EXPECTATION PROPAGATION

Сергей Николенко

СПбГУ — Санкт-Петербург

17 апреля 2024 г.

Random facts:

- 17 апреля 1492 г. был утверждён один из важнейших научных грантов: Христофор Колумб подписал контракт с Испанией, обязуясь открыть новый путь в Индию
- 17 апреля 1521 г. Мартин Лютер был отлучён от лона Римской Католической Церкви, а 17 апреля 1607 г. 21-летний Арман Жан Дю Плесси Де Ришельё стал епископом
- 17 апреля 1607 г. католическая церковь взяла реванш: 21-летний Арман Жан дю Плесси де Ришелье был посвящён в сан епископа
- 17 апреля 1722 г. Пётр I ввёл подать на ношение бороды: 50 рублей в год!
- 17 апреля 1875 г. полковник британских войск в Индии Невилл Чемберлен изобрёл снукер
- 17 апреля 1986 г. постановление ЦК КПСС «Об основных направлениях ускорения решения жилищной проблемы в стране» пообещало, что у каждой семьи к 2000 году будет отдельная квартира или дом
- 17 апреля 2011 г. на телеканале HBO вышла первая серия *Game of Thrones*

LDA

LDA

- Развитие идей pLSA – LDA (Latent Dirichlet Allocation).
- Это фактически байесовский вариант pLSA, сейчас нарисуем картинки, добавим априорные распределения и посмотрим, как сработают наши методы приближённого вывода.
- Задача та же: смоделировать большую коллекцию текстов (например, для information retrieval или классификации).

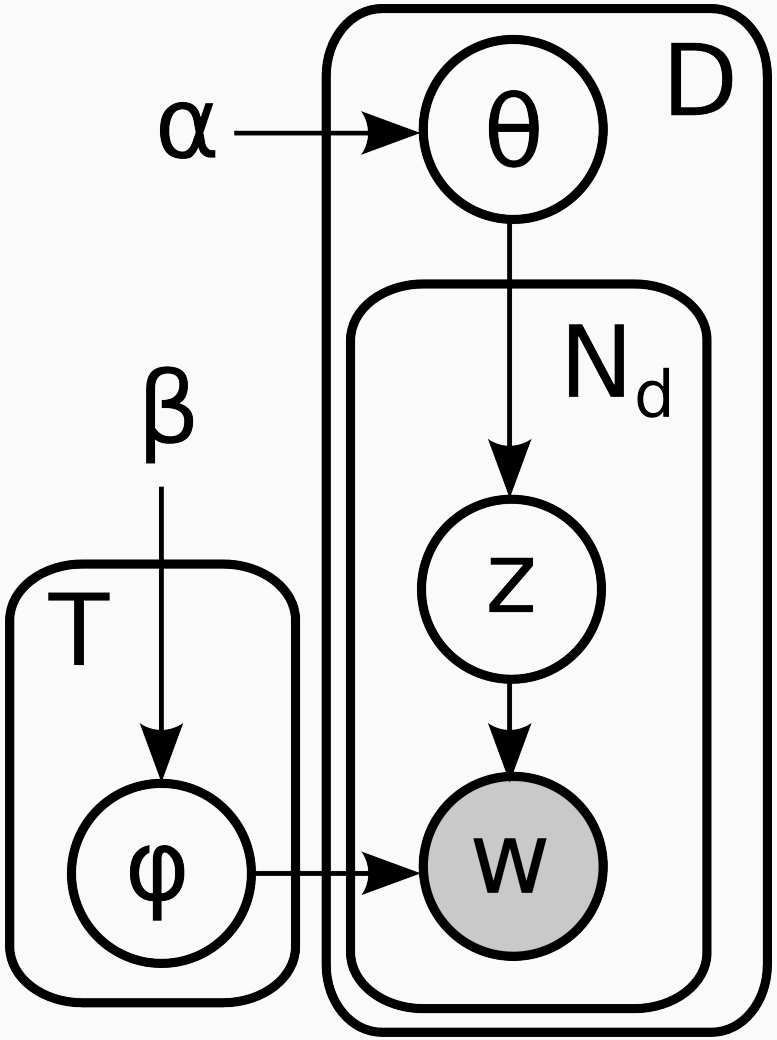
LDA

- У одного документа может быть несколько тем. Давайте построим иерархическую байесовскую модель:
  - на первом уровне – смесь, компоненты которой соответствуют «темам»;
  - на втором уровне – мультиномиальная переменная с априорным распределением Дирихле, которое задаёт «распределение тем» в документе.

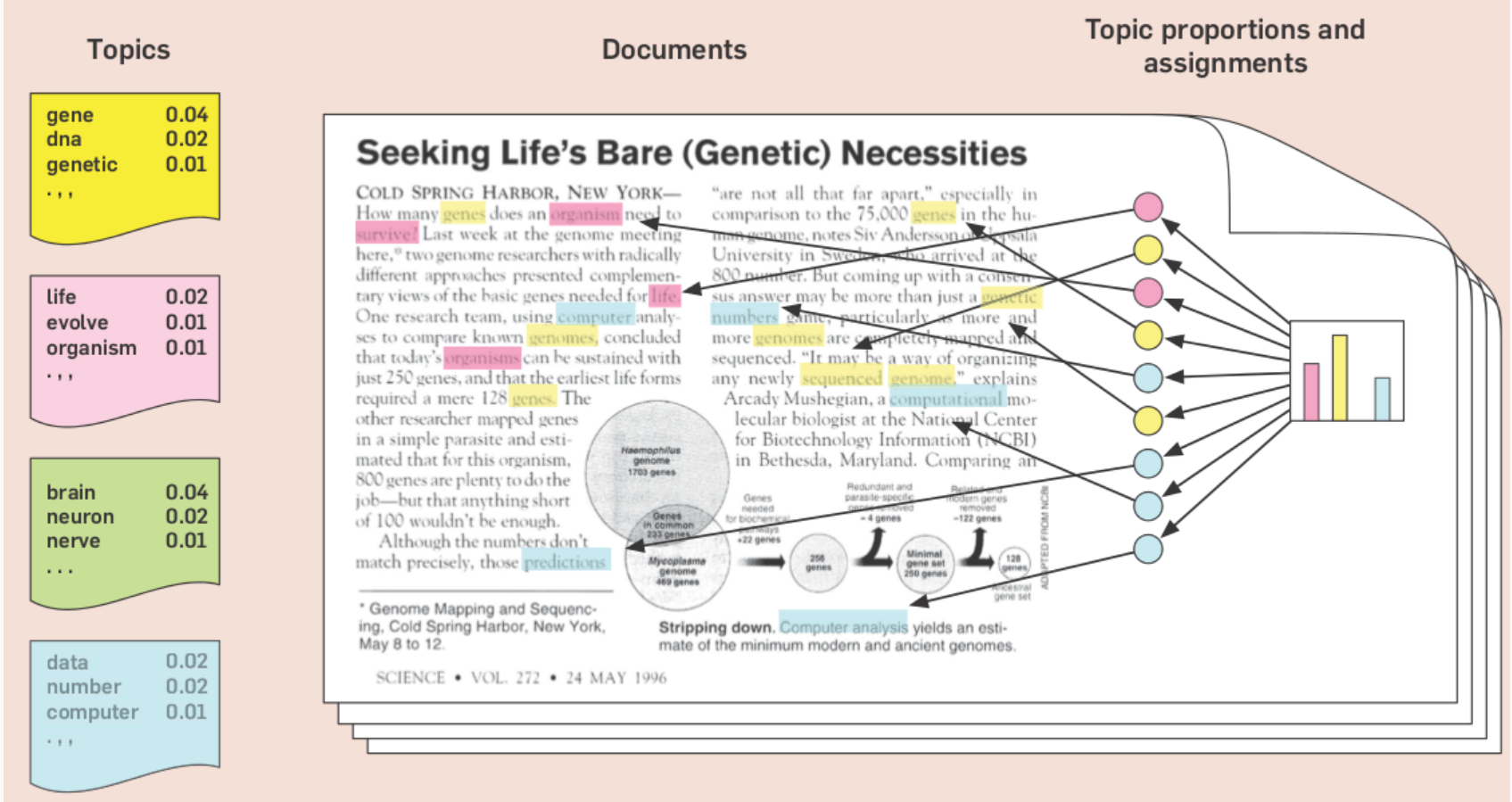
LDA

- Если формально: слова берутся из словаря $\{1, \dots, V\}$; слово – это вектор w, $w_i \in \{0, 1\}$, где ровно одна компонента равна 1.
- Документ – последовательность из N слов $\mathbf{w}$. Нам дан корпус из M документов $D = \{\mathbf{w}_d \mid d = 1..M\}$.
- Генеративная модель LDA выглядит так:
  - выбрать $\theta \sim \mathrm{Di}(\alpha)$;
  - для каждого из N слов w_n:
    - выбрать тему $z_n \sim \mathrm{Mult}(\theta)$;
    - выбрать слово $w_n \sim p(w_n \mid z_n, \beta)$ по мультиномиальному распределению.

LDA: графическая модель



LDA: что получается [Blei, 2012]



LDA: вывод

- Два основных подхода к выводу в сложных вероятностных моделях, в том числе LDA:
  - *вариационные приближения*: рассмотрим более простое семейство распределений с новыми параметрами и найдём в нём наилучшее приближение к неизвестному распределению;
  - *сэмплирование*: будем набрасывать точки из сложного распределения, не считая его явно, а запуская марковскую цепь под графиком распределения (частный случай – сэмплирование по Гиббсу).
- Сэмплирование по Гиббсу обычно проще расширить на новые модификации LDA, но вариационный подход быстрее и часто стабильнее.

Вывод в LDA

- Рассмотрим задачу байесовского вывода, т.е. оценки апостериорного распределения θ и z после нового документа:

$$p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{p(\mathbf{w} \mid \alpha, \beta)}.$$

- Правдоподобие набора слов $\mathbf{w}$ оценивается как

$$p(\mathbf{w} \mid \alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left[\prod_{i=1}^{k} \theta_i^{\alpha_i - 1}\right] \left[\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j}\right] d\theta,$$

и это трудно посчитать, потому что θ и β путаются друг с другом.

Вывод в LDA

- Вариационное приближение – рассмотрим семейство распределений

$$q(\theta, z \mid \mathbf{w}, \gamma, \phi) = p(\theta \mid \mathbf{w}, \gamma) \prod_{n=1}^{N} p(z_n \mid \mathbf{w}, \phi_n).$$

- Тут всё расщепляется, и мы добавили вариационные параметры γ (Дирихле) и ϕ (мультиномиальный).
- Заметим, что параметры для каждого документа могут быть свои – всё условно по $\mathbf{w}$.

LDA: сэмплирование по Гиббсу

- В базовой модели LDA сэмплирование по Гиббсу после несложных преобразований сводится к так называемому *сжатому сэмплированию по Гиббсу* (collapsed Gibbs sampling), где переменные z_w итеративно сэмплируются по следующему распределению:

$$p(z_w = t \mid \mathbf{z}_{-w}, \mathbf{w}, \alpha, \beta) \propto q(z_w, t, \mathbf{z}_{-w}, \mathbf{w}, \alpha, \beta) =$$

$$\frac{n^{(d)}_{-w,t} + \alpha}{\sum_{t' \in T} \left(n^{(d)}_{-w,t'} + \alpha\right)} \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta\right)},$$

где $n^{(d)}_{-w,t}$ – число слов в документе d, выбранных по теме t, а $n^{(w)}_{-w,t}$ – число раз, которое слово w было порождено из темы t, не считая текущего значения z_w; заметим, что оба этих счётчика зависят от других переменных $\mathbf{z}_{-w}$.

LDA: сэмплирование по Гиббсу

- Из сэмплов затем можно оценить переменные модели

$$\theta_{d,t} = \frac{n^{(d)}_{-w,t} + \alpha}{\sum_{t' \in T} \left(n^{(d)}_{-w,t'} + \alpha \right)},$$

$$\phi_{w,t} = \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta \right)},$$

где $\phi_{w,t}$ – вероятность получить слово w в теме t, а $\theta_{d,t}$ – вероятность получить тему t в документе d.

Варианты и расширения модели LDA

- В последние десять лет эта модель стала основой для множества различных расширений.
- Каждое из этих расширений содержит либо вариационный алгоритм вывода, либо алгоритм сэмплирования по Гиббсу для модели, которая, основываясь на LDA, включает в себя ещё и какую-либо дополнительную информацию или дополнительные предполагаемые зависимости.
- Обычно – или дополнительная структура на темах, или дополнительная информация.

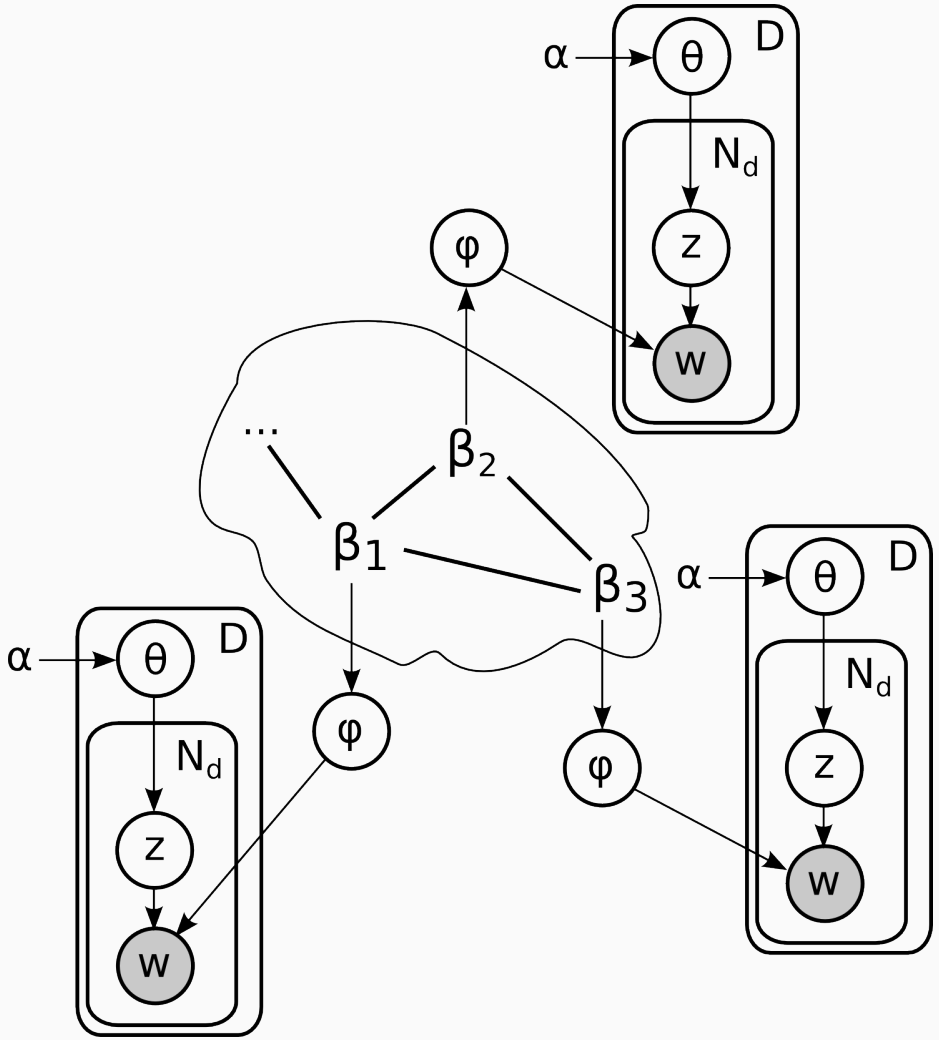
Коррелированные тематические модели

- В базовой модели LDA распределения слов по темам независимы и никак не скоррелированы; однако на самом деле, конечно, некоторые темы ближе друг к другу, многие темы делят между собой слова.
- Коррелированные тематические модели (correlated topic models, CTM); отличие от базового LDA здесь в том, что используется логистическое нормальное распределение вместо распределения Дирихле; логистическое нормальное распределение более выразительно, оно может моделировать корреляции между темами.
- Предлагается алгоритм вывода, основанный на вариационном приближении.

Марковские тематические модели

- Марковские тематические модели (Markov topic models, MTM): марковские случайные поля для моделирования взаимоотношений между темами в разных частях датасета (разных корпусах текстов).
- MTM состоит из нескольких копий гиперпараметров β_i в LDA, описывающих параметры разных корпусов с одними и теми же темами. Гиперпараметры β_i связаны между собой в марковском случайном поле (Markov random field, MRF).
- В результате тексты из i-го корпуса порождаются как в обычном LDA, используя соответствующее β_i.
- В свою очередь, β_i подчиняются априорным ограничениям, которые позволяют «делить» темы между корпусами, задавать «фоновые» темы, присутствующие во всех корпусах, накладывать ограничения на взаимоотношения между темами и т.д.

Марковские тематические модели



Реляционная тематическая модель

- Реляционная тематическая модель (relational topic model, RTM) – иерархическая модель, в которой отражён граф структуры сети документов.
- Генеративный процесс в RTM работает так:
  - сгенерировать документы из обычной модели LDA;
  - для каждой пары документов d_1, d_2 выбрать бинарную переменную y_{12}, отражающую наличие связи между d_1 и d_2:

$$y_{12} \mid \mathbf{z}_{d_1}, \mathbf{z}_{d_2} \sim \psi(\cdot \mid \mathbf{z}_{d_1}, \mathbf{z}_{d_2}, \eta).$$

- В качестве ψ берутся разные сигмоидальные функции; разработан алгоритм вывода, основанный на вариационном приближении.

Модели, учитывающие время

- Ряд важных расширений LDA касается учёта трендов, т.е. изменений в распределениях тем, происходящих со временем.
- Цель – учёт времени, анализ «горячих» тем, анализ того, какие темы быстро становятся «горячими» и столь же быстро затухают, а какие проходят «красной нитью» через весь исследуемый временной интервал.

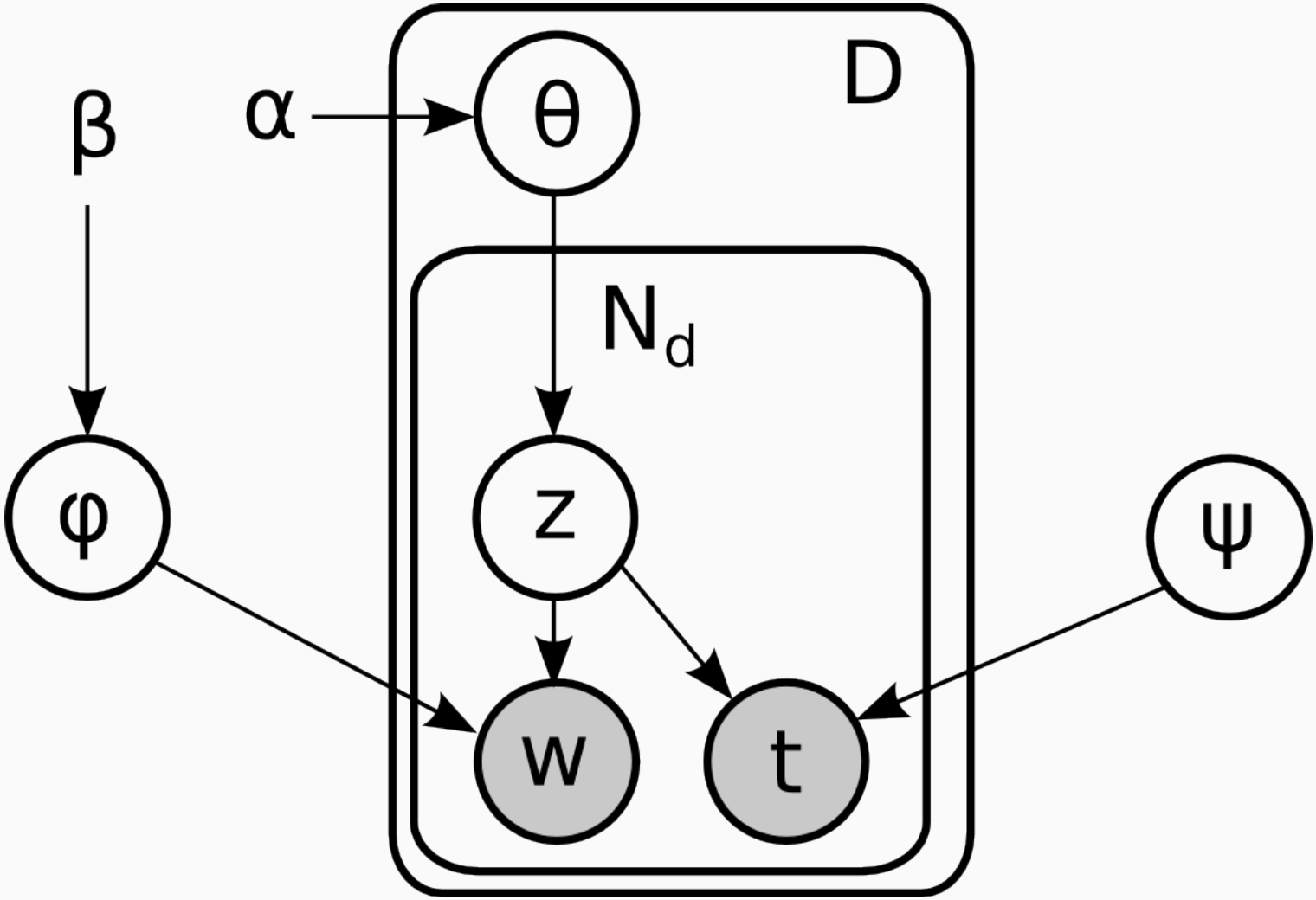
Topics over Time

- В модели TOT (Topics over Time) время предполагается непрерывным, и модель дополняется бета-распределениями, порождающими временные метки (timestamps) для каждого слова.
- Генеративная модель модели Topics over Time такова:
  - для каждой темы $z = 1..T$ выбрать мультиномиальное распределение ϕ_z из априорного распределения Дирихле β;
  - для каждого документа d выбрать мультиномиальное распределение θ_d из априорного распределения Дирихле α, затем для каждого слова $w_{di} \in d$:
    - выбрать тему z_{di} из θ_d;
    - выбрать слово w_{di} из распределения $\phi_{z_{di}}$;
    - выбрать время t_{di} из бета-распределения $\psi_{z_{di}}$.

Topics over Time

- Основная идея заключается в том, что каждой теме соответствует её бета-распределение ψ_z, т.е. каждая тема локализована во времени (сильнее или слабее, в зависимости от параметров ψ_z).
- Таким образом можно как обучить глобальные темы, которые всегда присутствуют, так и подхватить тему, которая вызвала сильный краткий всплеск, а затем пропала из виду; разница будет в том, что дисперсия ψ_z будет в первом случае меньше, чем во втором.

Topics over Time



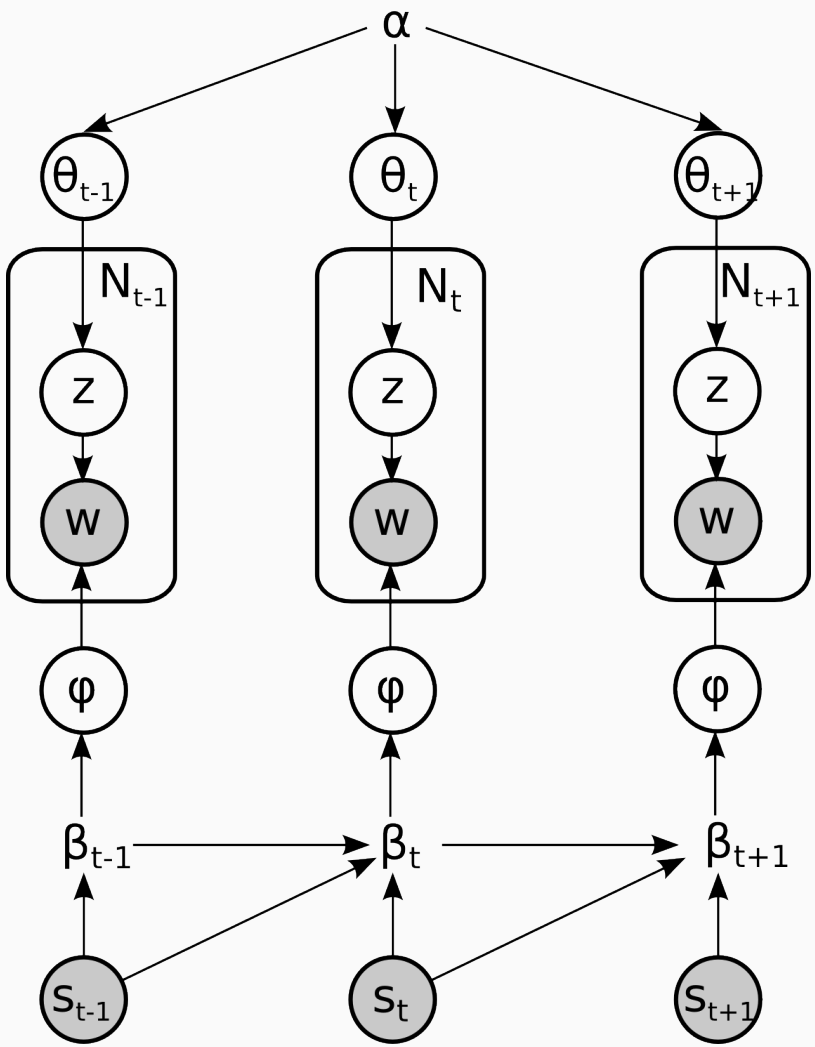
Динамические тематические модели

- *Динамические тематические модели* представляют временную эволюцию тем через эволюцию их гиперпараметров α и/или β.
- Бывают дискретные ([d]DTM), в которых время дискретно, и непрерывные, где эволюция гиперпараметра β (α здесь предполагается постоянным) моделируется посредством броуновского движения: для двух документов i и j (j позже i) верно, что

$$\beta_{j,k,w} \mid \beta_{i,k,w}, s_i, s_j \sim \mathcal{N}(\beta_{i,k,w}, v\Delta_{s_i,s_j}),$$

  где s_i и s_j – это отметки времени (timestamps) документов i и j, $\Delta(s_i, s_j)$ – интервал времени, прошедший между ними, v – параметр модели.
- В остальном генеративный процесс остаётся неизменным.

Непрерывная динамическая тематическая модель (cDTM)



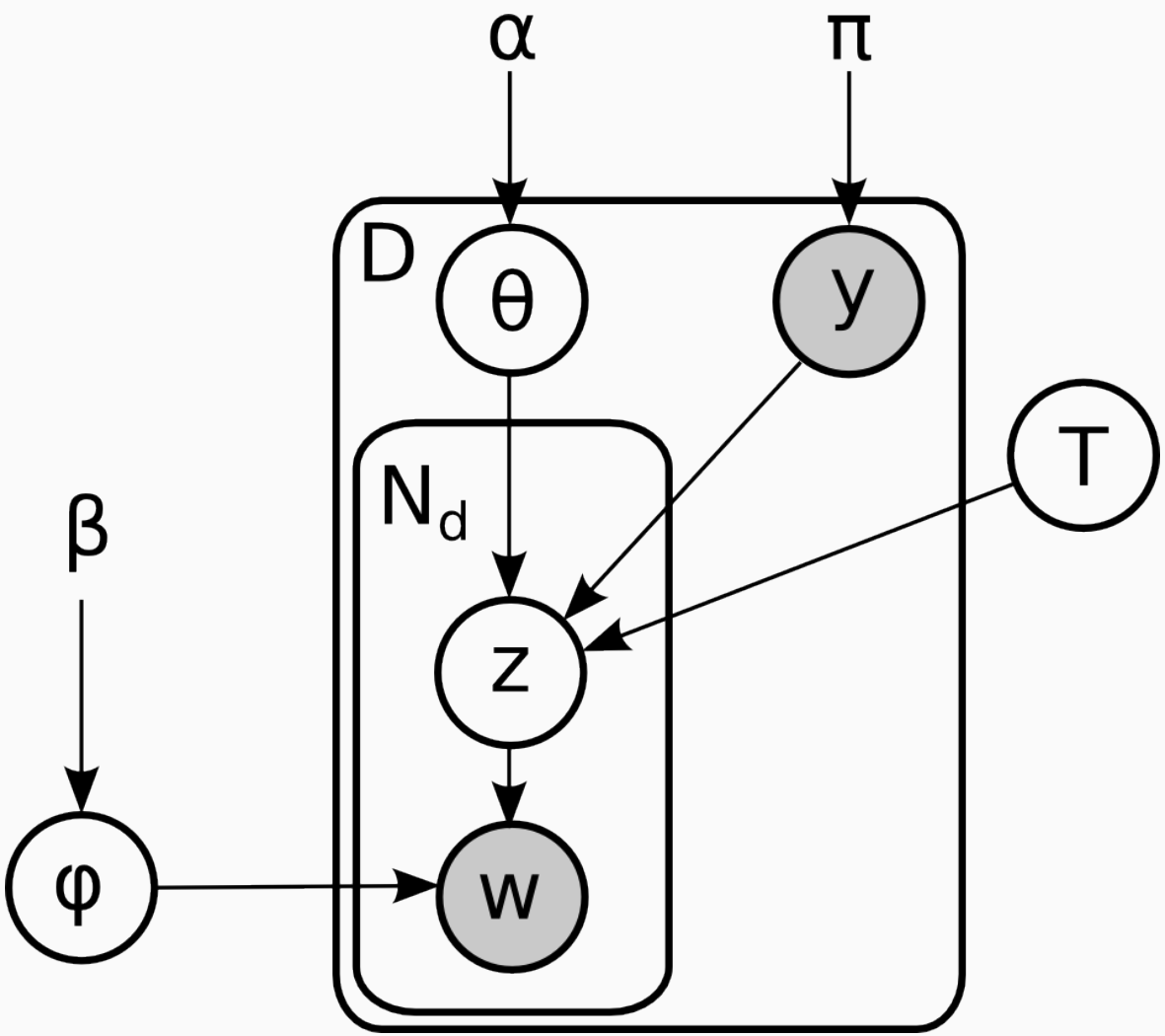
Supervised LDA

- Supervised LDA: документы снабжены дополнительной информацией, дополнительной переменной отклика (обычно известной).
- Распределение отклика моделируется обобщённой линейной моделью (распределением из экспоненциального семейства), параметры которой связаны с полученным в документе распределением тем.
- Т.е. в генеративную модель добавляется ещё один шаг: после того как темы всех слов известны,
  - сгенерировать переменную-отклик $y \sim \mathrm{glm}(\mathbf{z}, \eta, \delta)$, где $\mathbf{z}$ – распределение тем в документе, а η и δ – другие параметры glm.
- К примеру, в контексте рекомендательных систем дополнительный отклик может быть реакцией пользователя.

DiscLDA

- Дискриминативное LDA (DiscLDA), другое расширение модели LDA для документов, снабжённых категориальной переменной y, которая в дальнейшем станет предметом для классификации.
- Для каждой метки класса y в модели DiscLDA вводится линейное преобразование $T^y : \mathbb{R}^K \to \mathbb{R}^L_+$, которое преобразует K-мерное распределение Дирихле θ в смесь L-мерных распределений Дирихле $T^y\theta$.
- В генеративной модели меняется только шаг порождения темы документа z: вместо того чтобы выбирать z по распределению θ, сгенерированному для данного документа,
  - сгенерировать тему z по распределению $T^y\theta$, где T^y – преобразование, соответствующее метке данного документа y.

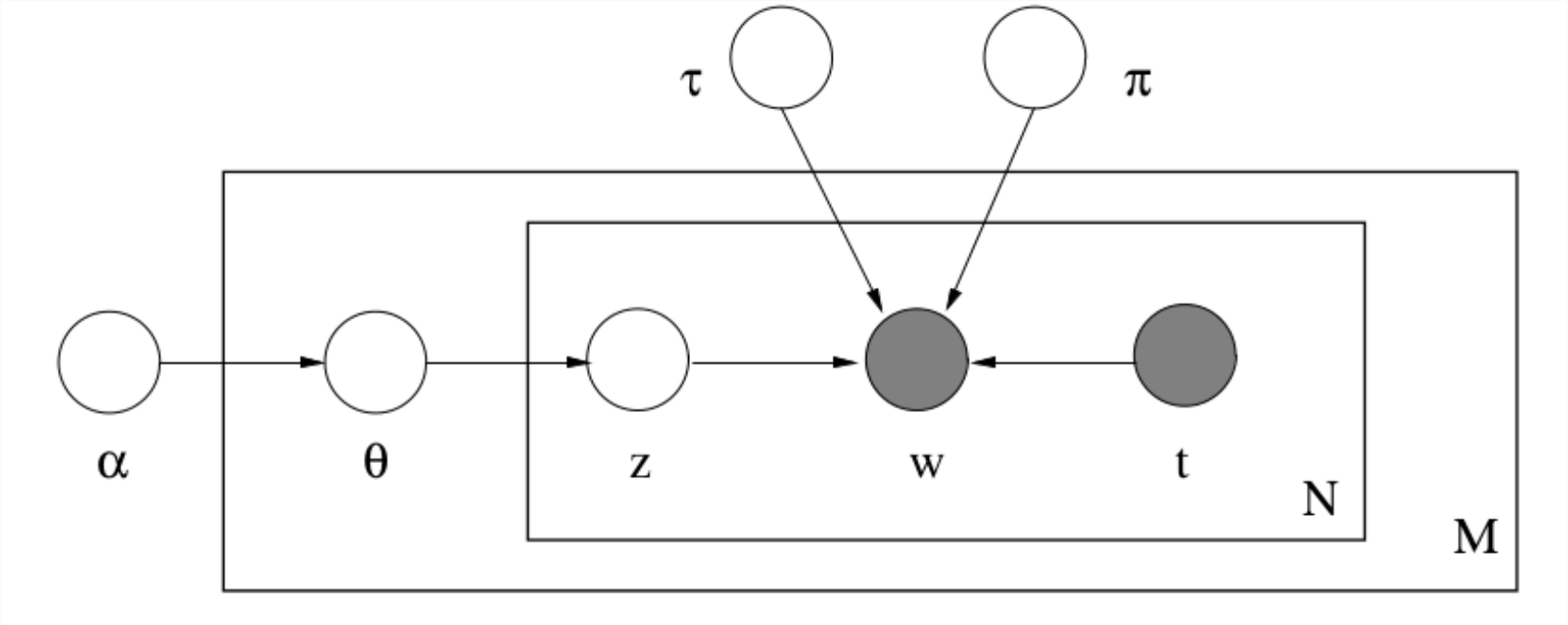
DiscLDA



TagLDA

- TagLDA: слова имеют теги, т.е. документ не является единым мешком слов, а состоит из нескольких мешков, и в разных мешках слова отличаются друг от друга.
- Например, у страницы может быть название – слова из названия важнее для определения темы, чем просто из текста. Или, например, теги к странице, поставленные человеком – опять же, это слова гораздо более важные, чем слова из текста.
- Математически разница в том, что теперь распределения слов в темах – это не просто мультиномиальные дискретные распределения, они факторизованы на распределение слово-тема и распределение слово-тег.

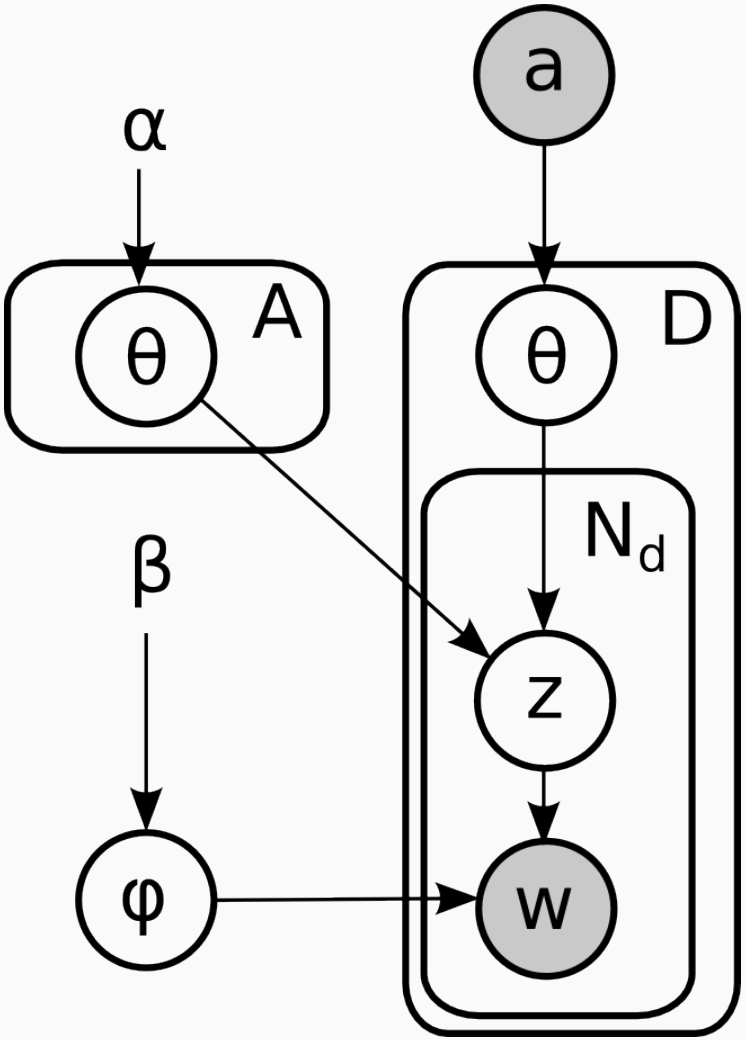
TagLDA



Author-Topic model

- Author-Topic modeling: кроме собственно текстов, присутствуют их авторы; или автор тоже представляется как распределение на темах, на которые он пишет, или тексты одного автора даже на разные темы будут похожи.
- Базовая генеративная модель Author-Topic model (остальное как в базовом LDA):
  - для каждого слова w:
    - выбираем автора x для этого слова из множества авторов документа a_d;
    - выбираем тему из распределения на темах, соответствующего автору x;
    - выбираем слово из распределения слов, соответствующего этой теме.

Author-Topic model



Author-Topic model

- Алгоритм сэмплирования, соответствующий такой модели, является вариантом сжатого сэмплирования по Гиббсу:

$$p(z_w = t, x_w = a \mid \mathbf{z}_{-w}, \mathbf{x}_{-w}, \mathbf{w}, \alpha, \beta) \propto$$

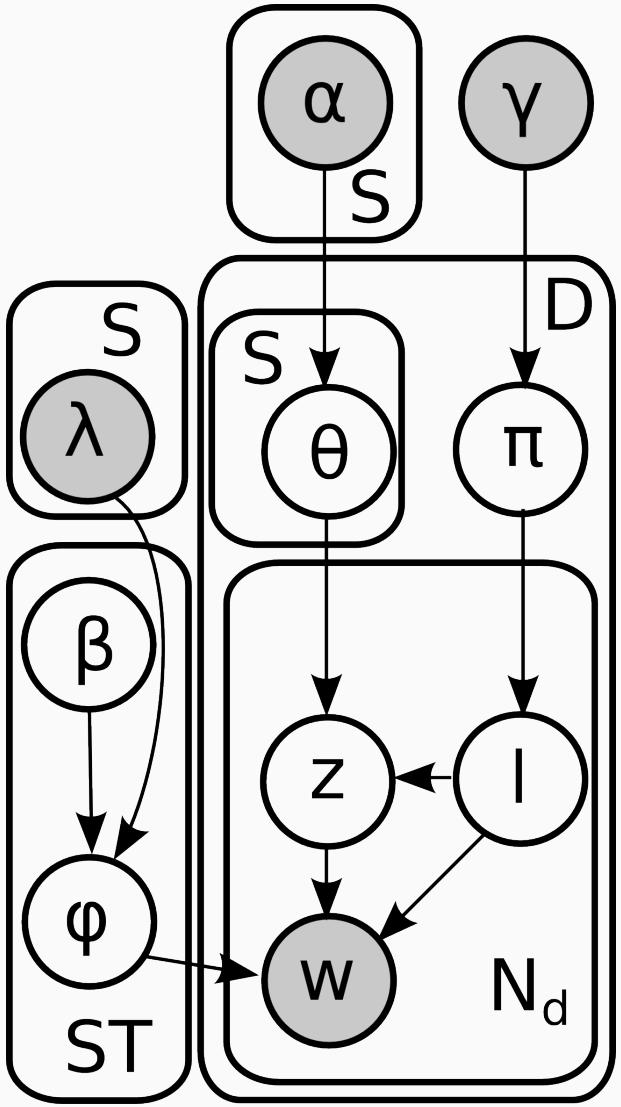
$$\propto \frac{n^{(a)}_{-a,t} + \alpha}{\sum_{t' \in T} \left(n^{(a)}_{-w,t'} + \alpha \right)} \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta \right)},$$

  где $n^{(a)}_{-a,t}$ – то, сколько раз автору a соответствовала тема t, не считая текущего значения x_w, а $n^{(w)}_{-w,t}$ – число раз, которое слово w было порождено из темы t, не считая текущего значения z_w; заметим, что оба этих счётчика зависят от других переменных $\mathbf{z}_{-w}$, $\mathbf{x}_{-w}$.

- Давайте теперь чуть подробнее разберём тематические модели с сентиментом...

Joint Sentiment-Topic

- JST: темы зависят от тональностей из распределения π_d документа, слова зависят от пар тональность-тема.
- Порождающий процесс – для каждой позиции слова j:
  (1) выберем метку тональности $l_j \sim \mathrm{Mult}(\pi_d)$;
  (2) выберем тему $z_j \sim \mathrm{Mult}(\theta_{d,l_j})$;
  (3) выберем слово $w \sim \mathrm{Mult}(\phi_{l_j,z_j})$.



Joint Sentiment-Topic

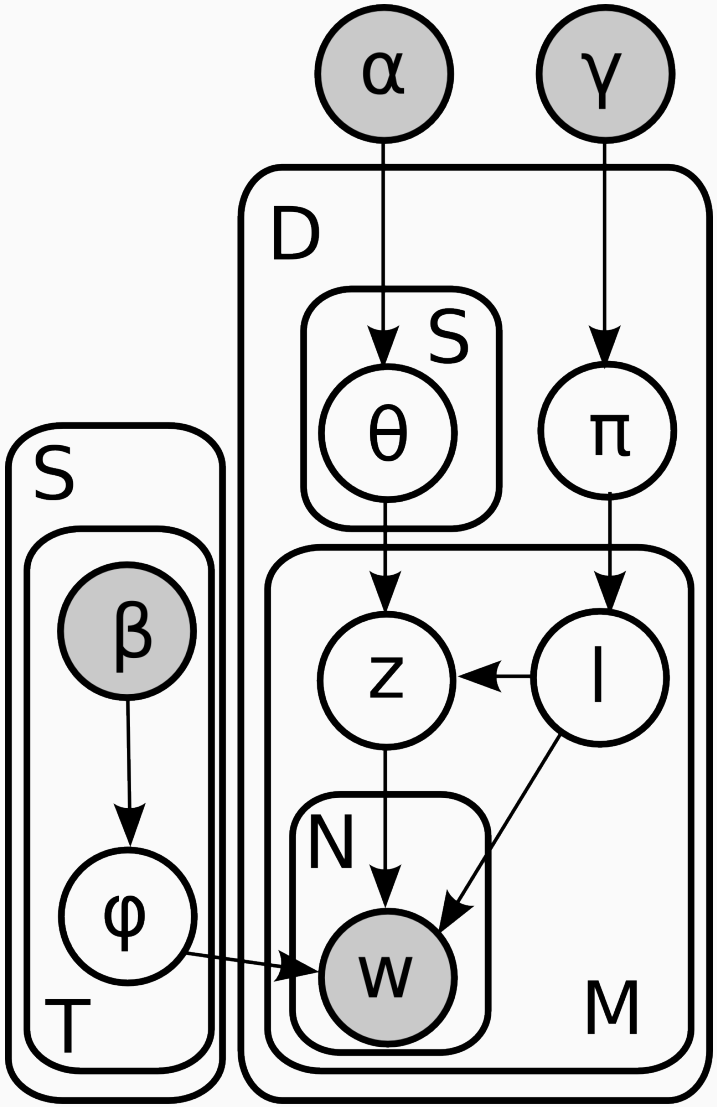
- В сэмплировании по Гиббсу можно выинтегрировать π_d:

$$p(z_j = t, l_j = k \mid \mathbf{z}_{-j}, \mathbf{w}, \alpha, \beta, \gamma, \lambda) \propto$$
$$\frac{n^{\neg j}_{*,k,t,d} + \alpha_{tk}}{n^{\neg j}_{*,k,*,d} + \sum_t \alpha_{tk}} \cdot \frac{n^{\neg j}_{w,k,t,*} + \beta_{kw}}{n^{\neg j}_{*,k,t,*} + \sum_w \beta_{kw}} \cdot \frac{n^{\neg j}_{*,k,*,d} + \gamma}{n^{\neg j}_{*,*,*,d} + S\gamma},$$

где $n_{w,k,t,d}$ — число слов w, порождённых темой t и меткой тональности k в документе d, α_{tk} — априорное распределение Дирихле для темы t с меткой тональности k.

Aspect and Sentiment Unification Model

- ASUM: aspects + sentiment для обзоров пользователей; разбиваем обзор на предложения, предполагая, что в каждом предложении один аспект.
- Базовая модель – Sentence LDA (SLDA): для каждого отзыва d с распределением θ_d, для каждого предложения в d,
  (1) выбираем метку тональности $l_s \sim \mathrm{Mult}(\pi_d)$,
  (2) выбираем тему $t_s \sim \mathrm{Mult}(\theta_{dl_s})$ при условии тональности l_s,
  (3) порождаем слова $w \sim \mathrm{Mult}(\phi_{l_s t_s})$.



Сэмплирование по Гиббсу в ASUM

- Обозначим через $s_{k,t,d}$ число предложений (а не слов), которым присвоена тема t и метка t в документе d:

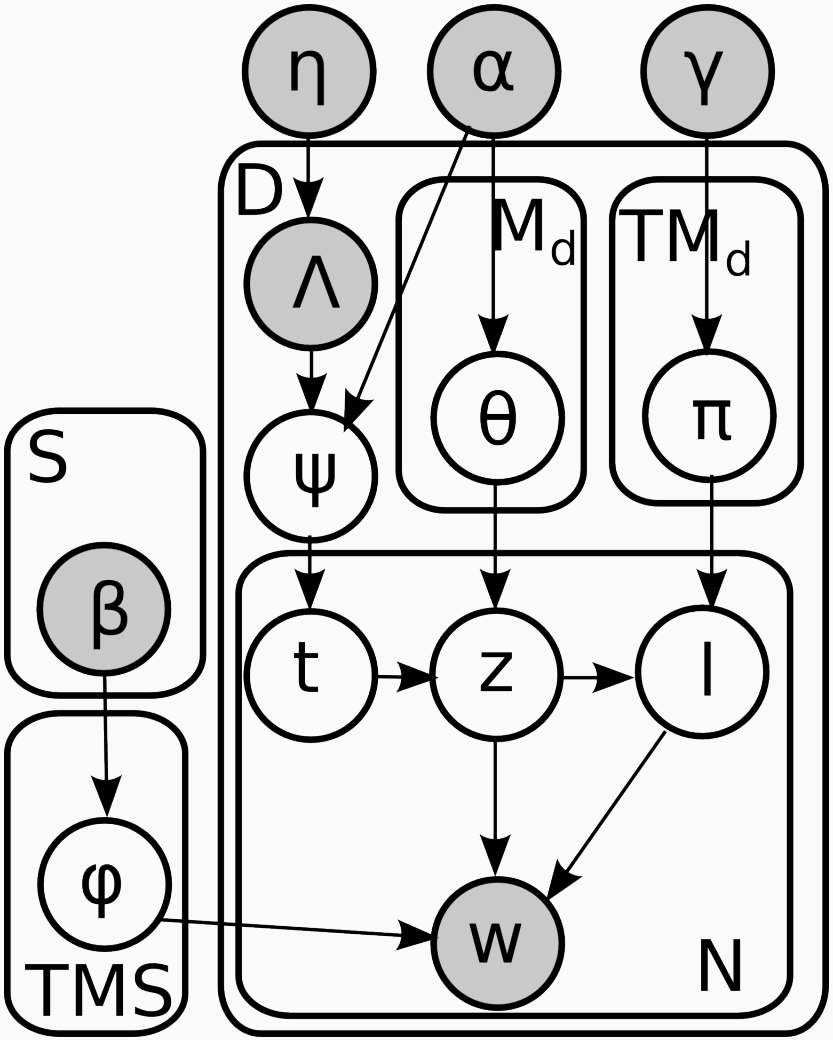
$$p(z_j = t, l_j = k \mid \mathbf{l}_{-j}, \mathbf{z}_{-j}, \mathbf{w}, \gamma, \alpha, \beta) \propto$$
$$\frac{s^{\neg j}_{k,t,d} + \alpha_t}{s^{\neg j}_{k,*,d} + \sum_t \alpha_t} \cdot \frac{s^{\neg j}_{k,*,d} + \gamma_k}{s^{\neg j}_{*,*,d} + \sum_{k'} \gamma_{k'}} \times$$
$$\times \frac{\Gamma\left(n^{\neg j}_{*,k,t,*} + \sum_w \beta_{kw}\right)}{\Gamma\left(n^{\neg j}_{*,k,t,*} + \sum_w \beta_{kw} + W_{*,j}\right)} \prod_w \frac{\Gamma\left(n^{\neg j}_{w,k,t,*} + \beta_{kw} + W_{w,j}\right)}{\Gamma\left(n^{\neg j}_{w,k,t,*} + \beta_{kw}\right)},$$

где $W_{w,j}$ – число слов w в предложении j.

User-Aware Sentiment Topic Models

- USTM: добавим ещё метаданные/теги для пользователя (место, пол, возраст и т.п.) к темам и тональностям.
- Каждый документ снабжён комбинацией тегов, темы порождаются при условии тегов, тональности при условии троек (документ, тег, тема), слова при условии тем, тональностей и тегов.
- Формально, распределение тегов ψ_d порождается для каждого документа (с априорным распределением Дирихле с параметром η), для каждой позиции j порождаем тег $a_j \sim \mathrm{Mult}(\psi_d)$ из ψ_d, а распределения тем, тональностей и слов будут условными по тегу a_j.

Графическая модель USTM



Сэмплирование по Гиббсу для USTM

- Обозначим через $n_{w,k,t,m,d}$ число слов w, порождённых темой t, меткой тональности k и тегом метаданных m в документе d; тогда

$$p(z_j = t, l_j = k, a_j = m \mid \mathbf{l}_{-j}, \mathbf{z}_{-j}, a_{-j}, \mathbf{w}, \gamma, \alpha, \beta) \propto$$
$$\frac{n^{\neg j}_{*,*,t,m,d} + \alpha}{n^{\neg j}_{*,*,*,m,d} + TM_d\alpha} \cdot \frac{n^{\neg j}_{w,*,t,m,*} + \beta}{n^{\neg j}_{*,*,t,m,*} + W\beta} \cdot$$
$$\frac{n^{\neg j}_{w,k,t,m,*} + \beta_{wk}}{n^{\neg j}_{*,k,t,m,*} + \sum_w \beta_{wk}} \cdot \frac{n^{\neg j}_{*,k,t,m,d} + \gamma}{n^{\neg j}_{*,*,t,m,d} + S\gamma},$$

где M_d — число тегов в документе d.

Примеры тем

#	sent.	sentiment words
1	neu	соус, салат, кусочек, сыр, тарелка, овощ, масло, лук, перец
	pos	приятный, атмосфера, уютный, вечер, музыка, ужин, романтический
	neg	ресторан, официант, внимание, сервис, обращать, обслуживать, уровень
2	neu	столик, заказывать, вечер, стол, приходить, место, заранее, встречать
	pos	место, хороший, вкус, самый, приятный, вполне, отличный, интересный
	neg	еда, вообще, никакой, заказывать, оказываться, вкус, ужасный, ничто
3	neu	девушка, спрашивать, вопрос, подходить, официантка, официант, говорить
	pos	большой, место, выбор, хороший, блюдо, цена, порция, небольшой, плюс
	neg	цена, обслуживание, качество, уровень, кухня, средний, ценник, высоко

Примеры окрашенных слов для разных аспектов

aspect	sentiment words
баранина	вкусный, сытный, аппетитный, душистый, деликатесный, сладкий, ароматный, черствый, ароматичный, пресный
караоке	музыкальный, попсовый, классно, развлекательный, улетный
пирог	вкусный, аппетитный, обсыпной, сытный, черствый, ароматный, сладкий
ресторан	шикарный, фешенебельный, уютный, люкс, роскошный, недорогой, шикарно, престижный, модный, развлекательный,
вывеска	обветшалый, выцветший, аляповатый, фешенебельный, фанерный, респектабельный, помпезный, ржавый
администратор	люкс, неисполнительный, ответственный, компетентный, толстяк, высококвалифицированный, высококлассный, толстяк
интерьер	уют, уютны, стильный, просторный, помпезный, роскошный, шикарный, шикарный, мрачноватый, комфортабельный
вежливый	вежливый, учтивы, обходительный, доброжелательный, тактичный

Краткое резюме

1. Классический метод категоризации: наивный байесовский классификатор.
2. Обобщаем наивный байес: кластеризация EM-алгоритмом.
3. Тематическое моделирование: pLSA, LDA, расширения LDA.

Expectation Propagation

Expectation Propagation

- Мы уже много говорили об аппроксимациях, которые работают, когда фактор-граф у модели сложный получается.
- Но что делать, когда сложные сами факторы?
- Давайте зайдём с другой стороны...

Expectation Propagation

- Раньше мы оптимизировали $\mathrm{KL}(q\|p)$, а теперь давайте попробуем $\mathrm{KL}(p\|q)$; для экспоненциального семейства:

$$q(\mathbf{z}) = h(\mathbf{z})g(\eta)e^{-\eta^\top \mathbf{u}(\mathbf{z})},$$
$$\mathrm{KL}(p\|q) = -\ln g(\eta) - \eta^\top \mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})] + \mathrm{const}.$$

- Минимизируем по η, взяв градиент...

Expectation Propagation

- Минимизируем по η, взяв градиент...

$$-\nabla \ln g(\eta) = \mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})].$$

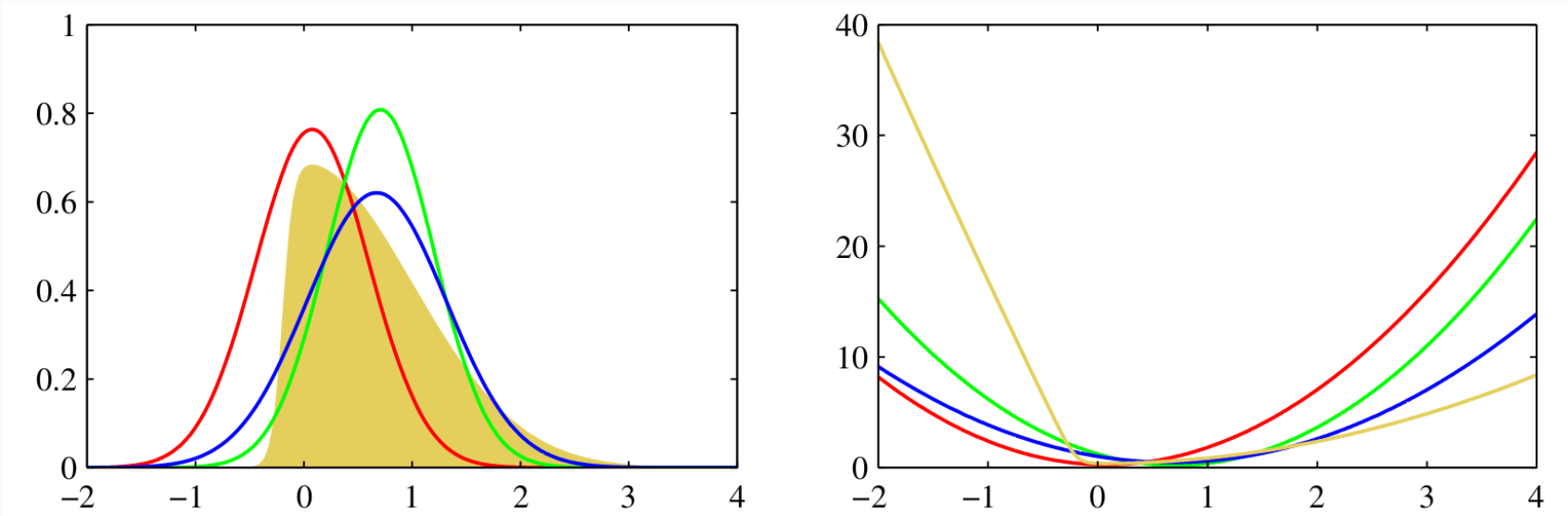
- Но можно проверить, что $-\nabla \ln g(\eta)$ — это ожидание $\mathbf{u}(\mathbf{z})$ по $q(\mathbf{z})$ (упражнение).
- Иначе говоря, получилось, что нужно просто совместить моменты:

$$\mathbb{E}_{q(\mathbf{z})}[\mathbf{u}(\mathbf{z})] = \mathbb{E}_{p(\mathbf{z})}[\mathbf{u}(\mathbf{z})].$$

- Например, для гауссиана $q(\mathbf{z}) = N(\mathbf{z}|\mu, \Sigma)$ надо просто взять среднее и матрицу ковариаций $p(\mathbf{z})$.

Expectation Propagation

- Пример с гауссианой:



- Лапласовское приближение (красным), вариационное (зелёным), EP (синим).
- Почему EP шире вариационного?

Expectation Propagation

- Теперь давайте применим это к сложным вероятностным моделям. Обычно модель раскладывается в произведение:

$$p(D, \theta) = \prod_i f_i(\theta).$$

- Мы бы хотели подсчитать

$$p(\theta \mid D) = \frac{1}{p(D)} \prod_i f_i(\theta), \text{ где } p(D) = \int \prod_i f_i(\theta) \mathrm{d}\theta.$$

- Соответственно, давайте искать приближение в виде

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta).$$

Expectation Propagation

- Мы бы хотели минимизировать

$$\mathrm{KL}(p\|q) = \mathrm{KL}\left(\frac{1}{p(D)}\prod_i f_i(\theta)\|\frac{1}{Z}\prod_i \tilde{f}_i(\theta)\right).$$

- Но аппроксимировать отдельные факторы плохо (почему?).
- Суть метода *Expectation Propagation* — аппроксимировать каждый $\tilde{f}_j(\theta)$ в контексте других $\tilde{f}_i(\theta)$, итеративно.
- Пусть мы сейчас оптимизируем $\tilde{f}_j(\theta)$. Зафиксируем остальные приближения; мы хотим найти наилучший

$$q'(\theta) \propto \tilde{f}_j(\theta)\prod_{i\neq j}\tilde{f}_i(\theta),$$

который аппроксимировал бы

$$q'(\theta) \approx f_j(\theta)\prod_{i\neq j}\tilde{f}_i(\theta).$$

Expectation Propagation

- По сути надо сначала посчитать

$$q_{-j}(\theta) = \frac{q(\theta)}{\tilde{f}_j(\theta)} = \prod_{i \neq j} \tilde{f}_i(\theta).$$

- Потом получить новое распределение

$$\frac{1}{Z_j} f_j(\theta) q_{-j}(\theta), Z_j = \int f_j(\theta) q_{-j}(\theta) \mathrm{d}\theta.$$

- И минимизировать

$$\mathrm{KL}\left(\frac{1}{Z_j} f_j(\theta) q_{-j}(\theta) \| q'(\theta)\right),$$

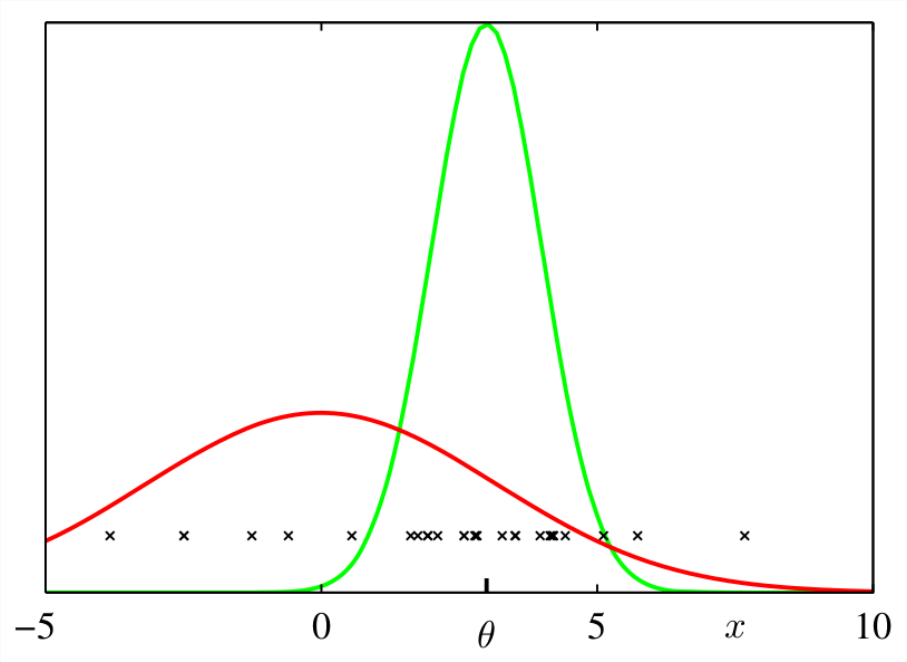
а для этого, как мы уже знаем, достаточно моменты посчитать.

Expectation Propagation

- И тогда просто возьмём

$$\tilde{f}_j(\theta) = \frac{q'(\theta)}{q_{-j}(\theta)}.$$

- Этот процесс теперь можно повторять итеративно, пока не сойдётся.
- Пример — давайте попробуем убрать шум:



Expectation Propagation

- Предположение в том, что у нас смесь гауссианов

$$p(\mathbf{x}|\theta) = (1-w)N(\mathbf{x}|\theta, \mathbf{I}) + wN(\mathbf{x} \mid 0, a\mathbf{I})$$

с априорным распределением

$$p(\theta) = N(\theta|0, b\mathbf{I}).$$

- Тогда совместное распределение будет

$$p(D, \theta) = p(\theta) \prod_{n=1}^{N} p(\mathbf{x}_n|\theta),$$

смесь 2^N гауссианов, и что-то надо делать...

Expectation Propagation

- Факторы, очевидно, $f_0(\theta) = p(\theta)$ и $f_n(\theta) = p(\mathbf{x}_n|\theta)$.
- В качестве приближения берём сферический гауссиан

$$q(\theta) = N(\theta \mid \mathbf{m}, v\mathbf{I}), \text{ то есть}$$
$$\tilde{f}_n(\theta) = s_n N(\theta \mid \mathbf{m}_n, v_n\mathbf{I}).$$

- Как теперь будет работать Expectation Propagation?

Expectation Propagation

- Упражнения:
  1. EP-шаг оставит $f_0(\theta)$ без изменений;
  2. для $f_n(\theta)$ строим $q_{-n}(\theta)$, и получится

$$\begin{aligned}
\mathbf{m}_{-j} &= \mathbf{m} + v_{-n} v_n^{-1} (\mathbf{m} - \mathbf{m}_n), \\
v_{-n}^{-1} &= v^{-1} - v_n^{-1}, \\
Z_n &= (1-w) N(\mathbf{x}_n | \mathbf{m}_{-n}, (v_{-n}+1)\mathbf{I}) + w N(\mathbf{x}_n \mid 0, a\mathbf{I});
\end{aligned}$$

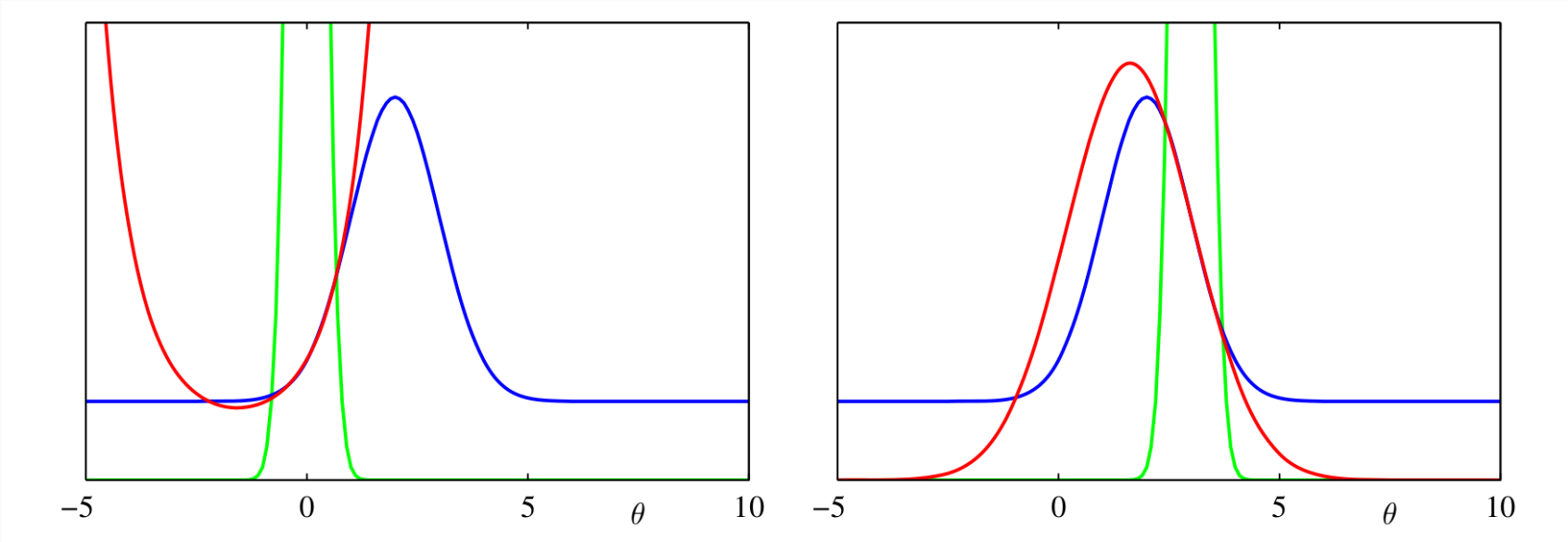
  3. а потом для нового приближения $q'(\theta)$

$$\begin{aligned}
\mathbf{m} &= \mathbf{m}_{-n} + \rho_n \frac{v_{-n}}{v_{-n}+1} (\mathbf{x}_n - \mathbf{m}_{-n}), \\
v &= v_{-n} - \rho_n \frac{v_{-n}^2}{v_{-n}+1} + \rho_n (1-\rho_n) \frac{v_{-n}^2 \|\mathbf{x}_n - \mathbf{m}_{-n}\|^2}{D(v_{-n}+1)^2}, \\
\rho_n &= 1 - \frac{w}{Z_n} N(\mathbf{x}_n | 0, a\mathbf{I});
\end{aligned}$$

  4. ρ_n легко интерпретировать как вероятность того, что $\mathbf{x}_n$ — не шум.

Expectation Propagation

- Вот пример:



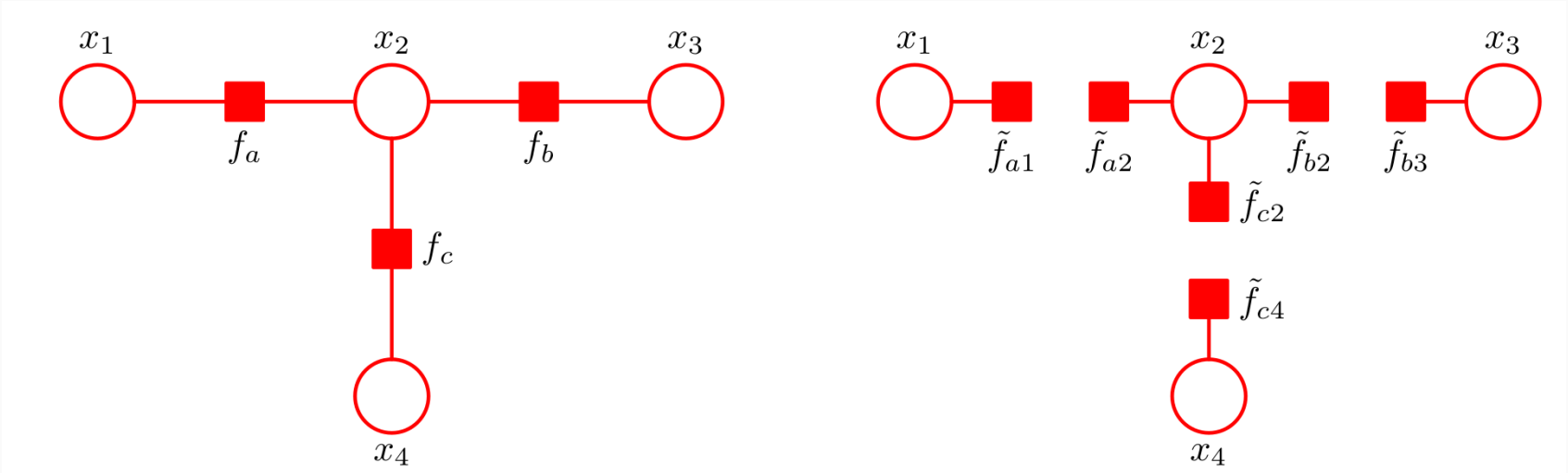
- $f_n(\theta)$ синим, $\tilde{f}_n(\theta)$ красным, $q_{-n}(\theta)$ зелёным.
- $q_{-n}(\theta)$ показывает, где нужно приближать f_n посредством $\tilde{f}_n$.

Expectation Propagation

- И на графах — давайте теперь подмножества θ у f_n тоже будут разные и небольшие.
- Например, приблизим

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$ посредством

$$q(\mathbf{x}) \propto \tilde{f}_{a1}(x_1)\tilde{f}_{a2}(x_2)\tilde{f}_{b2}(x_2)\tilde{f}_{b3}(x_3)\tilde{f}_{c2}(x_2)\tilde{f}_{c4}(x_4) :$$



Expectation Propagation

- Тогда, например, чтобы обновить $\tilde{f}_b(x_2, x_3) = \tilde{f}_{b2}(x_2)\tilde{f}_{b3}(x_3)$, нужно:
  - подсчитать $q_{-b}(\mathbf{x}) = \tilde{f}_{a1}(x_1)\tilde{f}_{a2}(x_2)\tilde{f}_{c2}(x_2)\tilde{f}_{c4}(x_4)$;
  - умножить на точный фактор f_b:

$$\hat{p}(\mathbf{x}) = q_{-b}(\mathbf{x})f_b(x_2, x_3) = \tilde{f}_{a1}(x_1)\tilde{f}_{a2}(x_2)\tilde{f}_{c2}(x_2)\tilde{f}_{c4}(x_4)f_b(x_2, x_3);$$

  - найти $q'(\mathbf{x}) = \arg\min \mathrm{KL}(\hat{p}\|q')$.
- А как минимизировать $\mathrm{KL}(p\|q)$, если q раскладывается на множители?..

- ...минимизироваться будет просто в маргинале по каждому фактору. Здесь:

$$
\begin{aligned}
\hat{p}(x_1) &\propto \tilde{f}_{a1}(x_1), \\
\hat{p}(x_2) &\propto \tilde{f}_{a2}(x_2)\tilde{f}_{c2}(x_2)\sum_{x_3} f_b(x_2, x_3), \\
\hat{p}(x_3) &\propto \sum_{x_2} \tilde{f}_{a2}(x_2)\tilde{f}_{c2}(x_2) f_b(x_2, x_3), \\
\hat{p}(x_4) &\propto \tilde{f}_{c4}(x_4),
\end{aligned}
$$

и $q'(\mathbf{x})$ получится просто перемножением этих маргиналов.

Expectation Propagation

- А новый $\tilde{f}_b(x_2, x_3) = \tilde{f}_{b2}(x_2)\tilde{f}_{b3}(x_3)$ получится делением на $q_{-b}(\mathbf{x})$:

$$\tilde{f}_{b2}(x_2) \propto \sum_{x_3} f_b(x_2, x_3),$$
$$\tilde{f}_{b3}(x_3) \propto \sum_{x_2} \tilde{f}_{a2}(x_2)\tilde{f}_{c2}(x_2) f_b(x_2, x_3).$$

- Ничего не напоминает?..

Expectation Propagation

- ...это же в точности сообщения из алгоритма передачи сообщений!
- Например, $\tilde{f}_{b2}(x_2)$ — это $\mu_{f_b \to x_2}(x_2)$.
- То же самое получится и в общем случае (проверьте!).
- Иначе говоря, для фактор-графов Expectation Propagation работает так: итеративно на каждом шаге посылаем сообщения, каждый раз находя оптимальное приближение к сообщению.

Байесовские рейтинг-системы и TrueSkill

Частичные сравнения

- Рейтинг-система – это модель, которая ранжирует участников (игроков) в единый линейный порядок по данным сравнений небольших подмножеств этих игроков (турниров).
- Более того, результаты турниров зашумлены (отчасти случайны).
- Соответственно, и применяются они в таких ситуациях (пример: контекстная реклама в Bing).

Рейтинг Эло

- Первая известная рейтинг-система, основанная на байсеовском подходе.
- Суть модели:
  - сила игры шахматиста в одной партии – случайная величина;
  - *рейтинг* – это ожидание этой величины; мы пытаемся оценить это ожидание;
  - исходная модель Эло – нормальное распределение силы игры вокруг рейтинга.

Рейтинг Эло

- Значит, сила игры в конкретной партии распределена как

$$p(x) = N(x; s, \beta) = \frac{1}{\beta\sqrt{2\pi}} e^{-\frac{1}{2\beta^2}(x-s)^2}.$$

- Сила игры задаётся двумя параметрами: средним s (собственно рейтингом) и дисперсией β^2.
- Эло предположил, что дисперсия β^2 постоянна (и даже от игрока не зависит), а среднее – это как раз рейтинг, который мы пытаемся оценить.

Рейтинг Эло

- Значит, математически говоря, мы ищем

$$\arg\max_{s,\beta^2} p(s, \beta^2 \mid D) = \arg\max_{s,\beta^2} \frac{p(D \mid s, \beta^2)p(s, \beta^2)}{p(D)} = \\ = \arg\max_{s,\beta^2} p(D \mid s, \beta^2)p(s, \beta^2).$$

- Как мы знаем, нормальное распределение является самосопряжённым, поэтому если сила игры нормально распределена вокруг рейтинга, то логично взять нормальное распределение как априорное для рейтинга.
- Таким образом, рейтинг игрока складывается из двух чисел: его среднего значения μ и дисперсии σ^2.
- Значение μ отображается в таблице рейтингов, а σ^2 показывает, насколько достоверна имеющаяся оценка.

Рейтинг Эло

- Предположим, что встречаются два игрока с некоторыми априорными распределениями на рейтинги $N(s_1; \mu_1, \sigma_1^2)$ и $N(s_2; \mu_2, \sigma_2^2)$.
- Тогда сила игры каждого из них в этой конкретной партии имеет распределение

$$p(x \mid \mu, \sigma) = \int_{-\infty}^{\infty} p(x \mid s) p(s \mid \mu, \sigma) ds =$$

$$= \int_{-\infty}^{\infty} N(x; s, \beta) N(s; \mu, \sigma) ds =$$

$$= \int_{-\infty}^{\infty} \frac{1}{\beta\sqrt{2\pi}} e^{-\frac{1}{2\beta^2}(x-s)^2} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(s-\mu)^2} ds = N(x; \mu_x, \sigma_x).$$

то есть мы снова приходим к нормальному распределению, но с другими параметрами.

Рейтинг Эло

- Задача обучения заключается в том, чтобы после новой партии принять во внимание её результат и пересчитать рейтинги.
- Эло разработал специальные аппроксимации и очень простые алгоритмы для этого случая (через «ожидаемые очки в турнире»), чтобы каждый шахматист мог сам на калькуляторе свой рейтинг посчитать, но они нас сейчас не очень интересуют.

Рейтинг-система TrueSkill

- Рейтинг-система TrueSkill была разработана в Microsoft Research для игровых серверов Xbox 360.
- Постановка задачи теперь становится максимально общей.
- Система TrueSkill вычисляет рейтинги игроков, которые объединяются в команды разного размера и участвуют в матчах (турнирах) с несколькими участниками.
- Задача — после каждого из таких турниров пересчитать апостериорные рейтинги.

Рейтинг-система TrueSkill

- Начнём понемногу «разворачивать» то, что происходит в каждом из этих турниров.
- Во-первых, мы не знаем достоверных априорных значений рейтингов, у нас есть только некоторое априорное распределение (мы считаем его нормальным)

$$f(s_i) = N(s; \mu_i, \sigma_i).$$

- Здесь μ_i — это собственно рейтинг игрока, а σ_i — «показатель достоверности» рейтинга, дисперсия.

Рейтинг-система TrueSkill

- Каждый «истинный» скилл является средним значением, вокруг которого распределены конкретные показатели силы игры того или иного игрока в данной конкретной партии (p_i, performance):

$$f(p_i \mid s_i) = N(p_i; s_i, \beta^2).$$

- В системе TrueSkill (как в рейтинге Эло) делается предположение, что β^2 – универсальная константа.
- Тогда p_i через исходные параметры выражается как

$$f(p_i \mid \mu_i, \sigma_i) = \int_{-\infty}^{\infty} N(p_i; s_i, \beta^2) N(s_i; \mu_i, \sigma_i) ds_i.$$

Рейтинг-система TrueSkill

- Затем показатели силы игры игроков в конкретных партиях объединяются и дают оценки на силу игры команд.
- В системе TrueSkill предполагается, что сила команды равна сумме сил её игроков:

$$t_i = \sum_i p_i.$$

Рейтинг-система TrueSkill

- После этого показатели силы команд в данном турнире нужно сравнить друг с другом; их сравнение и должно порождать тот порядок, который записан в результатах турнира.
- Будем считать, что ничья между командами с силой t_1 и t_2 означает, что

$$|t_1 - t_2| < \epsilon$$

для некоторого ϵ (тоже универсальная константа).

Рейтинг-система TrueSkill

- Мы должны подсчитать апостериорные рейтинги команд после получения данных.
- Данные приходят к нам в виде перестановки команд π: упорядоченных результатов турнира (в которых могут быть ничьи между соседними командами).
- Иначе говоря, нужно подсчитать

$$p(\mathbf{s} \mid \pi) = \frac{p(\pi|\mathbf{s})p(\mathbf{s})}{\int p(\pi|\mathbf{s})p(\mathbf{s})d\mathbf{s}}.$$

Рейтинг-система TrueSkill

- В нашей системе присутствуют, кроме s_i и π, ещё переменные p_i, t_i и d_i, причём плотность распределения всей системы мы только что представили в виде произведения распределений:

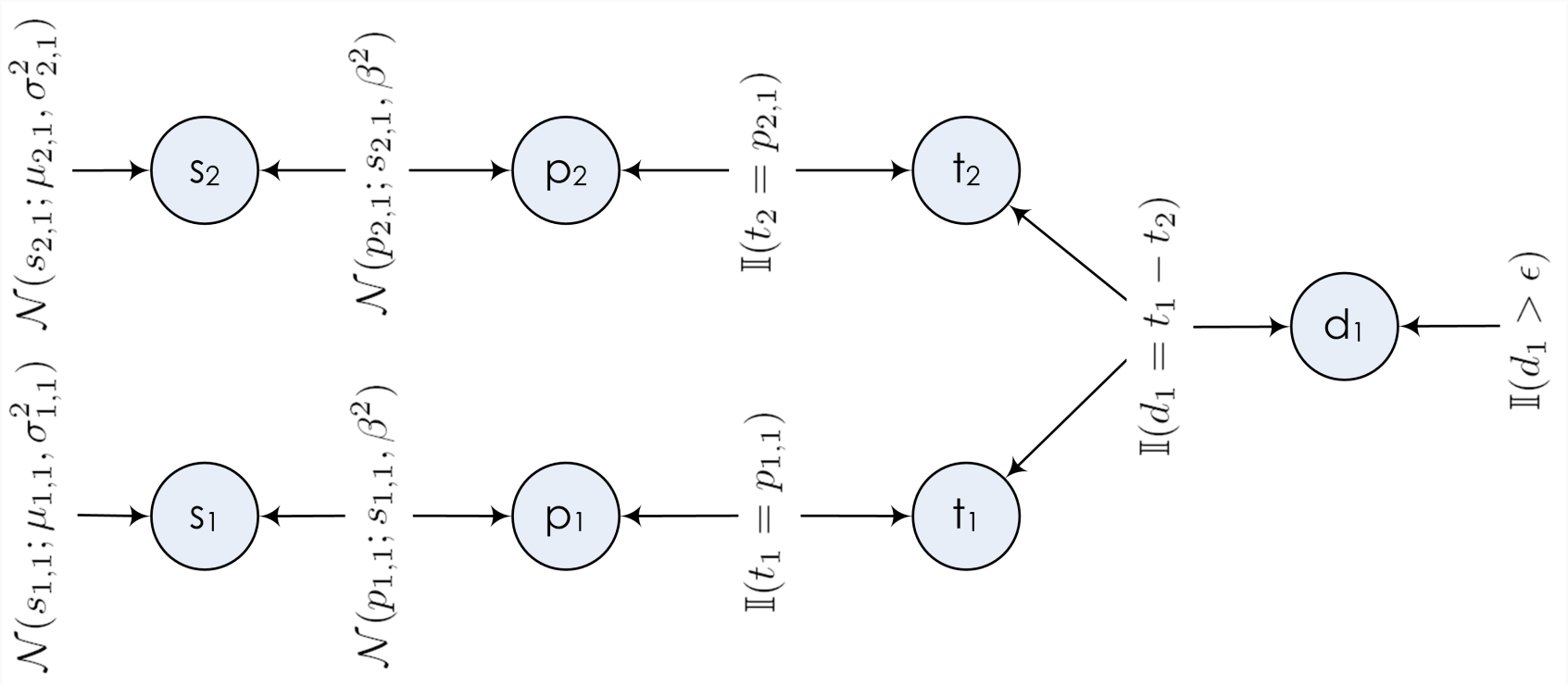
$$p(\pi, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s}) = p(\pi \mid \mathbf{d})p(\mathbf{d} \mid \mathbf{t})p(\mathbf{t} \mid \mathbf{p})p(\mathbf{p} \mid \mathbf{s})p(\mathbf{s}).$$

- А нам нужно вычислить

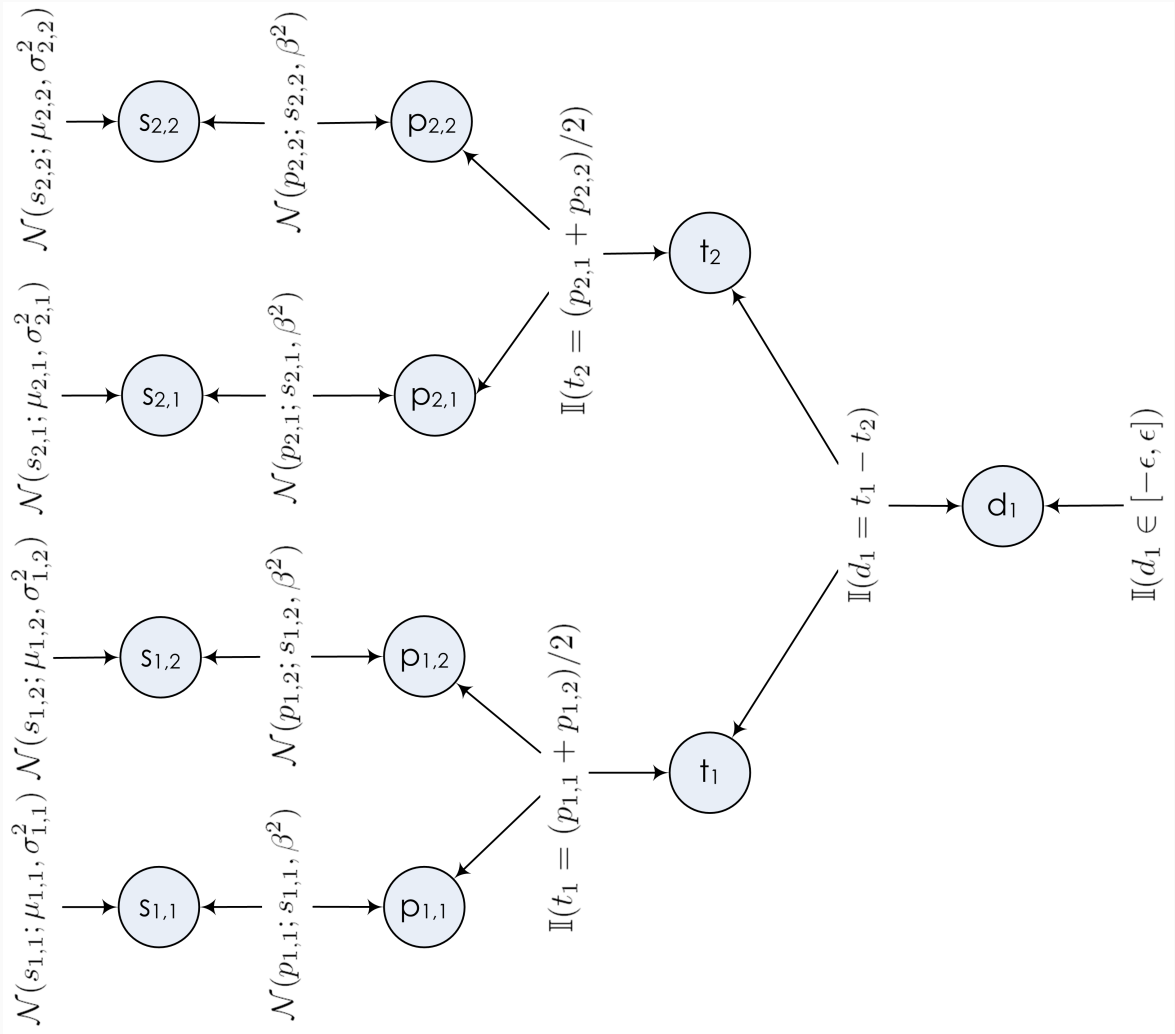
$$p(\pi|\mathbf{s}) = \int\int\int p(\pi, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s}) d\mathbf{d} d\mathbf{t} d\mathbf{p}.$$

- Получили обычную задачу маргинализации.

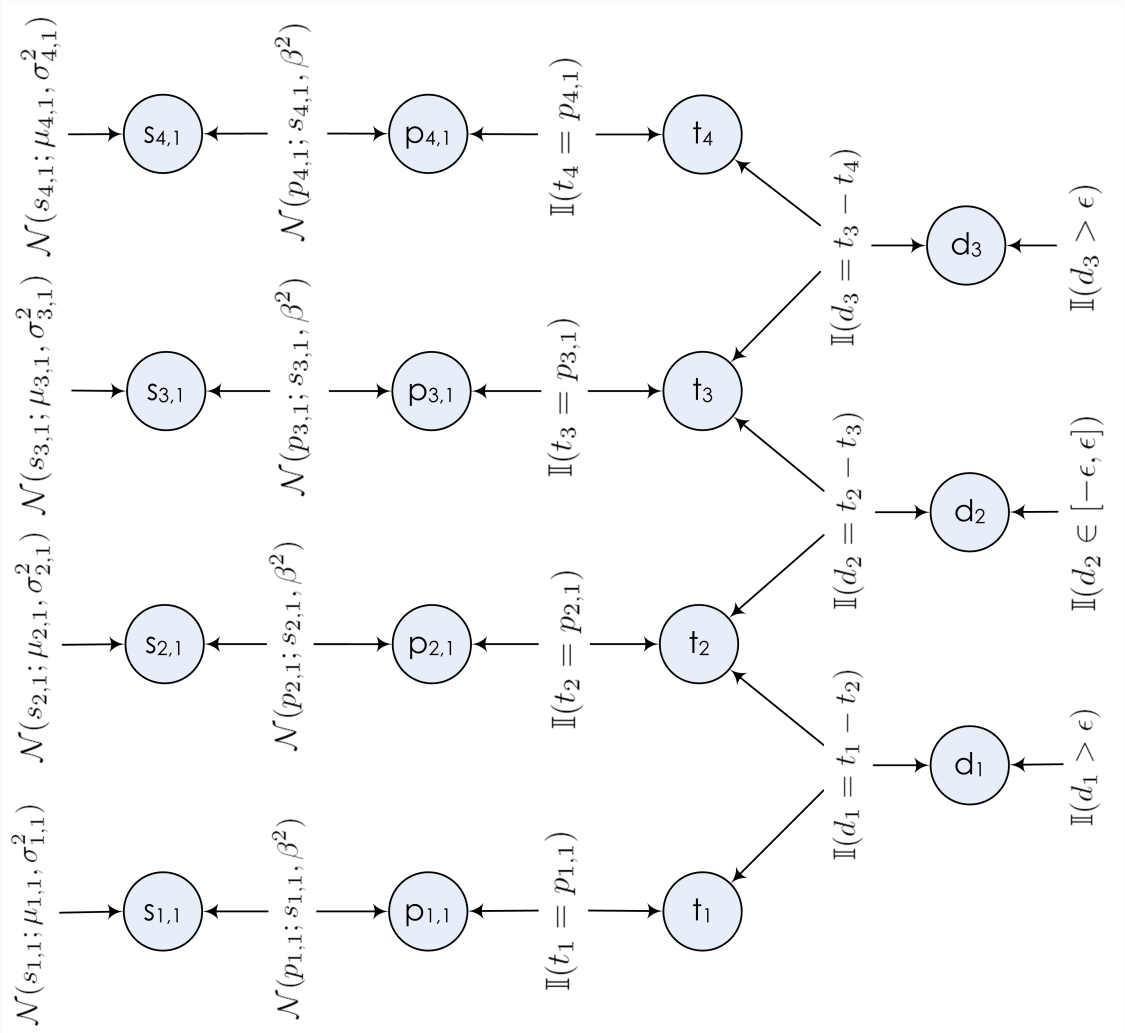
Матч двух игроков



Две команды по два игрока



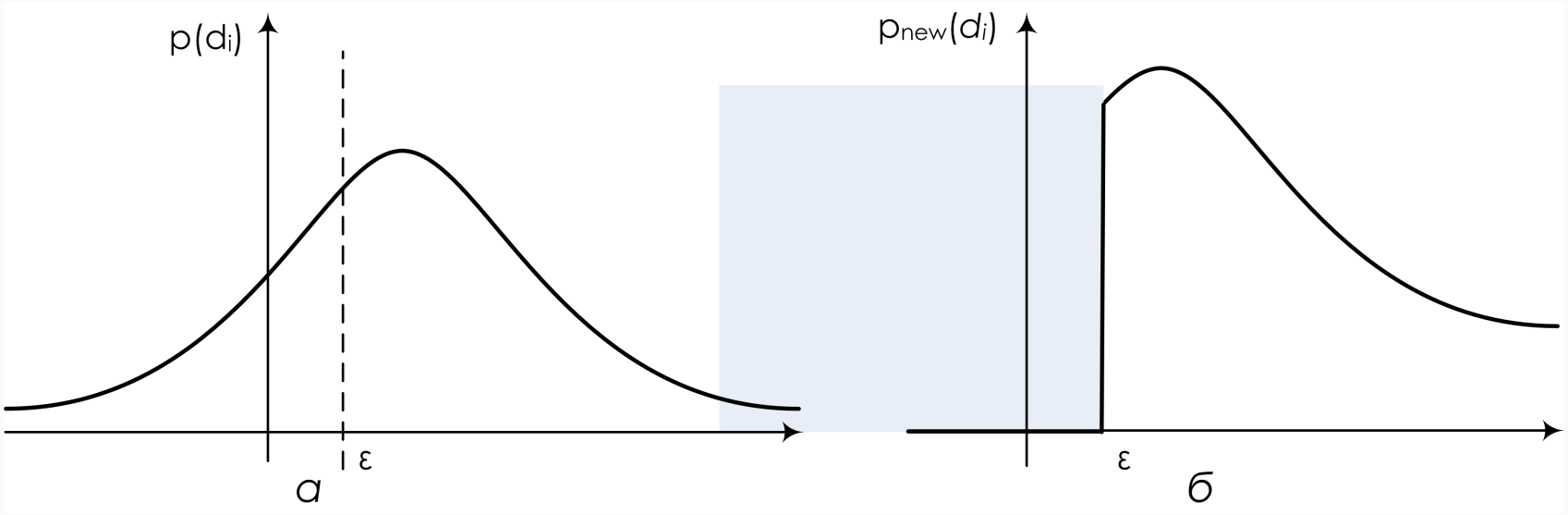
Матч четырёх игроков



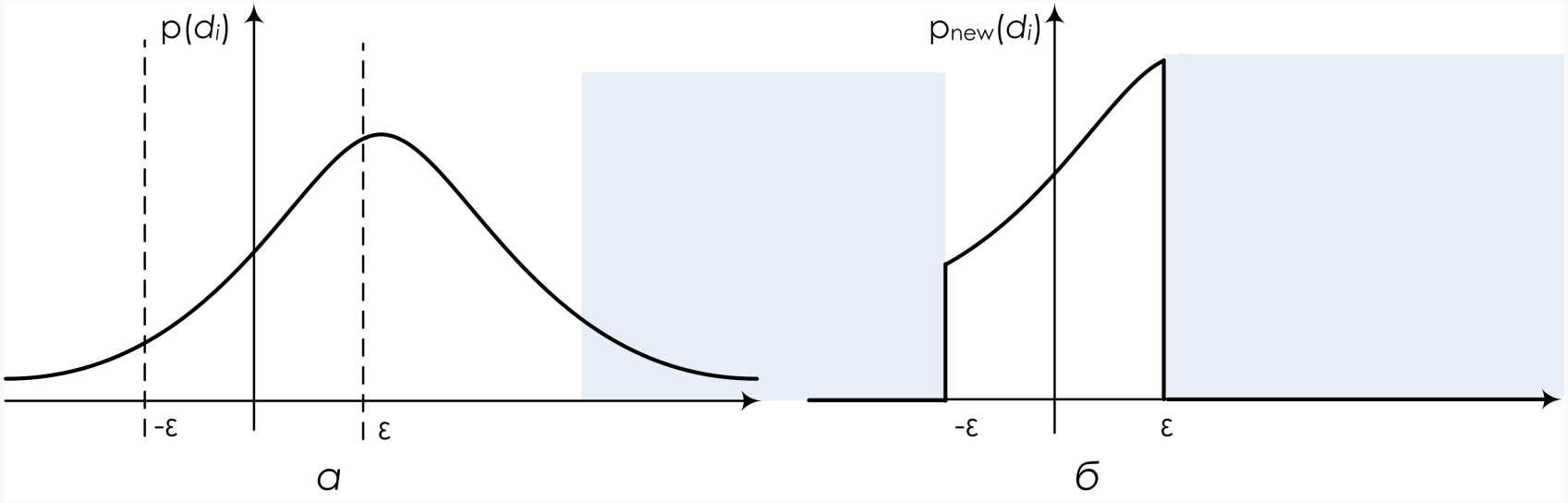
Приближённый вывод

- Казалось бы, граф – дерево, что ещё говорить.
- Но возникает проблема с выводом на нижнем уровне графа.
- Что делает функция, которая собственно определяет победу или ничью?

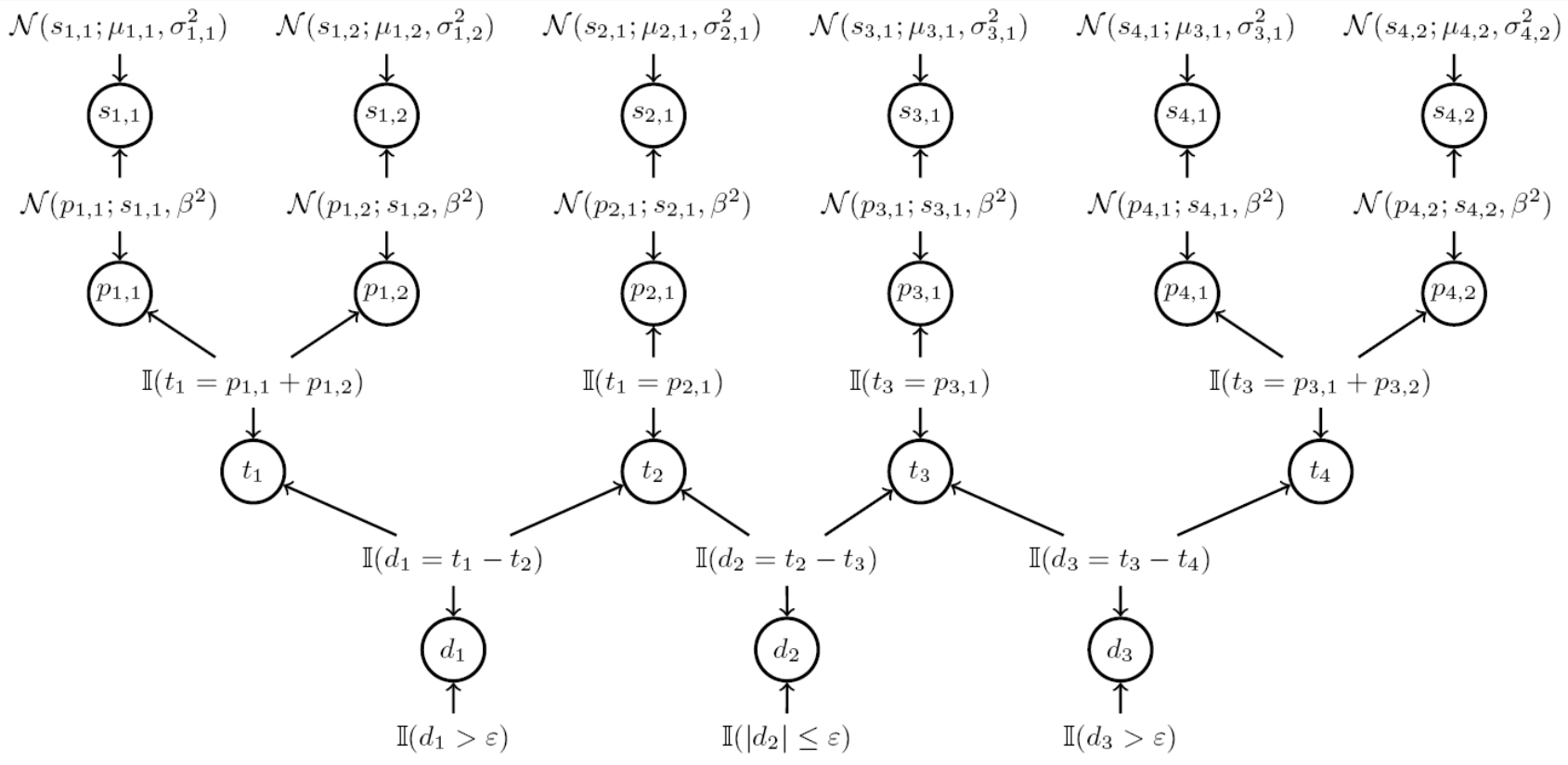
Победа



Ничья



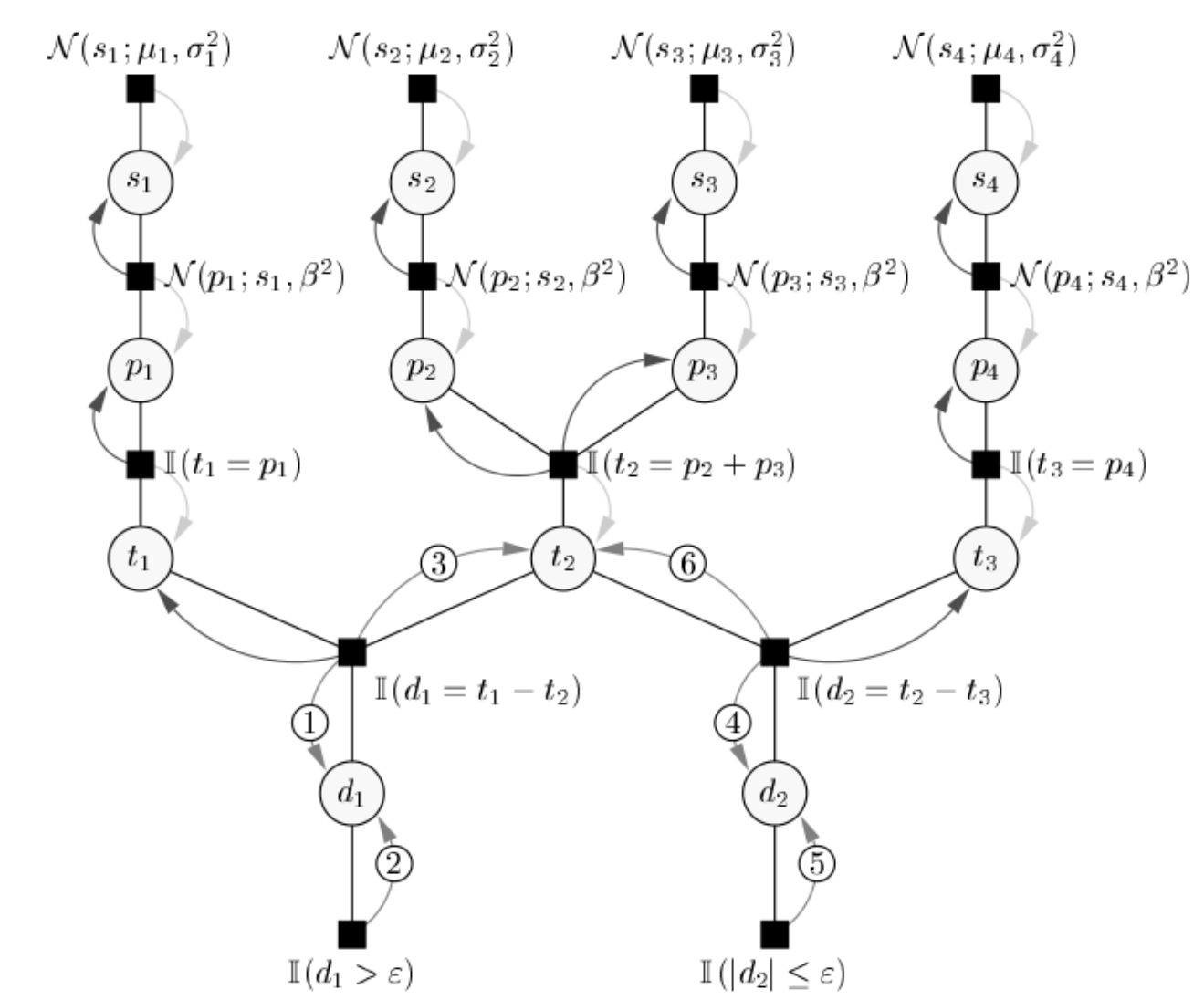
Общий случай



Приближённый вывод

- Для вывода на нижнем уровне графа применяется алгоритм Expectation Propagation [Minka, 2001]:
  - приближаем сообщение от функции к переменной (т.е. распределение p) некоторым семейством распределений $q(\lambda)$;
  - передаём сообщения взад-вперёд, пока оценки не сойдутся.
- В качестве приближённого семейства будем рассматривать семейство нормальных распределений. Тогда для поиска оптимального приближения надо просто найти ожидание и дисперсию (первые два момента) распределения p.
- Для таких «обрезанных» распределений это можно сделать явно, получатся конкретные формулы для передачи сообщений.

Пример


$\mathcal{N}(s_1;\mu_1,\sigma_1^2)$
$\mathcal{N}(s_2;\mu_2,\sigma_2^2)$
$\mathcal{N}(s_3;\mu_3,\sigma_3^2)$
$\mathcal{N}(s_4;\mu_4,\sigma_4^2)$
s_1
s_2
s_3
s_4
$\mathcal{N}(p_1;s_1,\beta^2)$
$\mathcal{N}(p_2;s_2,\beta^2)$
$\mathcal{N}(p_3;s_3,\beta^2)$
$\mathcal{N}(p_4;s_4,\beta^2)$
p_1
p_2
p_3
p_4
$\mathbb{I}(t_1 = p_1)$
$\mathbb{I}(t_2 = p_2 + p_3)$
$\mathbb{I}(t_3 = p_4)$
t_1
t_2
t_3
3
6
$\mathbb{I}(d_1 = t_1 - t_2)$
$\mathbb{I}(d_2 = t_2 - t_3)$
1
4
d_1
d_2
2
5
$\mathbb{I}(d_1 > \varepsilon)$
$\mathbb{I}(|d_2| \leq \varepsilon)$

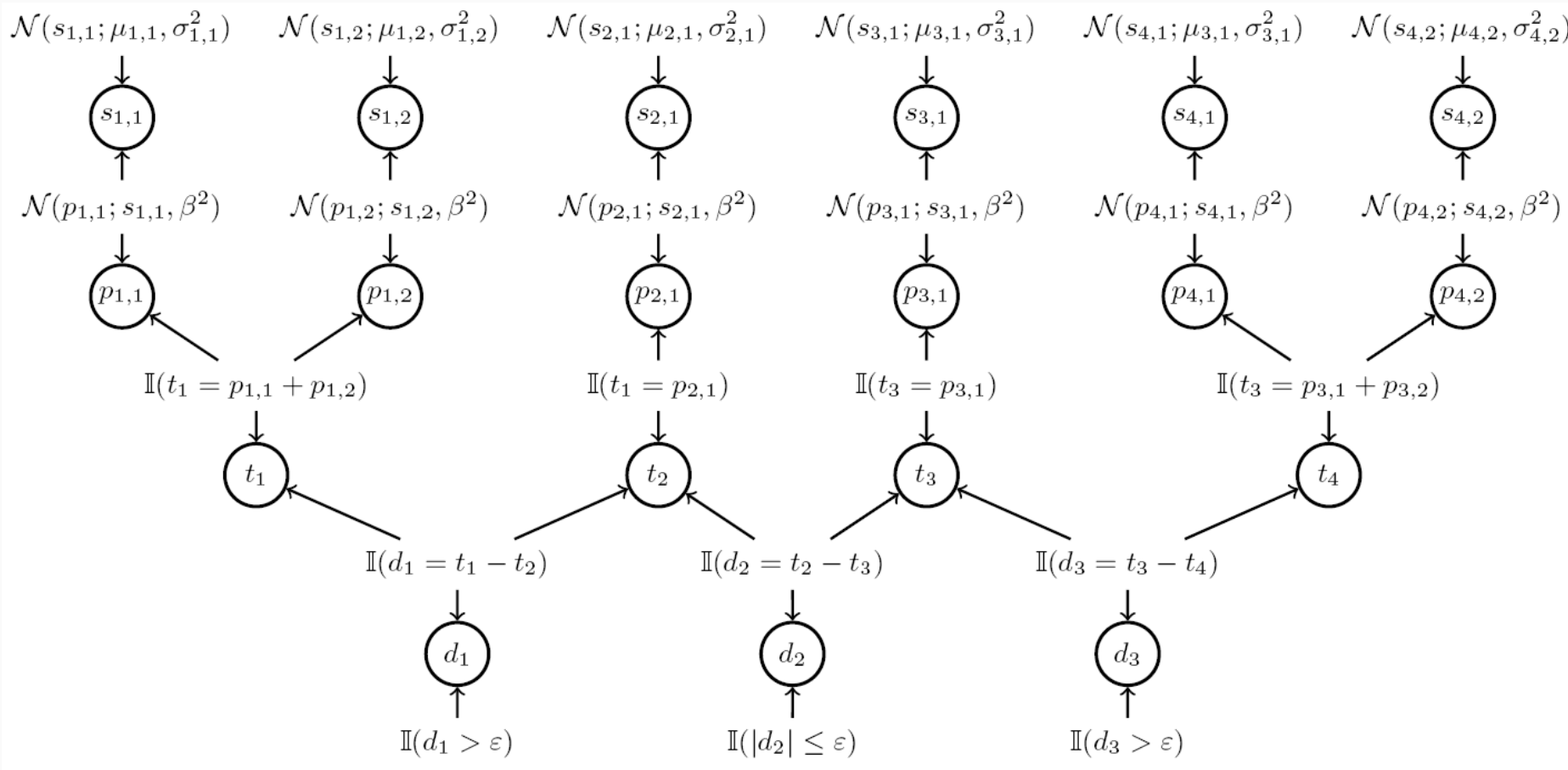
Проблемы TrueSkill

- У TrueSkill есть проблемы, с которыми мы столкнулись, когда попытались применить её на практике.
- Мы хотели сделать рейтинг спортивного «Что? Где? Когда?»:
  - участвуют команды по ≤ 6 человек, причём часто встречаются неполные команды;
  - игроки постоянно переходят между командами (поэтому TrueSkill);
  - в одном турнире могут участвовать до тысячи команд (синхронные турниры);
  - командам задаётся фиксированное число вопросов (36, 60, 90), т.е. в крупных турнирах очень много команд делят одно и то же место.

Проблемы TrueSkill

- У системы TrueSkill при этом тут же начинаются проблемы.
- Главная проблема – большие ничьи на много команд.
- Вторая проблема – сила команды как сумма сил игроков.

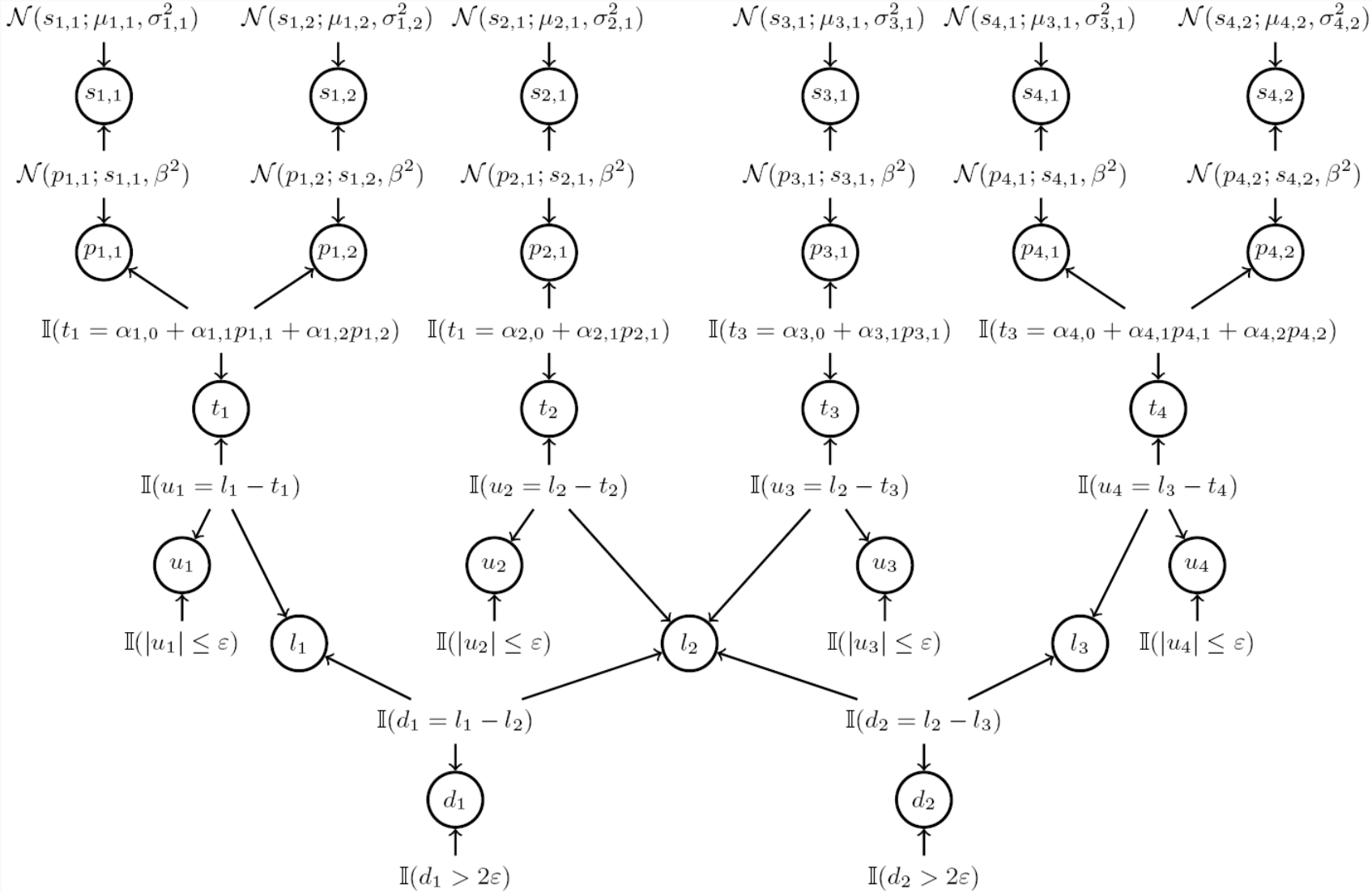
Пример


$\mathcal{N}(s_{1,1};\mu_{1,1},\sigma^2_{1,1})$
$\mathcal{N}(s_{1,2};\mu_{1,2},\sigma^2_{1,2})$
$\mathcal{N}(s_{2,1};\mu_{2,1},\sigma^2_{2,1})$
$\mathcal{N}(s_{3,1};\mu_{3,1},\sigma^2_{3,1})$
$\mathcal{N}(s_{4,1};\mu_{3,1},\sigma^2_{3,1})$
$\mathcal{N}(s_{4,2};\mu_{4,2},\sigma^2_{4,2})$
$s_{1,1}$
$s_{1,2}$
$s_{2,1}$
$s_{3,1}$
$s_{4,1}$
$s_{4,2}$
$\mathcal{N}(p_{1,1};s_{1,1},\beta^2)$
$\mathcal{N}(p_{1,2};s_{1,2},\beta^2)$
$\mathcal{N}(p_{2,1};s_{2,1},\beta^2)$
$\mathcal{N}(p_{3,1};s_{3,1},\beta^2)$
$\mathcal{N}(p_{4,1};s_{4,1},\beta^2)$
$\mathcal{N}(p_{4,2};s_{4,2},\beta^2)$
$p_{1,1}$
$p_{1,2}$
$p_{2,1}$
$p_{3,1}$
$p_{4,1}$
$p_{4,2}$
$\mathbb{I}(t_1 = p_{1,1} + p_{1,2})$
$\mathbb{I}(t_1 = p_{2,1})$
$\mathbb{I}(t_3 = p_{3,1})$
$\mathbb{I}(t_3 = p_{3,1} + p_{3,2})$
t_1
t_2
t_3
t_4
$\mathbb{I}(d_1 = t_1 - t_2)$
$\mathbb{I}(d_2 = t_2 - t_3)$
$\mathbb{I}(d_3 = t_3 - t_4)$
d_1
d_2
d_3
$\mathbb{I}(d_1 > \varepsilon)$
$\mathbb{I}(|d_2| \leq \varepsilon)$
$\mathbb{I}(d_3 > \varepsilon)$

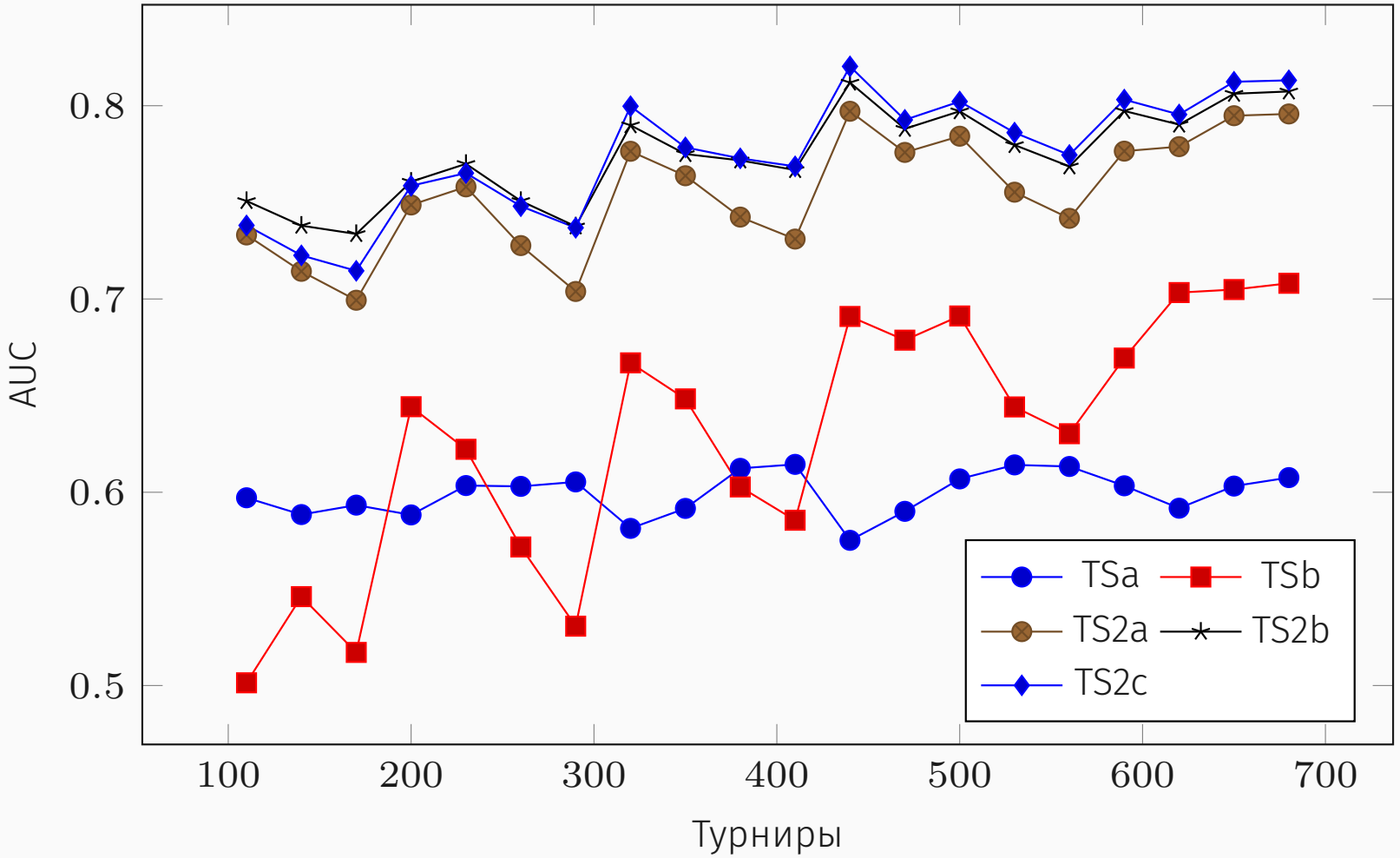
Проблемы TrueSkill

- [Nikolenko, Sirotkin, 2011]: изменив структуру factor graph'а, получилось решить проблему с дележом мест.
- Проблема с силой команд, конечно, должна решаться индивидуально в каждом конкретном приложении.

Пример


$\mathcal{N}(s_{1,1};\mu_{1,1},\sigma^2_{1,1})$
$\mathcal{N}(s_{1,2};\mu_{1,2},\sigma^2_{1,2})$
$\mathcal{N}(s_{2,1};\mu_{2,1},\sigma^2_{2,1})$
$\mathcal{N}(s_{3,1};\mu_{3,1},\sigma^2_{3,1})$
$\mathcal{N}(s_{4,1};\mu_{3,1},\sigma^2_{3,1})$
$\mathcal{N}(s_{4,2};\mu_{4,2},\sigma^2_{4,2})$
$s_{1,1}$
$s_{1,2}$
$s_{2,1}$
$s_{3,1}$
$s_{4,1}$
$s_{4,2}$
$\mathcal{N}(p_{1,1};s_{1,1},\beta^2)$
$\mathcal{N}(p_{1,2};s_{1,2},\beta^2)$
$\mathcal{N}(p_{2,1};s_{2,1},\beta^2)$
$\mathcal{N}(p_{3,1};s_{3,1},\beta^2)$
$\mathcal{N}(p_{4,1};s_{4,1},\beta^2)$
$\mathcal{N}(p_{4,2};s_{4,2},\beta^2)$
$p_{1,1}$
$p_{1,2}$
$p_{2,1}$
$p_{3,1}$
$p_{4,1}$
$p_{4,2}$
$\mathbb{I}(t_1 = \alpha_{1,0} + \alpha_{1,1}p_{1,1} + \alpha_{1,2}p_{1,2})$
$\mathbb{I}(t_1 = \alpha_{2,0} + \alpha_{2,1}p_{2,1})$
$\mathbb{I}(t_3 = \alpha_{3,0} + \alpha_{3,1}p_{3,1})$
$\mathbb{I}(t_3 = \alpha_{4,0} + \alpha_{4,1}p_{4,1} + \alpha_{4,2}p_{4,2})$
t_1
t_2
t_3
t_4
$\mathbb{I}(u_1 = l_1 - t_1)$
$\mathbb{I}(u_2 = l_2 - t_2)$
$\mathbb{I}(u_3 = l_2 - t_3)$
$\mathbb{I}(u_4 = l_3 - t_4)$
u_1
u_2
u_3
u_4
$\mathbb{I}(|u_1| \le \varepsilon)$
$\mathbb{I}(|u_2| \le \varepsilon)$
$\mathbb{I}(|u_3| \le \varepsilon)$
$\mathbb{I}(|u_4| \le \varepsilon)$
l_1
l_2
l_3
$\mathbb{I}(d_1 = l_1 - l_2)$
$\mathbb{I}(d_2 = l_2 - l_3)$
d_1
d_2
$\mathbb{I}(d_1 > 2\varepsilon)$
$\mathbb{I}(d_2 > 2\varepsilon)$

Экспериментальные результаты



Средний AUC по скользящему окну в 50 турниров.

Спасибо за внимание!