimizer* $\mu^*, {\sigma^2}^*$ *of the KL divergence* $\mathrm{KL}\big(p(\cdot)|\mathcal{N}(\cdot; \mu, \sigma^2)\big)$ *to a Gaussian distribution is*

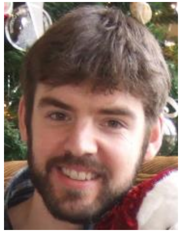
$$\mu^* = E_{x\sim p(x)}[x] \quad \text{and} \quad {\sigma^2}^* = E_{x\sim p(x)}[x^2] - (\mu^*)^2$$

Solomon Kullback
(1909 – 1994)

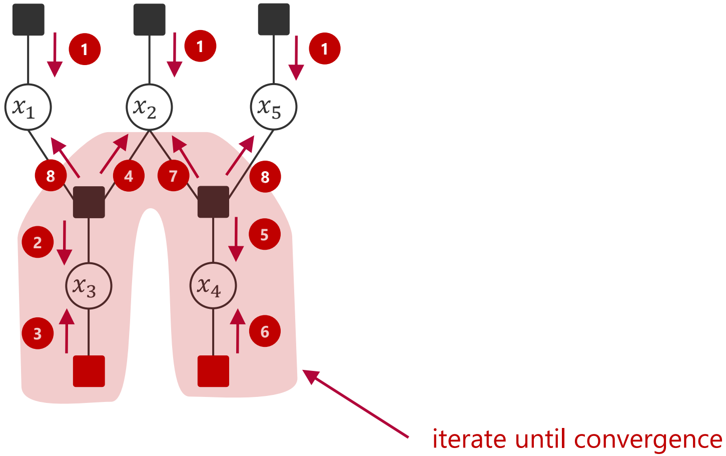
Richard Leibler
(1914 – 2003)

Expectation Propagation

- **Idea**: If we have factors in the factor graph that require approximate messages, we keep iterating on the whole path between them until convergence minimizing $\mathrm{KL}\big(p(\cdot)|\mathcal{N}(\cdot;\mu,\sigma^2)\big)$ locally for the affected marginals of the approximate factor.
- **Theorem (Minka, 2003)**: Approximate message passing will converge if the approximating distribution is in the exponential family!



Tom Minka



Thank You!

Probabilistic Machine Learning

Bayesian Ranking

Ralf Herbrich

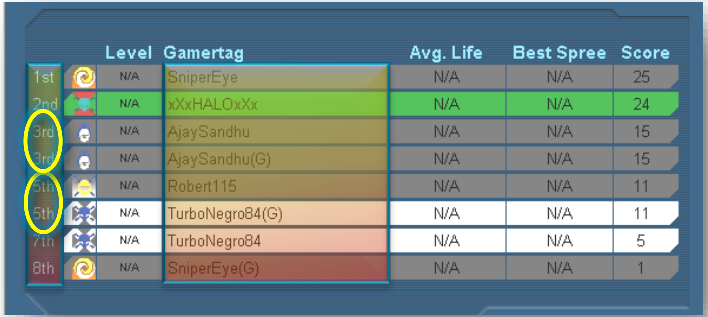
Hasso Plattner Institut
Digital Engineering • Universität Potsdam

The Skill Rating Problem

- **Given**:
  - **Match outcomes**: Orderings among k teams consisting of $n_1, n_2, \dots, n_k$ players.

	Team	Score
1st	Red Team	50
2nd	Blue Team	40

	Level	Gamertag	Avg. Life	Best Spree	Score
1st	10	BlueBot	00:00:49	6	15
1st	7	SniperEye	00:00:41	4	14
1st	9	ProThepirate	00:01:07	3	13
1st	10	dazdemon	00:00:59	3	8
2nd	10	WastedHarry	00:00:41	4	17
2nd	3	Ascla	00:00:37	2	10
2nd	9	Antidote4Losing	00:00:41	2	9
2nd	12	Blackknight9	00:00:48	3	4



	Level	Gamertag	Avg. Life	Best Spree	Score
1st	N/A	SniperEye	N/A	N/A	25
2nd	N/A	xXxHALOxXx	N/A	N/A	24
3rd	N/A	AjaySandhu	N/A	N/A	15
3rd	N/A	AjaySandhu(G)	N/A	N/A	15
5th	N/A	Robert115	N/A	N/A	11
5th	N/A	TurboNegro84(G)	N/A	N/A	11
7th	N/A	TurboNegro84	N/A	N/A	5
8th	N/A	SniperEye(G)	N/A	N/A	1

- **Questions**:
  1. Skill s_i for each player such that $s_i > s_j \Leftrightarrow P(\text{Player } i \text{ wins}) > P(\text{Player } j \text{ wins})$
  2. Global ranking among all players
  3. Fair matches between teams of players

Two-Player Match Outcome Model

- **Simple Two-Player Games**: Our data is the identity i and j of the two players and the outcome $y \in \{-1, +1\}$ of a match between them
  - **Bradley-Terry Model (1952)**: Model of a win given skills s_i and s_j is

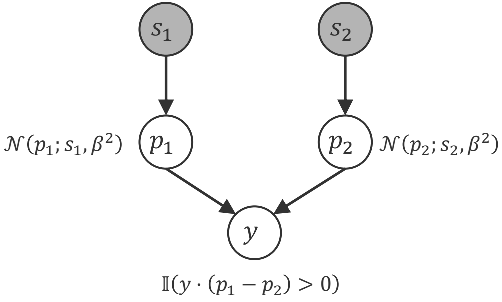
$$P(y = 1 | s_i, s_j) = \frac{\exp(s_i)}{\exp(s_i) + \exp(s_j)} = \boxed{\frac{\exp(s_i - s_j)}{1 + \exp(s_i - s_j)}}$$

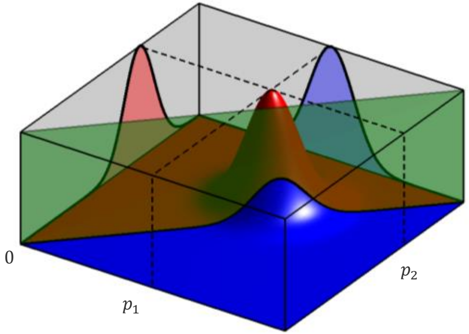
Logistic sigmoid in skill difference

  - **Thurstone Case V Model (1927)**: Model of a win given skills s_i and s_j is

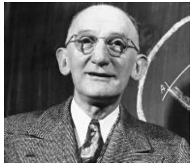
$$P(y = 1 | s_i, s_j) = \boxed{\int_0^\infty \mathcal{N}(t; s_i - s_j, 2\beta^2)\, dt}$$

Probit sigmoid in skill difference



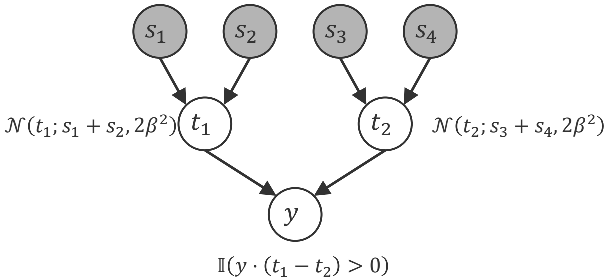



Ralph A. Bradley (1923 – 2001)


Louis Leon Thurstone (1887 – 1955)

Two-Team Match Outcome Model

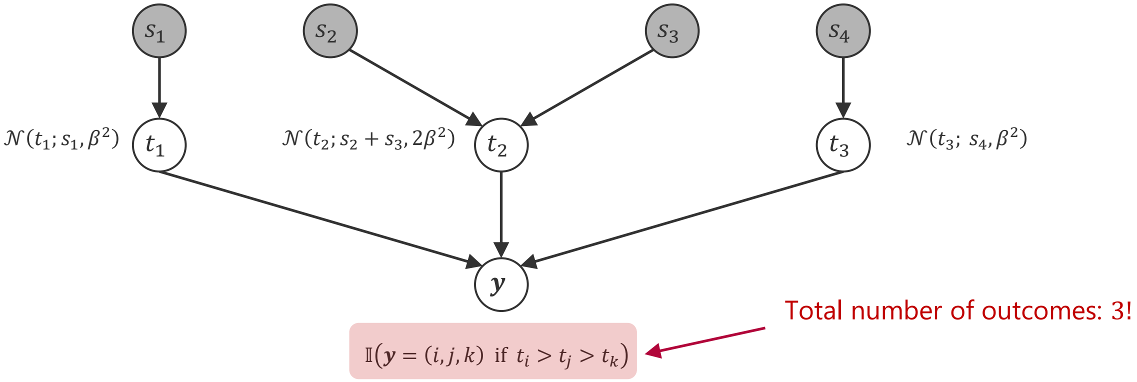
- **Team Assumption**: Skill of a team is the sum of the skill of its players



- **Pro**: Games where the team scores are additive (e.g., kill count in first-person shooter)
- **Con**: Games where the outcome is determined by a single player (e.g., fastest car in a race)
- **Observation**: Match outcomes correlate the skills of players
  - **Same Team**: Anti-correlated
  - **Opposite Teams**: Correlated

Multi-Team Match Outcome Model

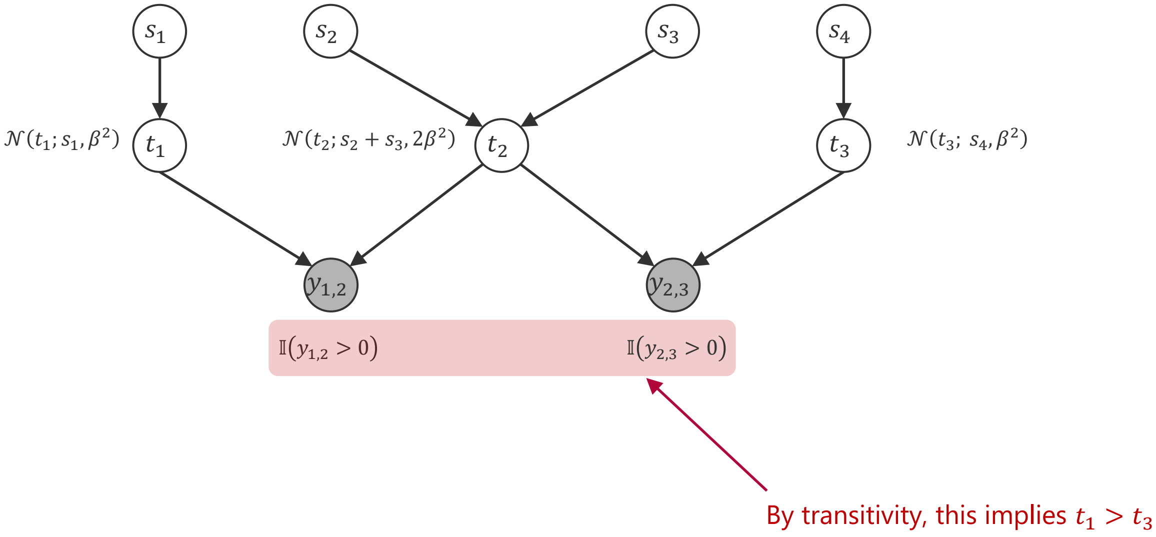
- **Possible Outcomes**: Permutations $\boldsymbol{y} \in \{1,2,3\}^3$ of players



- **Easy** to **sample** for given skills but computationally difficult to "invert"!

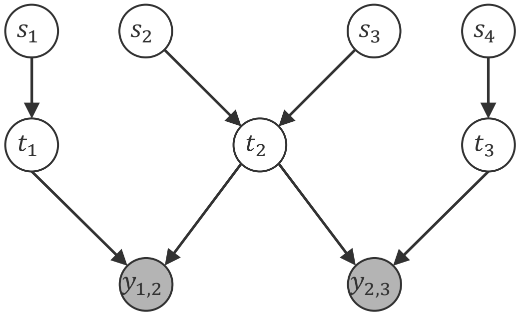
From Match Outcomes to Pairwise Rankings

- **Learning**: In the ranking setting, we observe multi-team match outcomes and want to infer the skills!
- **Idea**: Leverage the transitivity of the real line of latent scores!

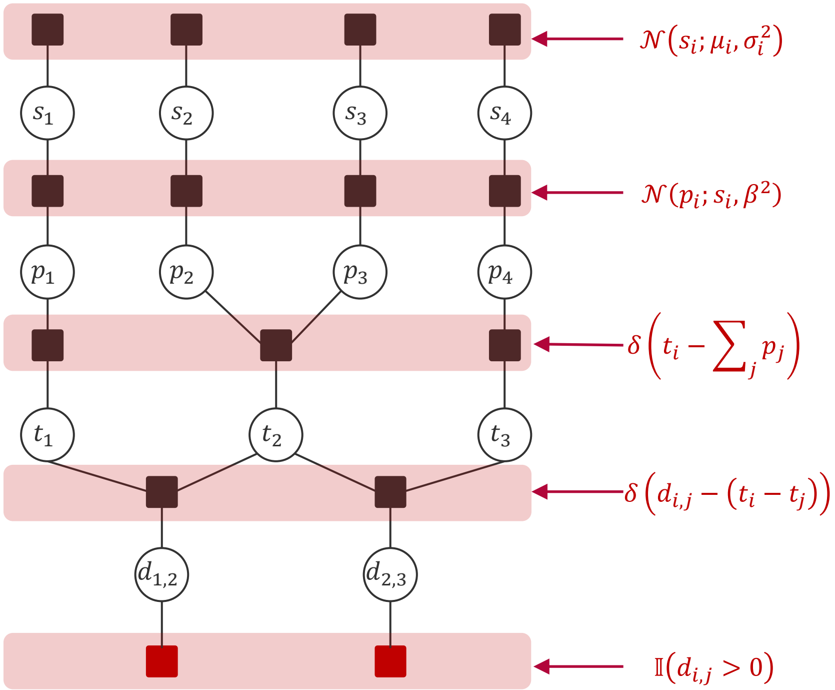


TrueSkill Factor Graphs

Bayesian Network

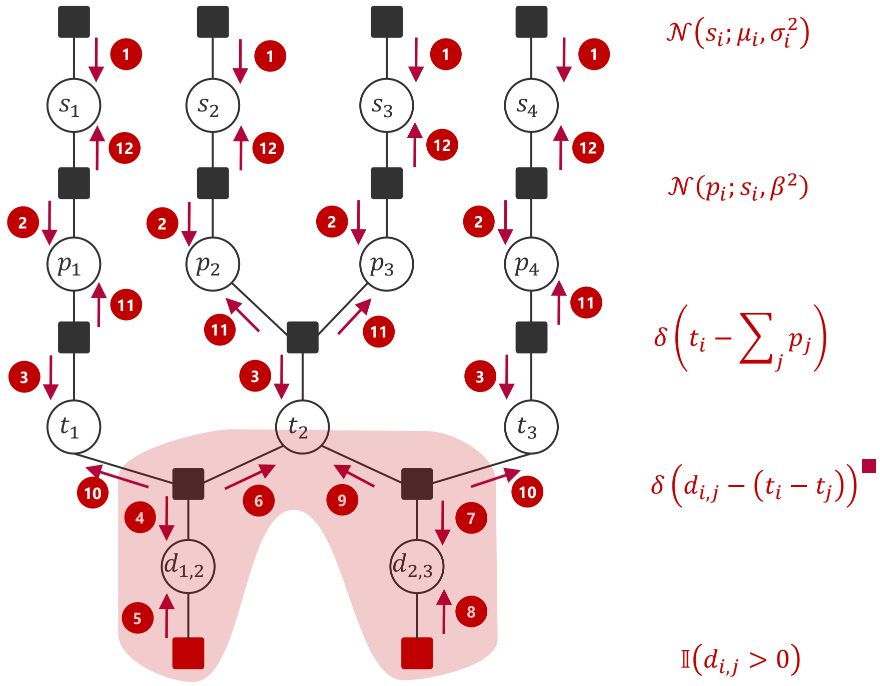


Factor Graph



(Approximate) Message Passing in TrueSkill Factor Graphs

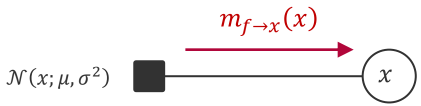
TrueSkill Factor Graph



- Four Phases
  1. Pass prior messages (1)
  2. Pass messages *down* to the team performances (2 to 3)
  3. Iterate the approximate messages on the pairwise team differences (4 to 9)
  4. Pass messages back from *up* from team performances to player skill (10 – 12)
- Since this is a *tree*, the algorithm is guaranteed to converge!

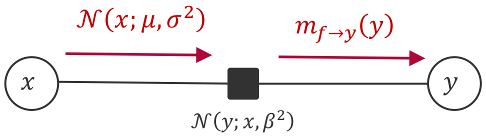
Message Update Equations

Gaussian Factor



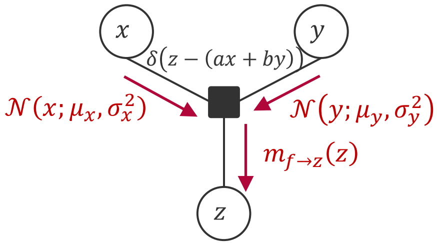
$$m_{f\to x}(x) = \mathcal{N}(x;\mu,\sigma^2)$$

Gaussian Mean Factor



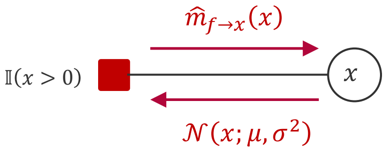
$$m_{f\to y}(y) = \int \mathcal{N}(y;x,\beta^2)\cdot\mathcal{N}(x;\mu,\sigma^2)\,dx = \mathcal{N}(y;\mu,\sigma^2+\beta^2)$$

Weighted Sum Factor



$$m_{f\to z}(z) = \mathcal{N}\left(z;a\mu_x+b\mu_y,a^2\sigma_x^2+b^2\sigma_y^2\right)$$

Greater-Than Factor



$$\widehat{m}_{f\to x}(x) = \frac{\hat{p}(x)}{m_{x\to f}(x)} = \frac{\mathcal{N}(x;\hat{\mu},\hat{\sigma}^2)}{\mathcal{N}(x;\mu,\sigma^2)}$$

Mean and variance of
a truncated Gaussian $\mathcal{N}(x;\mu,\sigma^2)$

Truncated Gaussians

- **Truncated Gaussians**. *A truncated Gaussian given by* $p(x) \propto \mathbb{I}(x > 0) \cdot \mathcal{N}(x; \mu, \sigma^2)$ *has the following three moments*

$$Z(\mu, \sigma) = \int_{-\infty}^{+\infty} p(x)\, dx = 1 - F(0; \mu, \sigma^2)$$

Follows from definition of F

$$E[X] = \int_{-\infty}^{+\infty} x \cdot p(x)\, dx = \mu + \sigma \cdot v\left(\frac{\mu}{\sigma}\right)$$

Additive update that goes to zero as $\frac{\mu}{\sigma} \to \infty$

$$\mathrm{var}[X] = \int_{-\infty}^{+\infty} (x - E[X])^2 \cdot p(x)\, dx = \sigma^2 \cdot \left(1 - w\left(\frac{\mu}{\sigma}\right)\right)$$

Multiplicative update that goes to 1 as $\frac{\mu}{\sigma} \to \infty$

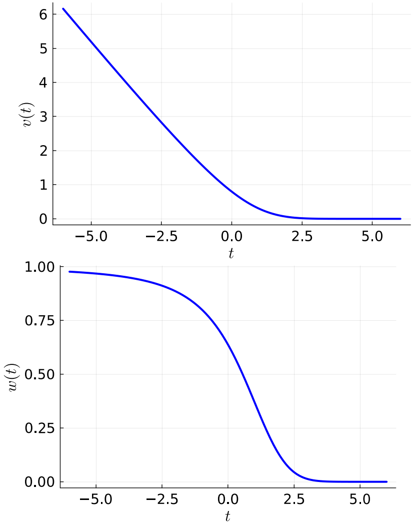
where the probit $F(t; \mu, \sigma^2) := \int_{-\infty}^{t} \mathcal{N}(x; \mu, \sigma^2)\, dx$ *and*

$$v(t) := \frac{\mathcal{N}(t; 0,1)}{F(t; 0,1)}$$

Converges to $-t$ as $t \to -\infty$

$$w(t) := v(t) \cdot [v(t) + t]$$

- This can be generalized to an arbitrary interval $[a, b]$ where the Gaussian is truncated!



Probabilistic Machine Learning

Bayesian Ranking

Decision Making: Match Quality and Leaderboards

- **Match Quality**: Decide if two players i and j should be matched
  - **Idea**: Pick the pair (i, j) where the two players have equal skills

$$\text{Quality}(i,j) = \frac{P\left(p_i \approx p_j \middle| \mu_i - \mu_j, \sigma_i^2 + \sigma_j^2\right)}{P\left(p_i \approx p_j \middle| \mu_i - \mu_j = 0, \sigma_i^2 + \sigma_j^2 = 0\right)}$$

  - **Observation**: This pair (i, j) approximately maximizes the information (entropy!) of the predicted match outcome because it gets closest to 50% winning probability

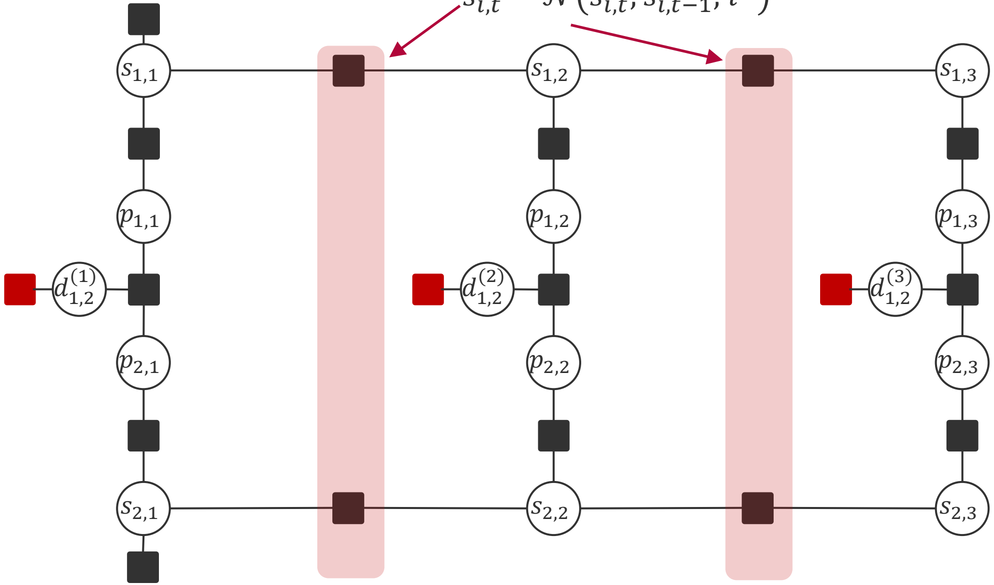
- **Leaderboard**: Decide how to display the best to worst player
  - **Observation**: There is an asymmetry in making a ranking mistake
    - **Cheap**: Ranking a truly good player lower than they should be (why?)
    - **Expensive**: Ranking a truly bad player higher than they should be (why?)
    - The loss minimizer of this decision process is a **quantile** $\mu - k \cdot \sigma$

1	27	SEWiCSYDE OWNS
2	26	FATAL REVENGE
3	25	Paranoia 1
4	25	Paulk
5	25	IxX OMG XxI
6	25	BittyTom
7	24	brian 2OO7
8	24	SEXY MOZES
9	24	droplates
10	24	jaCKdaSaMuRai
11	24	ll Me ll
12	24	iamNightMare
13	24	a retarded007
14	24	Perfected Brit
15	24	THE MUFFIN MANx
16	23	TheVunit
17	23	Mr Sushi87

Skill Dynamics

- **Dynamics**: In reality, skills of players evolve over time and are not stationary
  - **Idea**: Since we do not know which direction, assume that the skill of player i at time t depends on the skill of the same player at time $t-1$ via

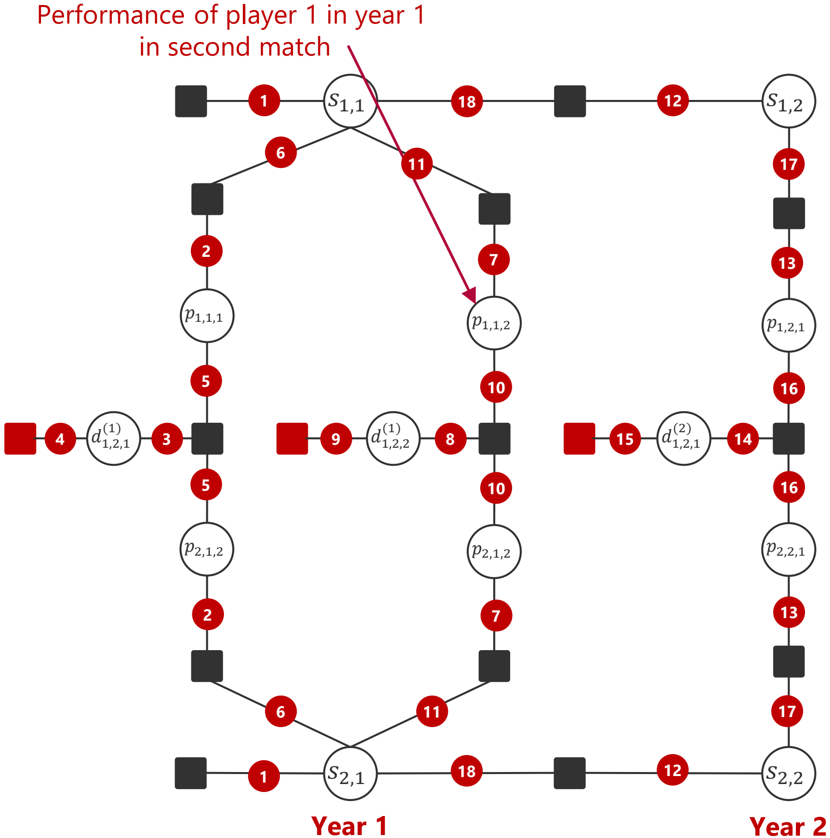
$$s_{i,t} \sim \mathcal{N}\left(s_{i,t}; s_{i,t-1}, \tau^2\right)$$



Probabilistic Machine Learning

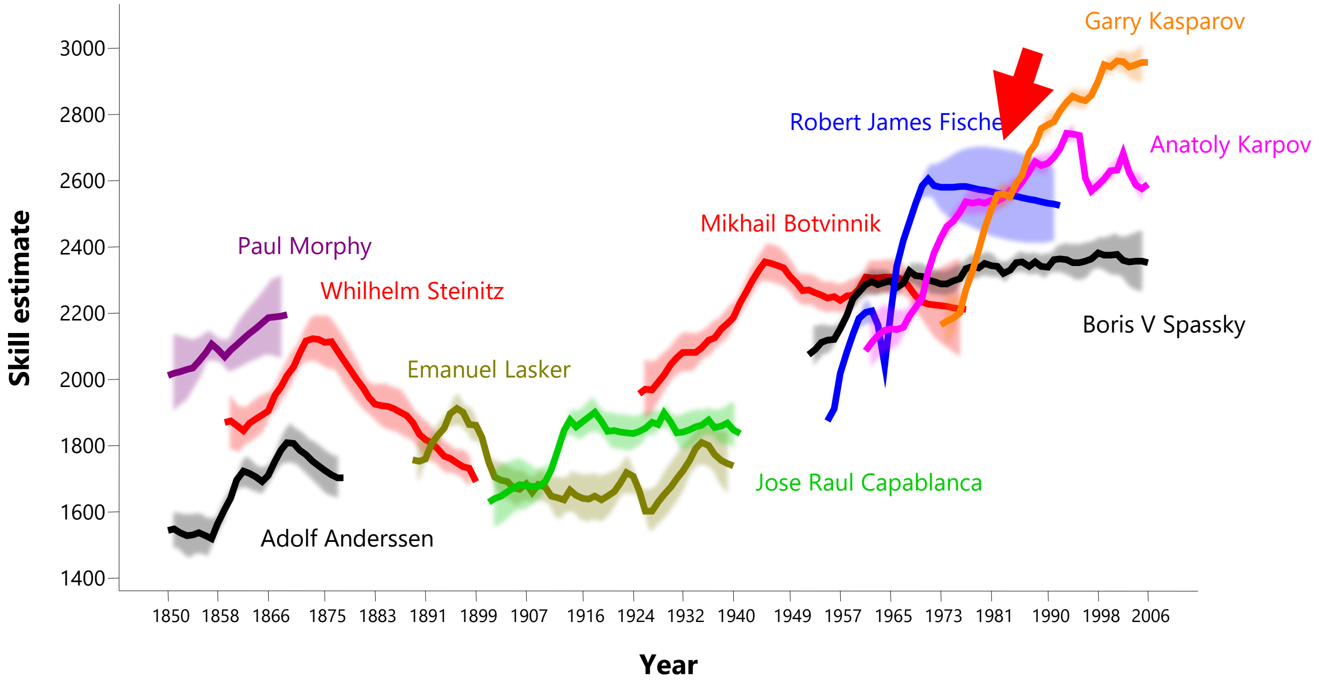
Bayesian Ranking

TrueSkill Through Time: Message Schedule



- Four Phases
  1. **Prior (1)**: Send prior messages to each skill variable for the first year of a player
  2. **Annual Matches (2-11)**: Loop over all (2-player) matches in a year until the skill marginals for all active player in that year does not change (much) anymore
  3. **Forward Dynamics (12)**: Send skill dynamics messages forward in time from t to $t+1$ and keep running step 2. (13 – 17).
  4. **Backward Dynamics (18)**: Send skill dynamics messages backward in time from year $t+1$ to t and keep running step 2. (2-11)
- Stop when no variable in the outer loop changes much anymore.

TrueSkill-Through-Time: Chess Players



History of Chess
3.5M match outcomes
20 million variables
40 million factors

Thank You!