Тематическое моделирование

Сергей Николенко

DataFest^2, 6 марта 2016 г.

Outline

Категоризация текстов

- Классическая задача машинного обучения и information retrieval – категоризация текстов.
- Дан набор текстов, разделённый на категории. Нужно обучить модель и потом уметь категоризовать новые тексты.
- Атрибуты $w_1, w_2, \ldots, w_n$ – это слова, v – тема текста (или атрибут вроде «спам / не спам»).
- Bag-of-words model: забываем про порядок слов, составляем словарь. Теперь документ – это вектор, показывающий, сколько раз каждое слово из словаря в нём встречается.

Naive Bayes

- Заметим, что даже это – сильно упрощённый взгляд: для слов ещё довольно-таки важен порядок, в котором они идут...
- Но и это ещё не всё: получается, что $p(w_1, w_2, \ldots, w_n | x = v)$ – это вероятность *в точности такого набора слов* в сообщениях на разные темы. Очевидно, такой статистики взять неоткуда.
- Значит, надо дальше делать упрощающие предположения.
- Наивный байесовский классификатор – самая простая такая модель: давайте предположим, что все слова в словаре условно независимы при условии данной категории.

Naive Bayes

- Иначе говоря:

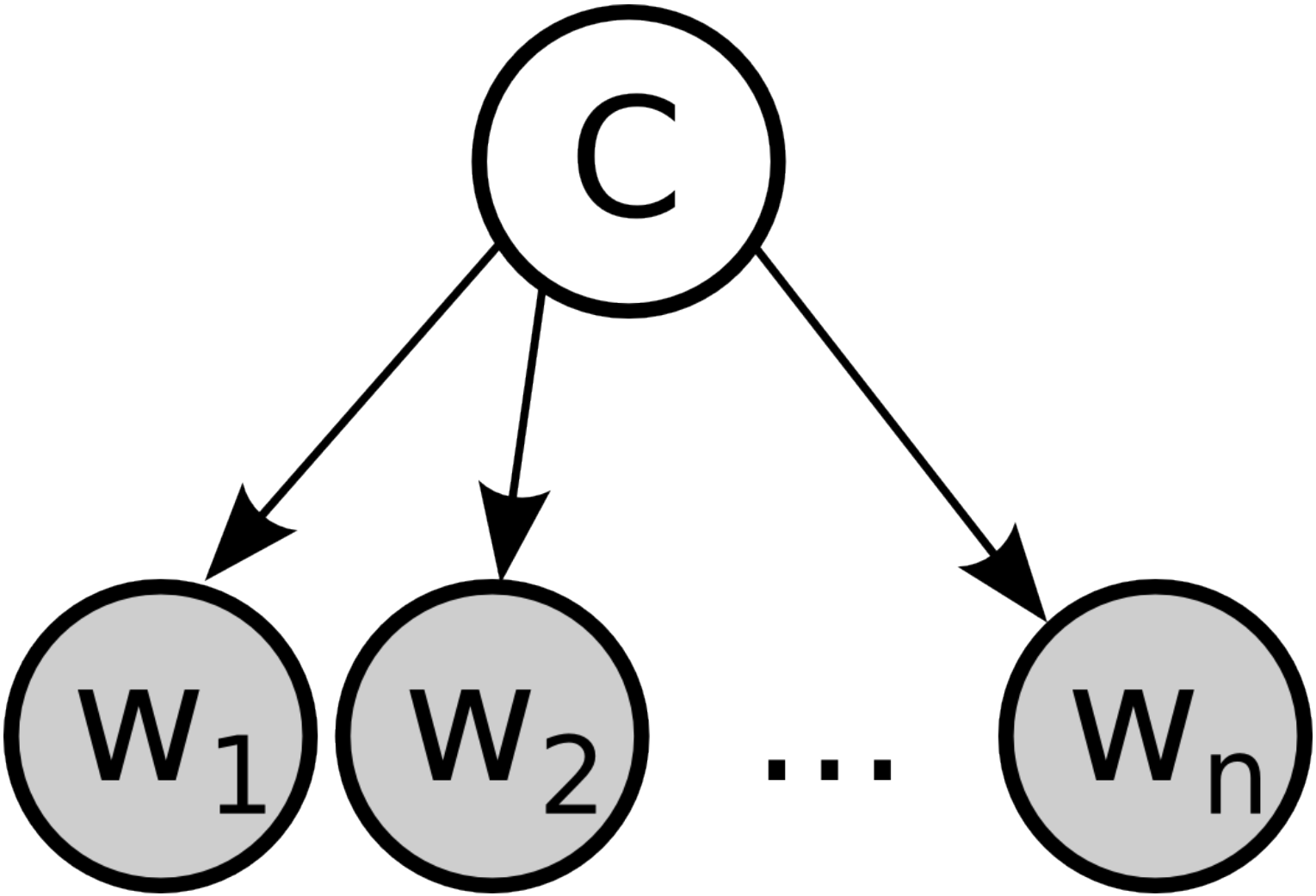
$$p(w_1, w_2, \ldots, w_n | x = v) = p(w_1 | x = v) p(w_2 | x = v) \ldots p(w_n | x = v).$$

- Итак, наивный байесовский классификатор выбирает v как

$$v_{NB}(w_1, w_2, \ldots, w_n) = \arg\max_{v \in V} p(x = v) \prod_{i=1}^{n} p(w_i | x = v).$$

- В парадигме классификации текстов мы предполагаем, что разные слова в тексте на одну и ту же тему появляются независимо друг от друга. Однако, несмотря на такие бредовые предположения, naive Bayes на практике работает очень даже неплохо (и этому есть разумные объяснения).

Naive Bayes: графическая модель



Как можно обобщить наивный байес

- В наивном байесе есть два сильно упрощающих дело предположения:
  - мы знаем метки тем всех документов;
  - у каждого документа только одна тема.
- Мы сейчас уберём оба эти ограничения.
- Во-первых, что можно сделать, если мы не знаем метки тем, т.е. если датасет неразмеченный?

Кластеризация

- Тогда это превращается в задачу кластеризации.
- Её можно решать EM-алгоритмом (Expectation-Maximization, используется в ситуациях, когда есть много скрытых переменных, причём если бы мы их знали, модель стала бы простой):
  - на E-шаге считаем ожидания того, какой документ какой теме принадлежит;
  - на M-шаге пересчитываем наивным байесом вероятности $p(w \mid t)$ при фиксированных метках.
- Это простое обобщение.

Как ещё можно обобщить наивный байес

- А ещё в наивном байесе у документа только одна тема.
- Но это же не так! На самом деле документ говорит о многих темах (но не слишком многих).
- Давайте попробуем это учесть.

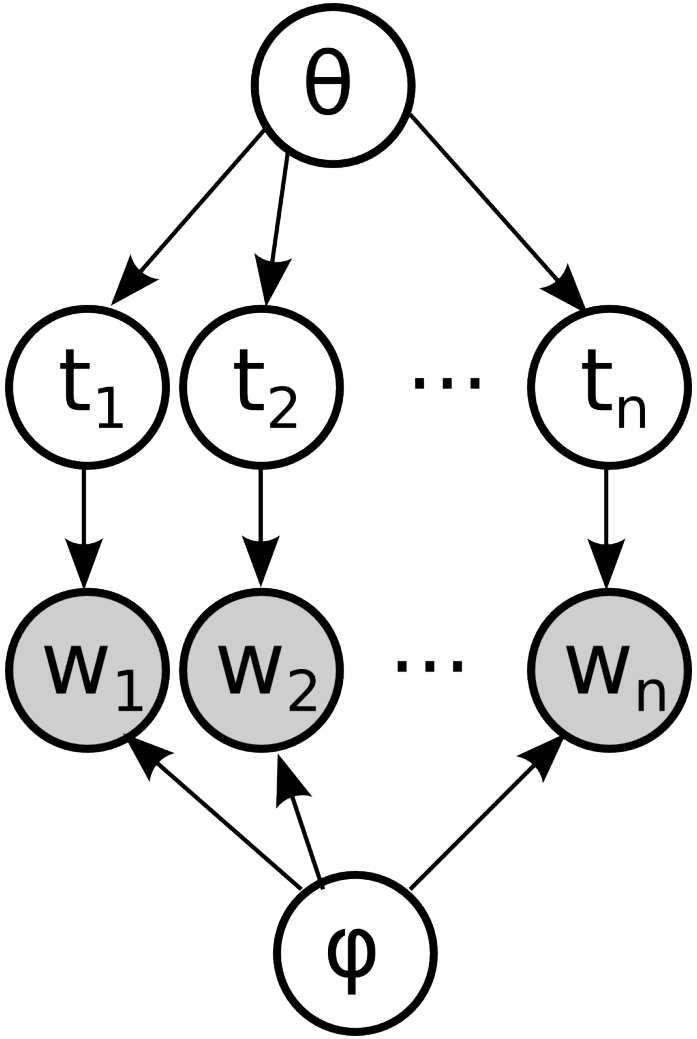
pLSA

- Рассмотрим такую модель:
  - каждое слово в документе d порождается некоторой темой $t \in T$;
  - документ порождается некоторым распределением на темах $p(t \mid d)$;
  - слово порождается именно темой, а не документом: $p(w \mid d, t) = p(w \mid d)$;
  - итого получается такая функция правдоподобия:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t) p(t \mid d).$$

- Эта модель называется probabilistic latent semantic analysis, pLSA (Hoffmann, 1999).

pLSA: графическая модель документа



pLSA

- Как её обучать? Мы можем оценить $p(w \mid d) = \frac{n_{wd}}{n_d}$, а нужно найти:
  - $\phi_{wt} = p(w \mid t)$;
  - $\theta_{td} = p(t \mid d)$.
- Максимизируем правдоподобие

$$p(D) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}} = \prod_{d \in D} \prod_{w \in d} \left[\sum_{t \in T} p(w \mid t) p(t \mid d) \right]^{n_{dw}}.$$

- Как максимизировать такие правдоподобия?

pLSA

- EM-алгоритмом. На E-шаге ищем, сколько слов w в документе d из темы t:

$$n_{dwt} = n_{dw} p(t \mid d, w) = n_{dw} \frac{\phi_{wt}\theta_{td}}{\sum_{s \in T} \phi_{ws}\theta_{sd}}.$$

- А на M-шаге пересчитываем параметры модели:

$$n_{wt} = \sum_d n_{dwt}, \qquad n_t = \sum_w n_{wt}, \quad \phi_{wt} = \frac{n_{wt}}{n_t},$$
$$n_{td} = \sum_{w \in d} n_{dwt}, \quad \theta_{td} = \frac{n_{td}}{n_d}.$$

- Вот и весь вывод в pLSA.

pLSA

- Можно даже не хранить всю матрицу n_{dwt}, а двигаться по документам, каждый раз добавляя n_{dwt} сразу к счётчикам n_{wt}, n_{td}.
- Чего тут не хватает?
  - Во-первых, разложение такое, конечно, будет сильно не единственным.
  - Во-вторых, параметров очень много, явно будет оверфиттинг, если корпус не на порядки больше числа тем.
  - А совсем хорошо было бы получать не просто устойчивое решение, а обладающее какими-нибудь заданными хорошими свойствами.
- Всё это мы можем решить как?

pLSA

- Правильно, регуляризацией. Есть целая наука о разных регуляризаторах для pLSA (K.B. Воронцов).
- В общем виде так: добавим регуляризаторы R_i в логарифм правдоподобия:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + \sum_i \tau_i R_i(\Phi, \Theta).$$

pLSA

- Тогда в EM-алгоритме на M-шаге появятся частные производные R:

$$n_{wt} = \left[\sum_{d \in D} n_{dwt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right]_+,$$

$$n_{td} = \left[\sum_{w \in d} n_{dwt} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right]_+$$

- Чтобы доказать, EM надо рассмотреть как решение задачи оптимизации через условия Каруша-Куна-Такера.

pLSA

- И теперь мы можем кучу разных регуляризаторов вставить в эту модель:
  - регуляризатор сглаживания (позже, это примерно как LDA);
  - регуляризатор разреживания: максимизируем KL-расстояние между распределениями ϕ_{wt} и θ_{td} и равномерным распределением;
  - регуляризатор контрастирования: минимизируем ковариации между векторами ϕ_t, чтобы в каждой теме выделилось своё лексическое ядро (характерные слова);
  - регуляризатор когерентности: будем награждать за слова, которые в документах стоят ближе друг к другу;
  - и так далее, много всего можно придумать.

Outline

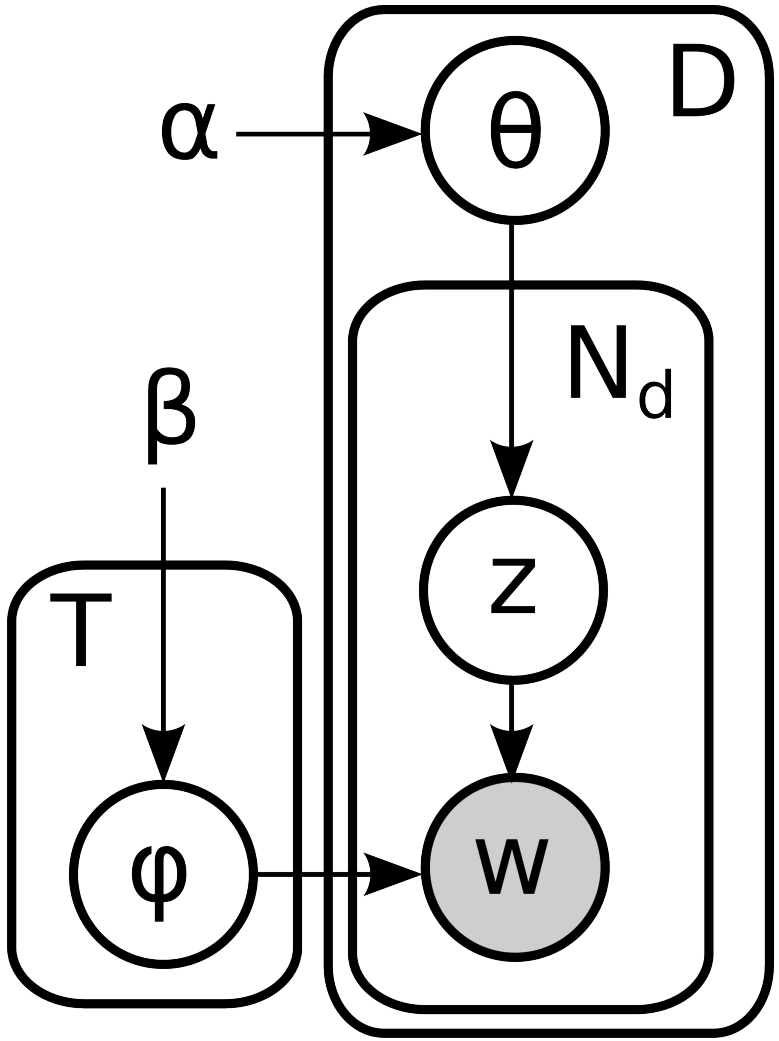
LDA

- Развитие идей pLSA – LDA (Latent Dirichlet Allocation), фактически байесовский вариант pLSA.
- Задача та же: смоделировать большую коллекцию текстов (например, для information retrieval или классификации).
- 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.

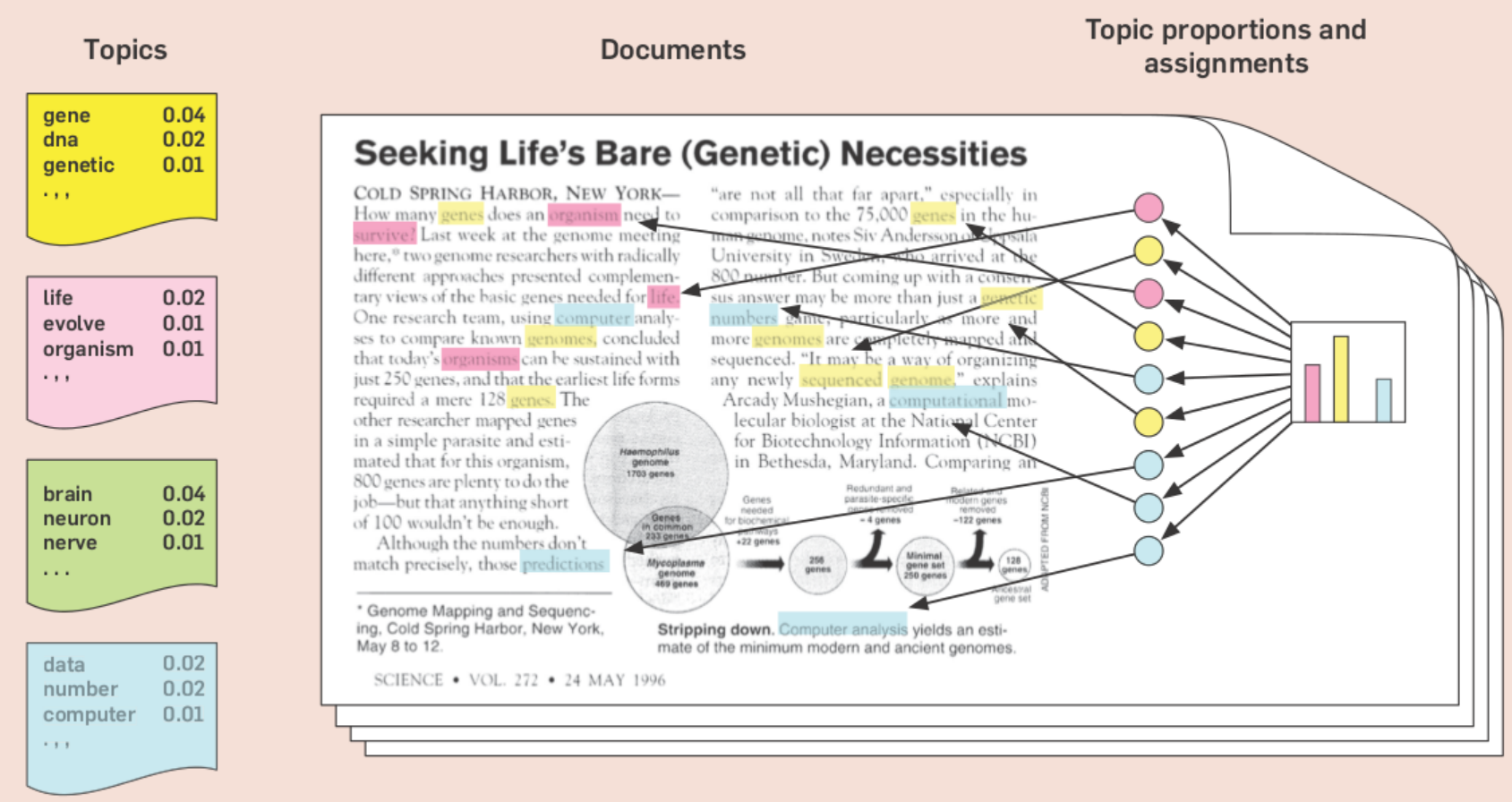
LDA

- Если формально: слова берутся из словаря $\{1, \ldots, V\}$; слово – это вектор w, $w_i \in \{0, 1\}$, где ровно одна компонента равна 1.
- Документ – последовательность из N слов $\boldsymbol{w}$. Нам дан корпус из M документов $\mathcal{D} = \{\boldsymbol{w}_d \mid d = 1..M\}$.
- 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.

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.

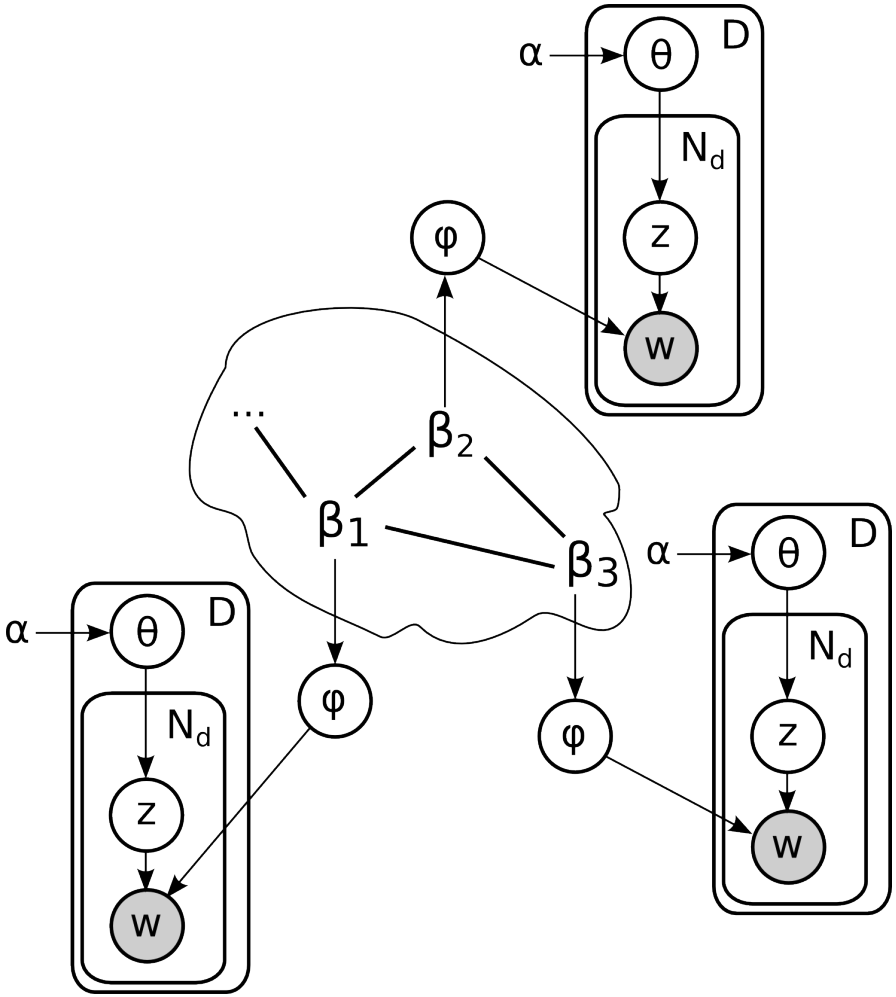
Коррелированные тематические модели

- 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) – меняем априорное распределение на более выразительное, которое может моделировать корреляции.
- 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).
- Несколько 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).

Марковские тематические модели



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.

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:

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

- 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.

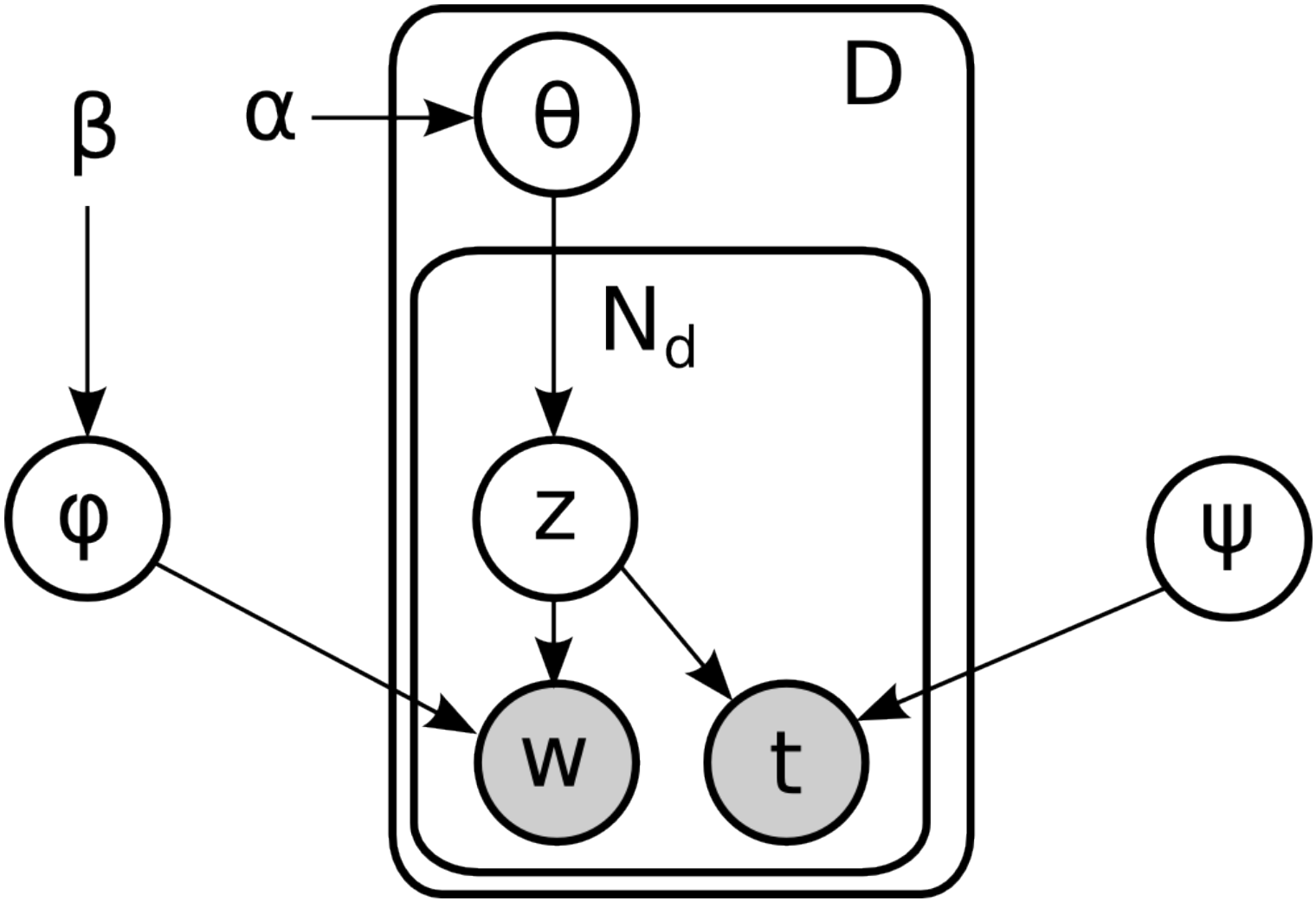
Topics over Time

- 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

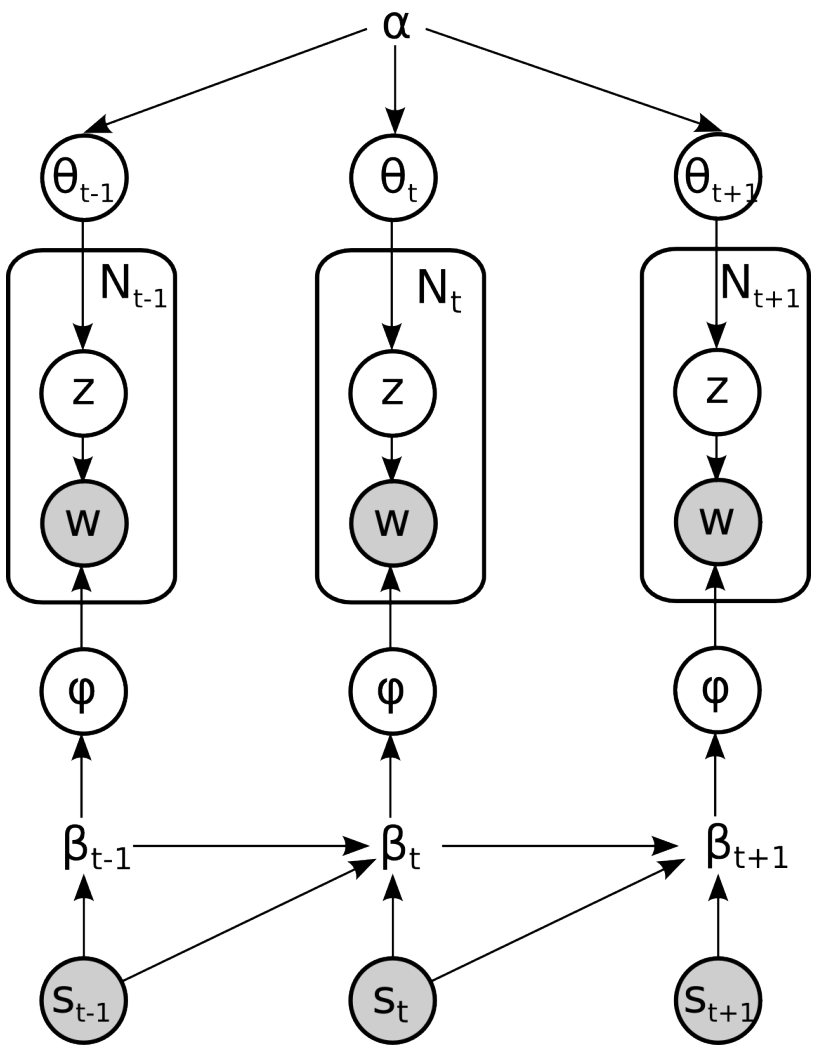
- *Динамические тематические модели* представляют временную эволюцию тем через эволюцию их гиперпараметров α и/или β.
- Бывают дискретные ([d]DTM), в которых время дискретно, и непрерывные, где эволюция гиперпараметра β (α здесь предполагается постоянным) моделируется посредством броуновского движения: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

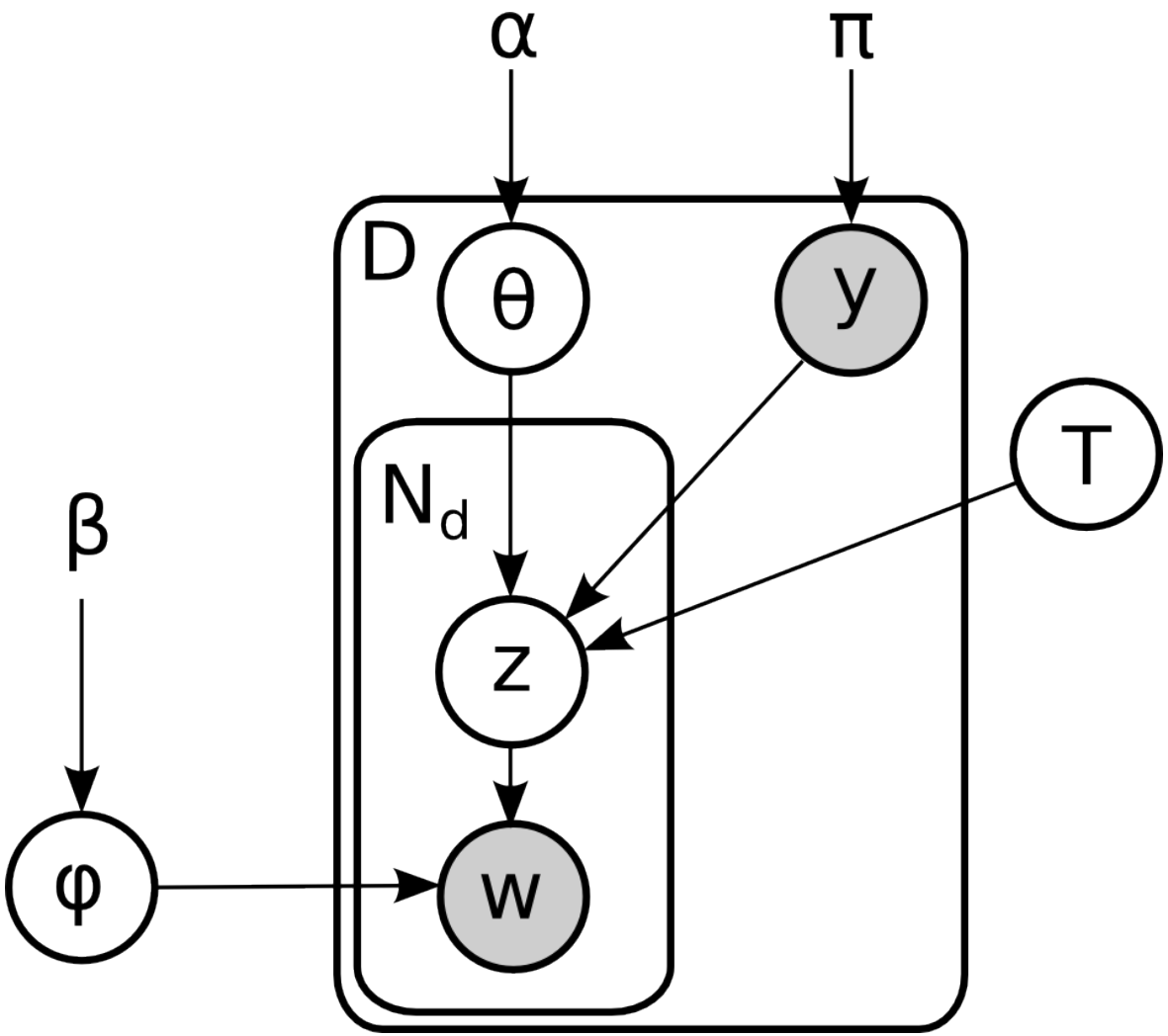
Непрерывная динамическая тематическая модель (cDTM)



DiscLDA

- Дискриминативное LDA (DiscLDA), 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.

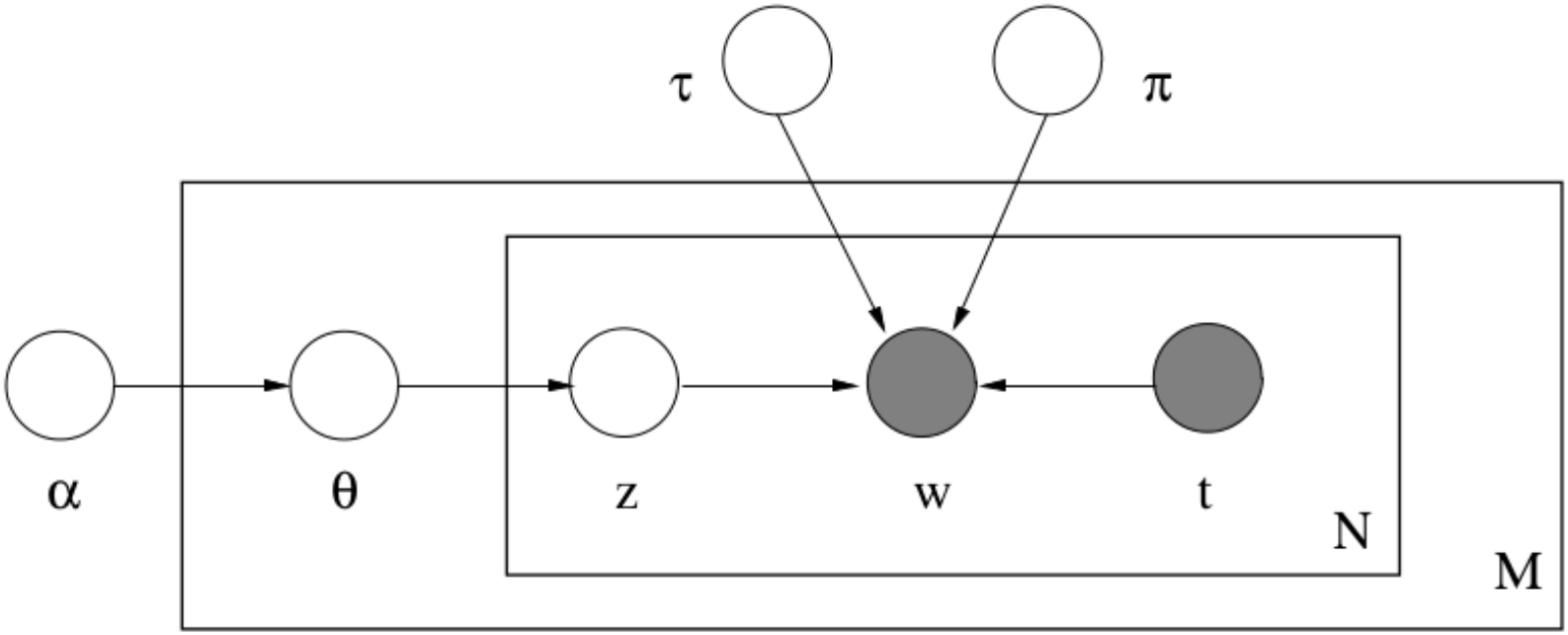
DiscLDA



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.

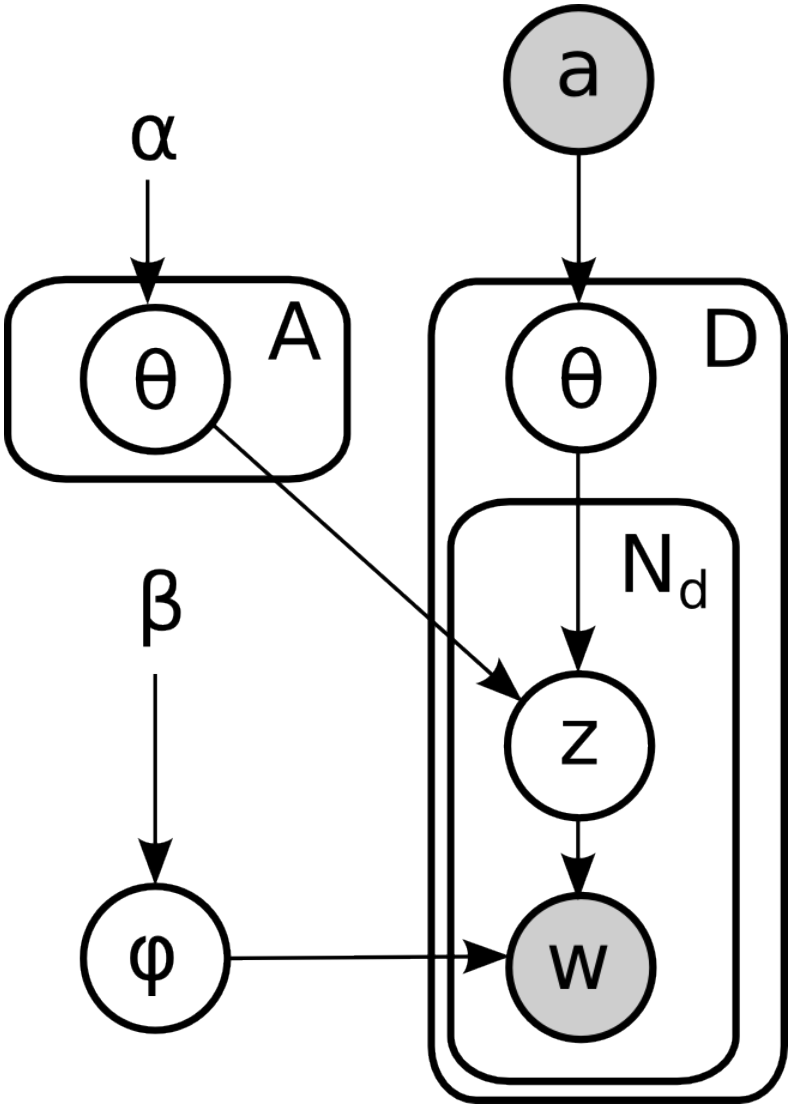
TagLDA



Author-Topic model

- Author-Topic modeling: кроме собственно текстов, присутствуют их авторы.
- У автора тоже вводится распределение на темах, на которые он пишет, и тексты автора будут скорее о более вероятных для него темах.
- Т.е. это всё же скорее об авторах научных статей, чем об атрибуции текстов (там другие методы).

Author-Topic model



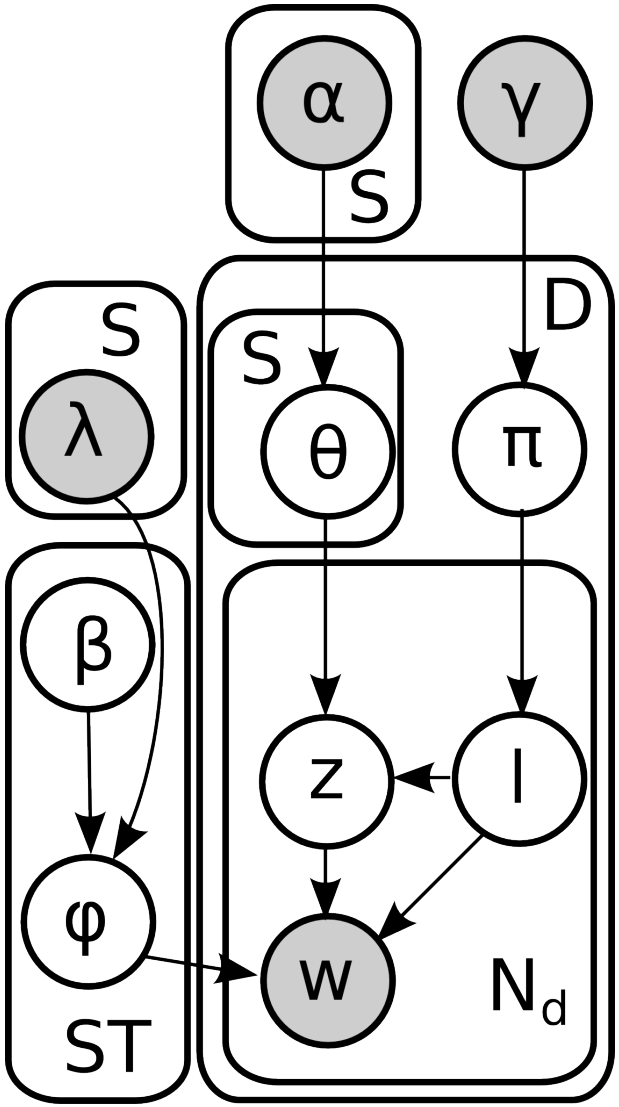
Базовая генеративная модель Author-Topic model (остальное как в базовом LDA):

- для каждого слова w:
  - выбираем автора x для этого слова из множества авторов документа $\boldsymbol{a}_d$;
  - выбираем тему из распределения на темах, соответствующего автору x;
  - выбираем слово из распределения слов, соответствующего этой теме.

LDA для sentiment analysis

- LDA для sentiment analysis: можно ли выделить из, например, обзоров продуктов разные темы (*аспекты*): например, для отеля – место, качество обслуживания, еда, комнаты и т.п.
- К этому ещё добавляется, что разные слова в разных аспектах могут нести разную нагрузку.
- Здесь есть несколько расширений LDA.

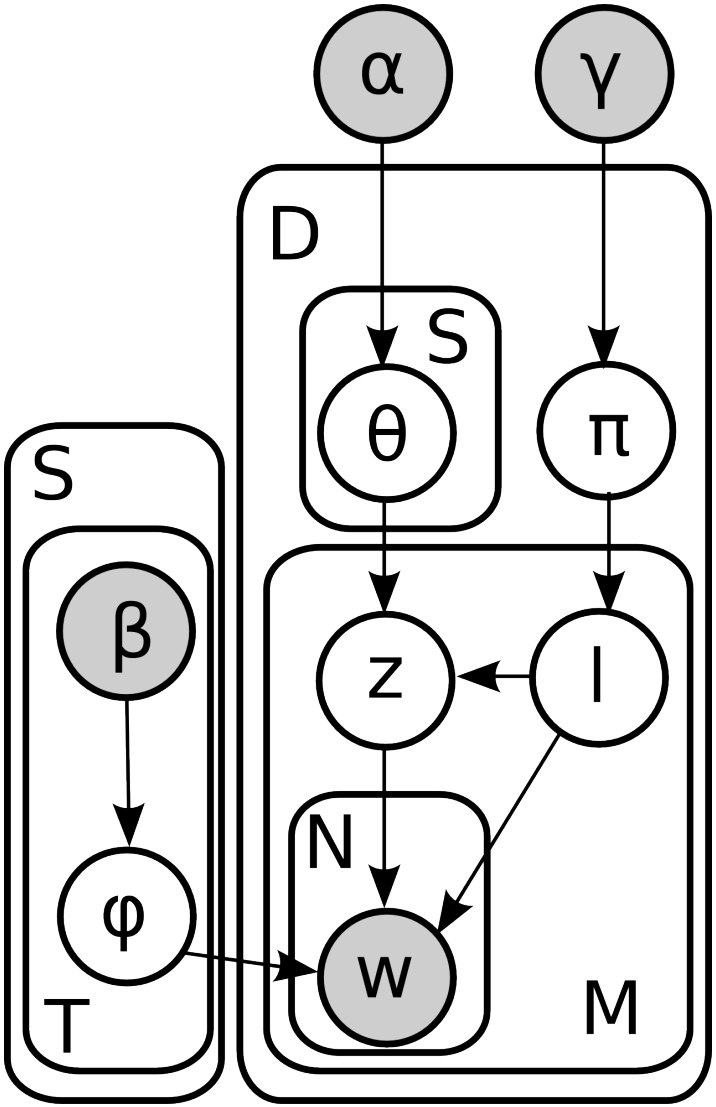
JST



Joint Sentiment-Topic model (JST):

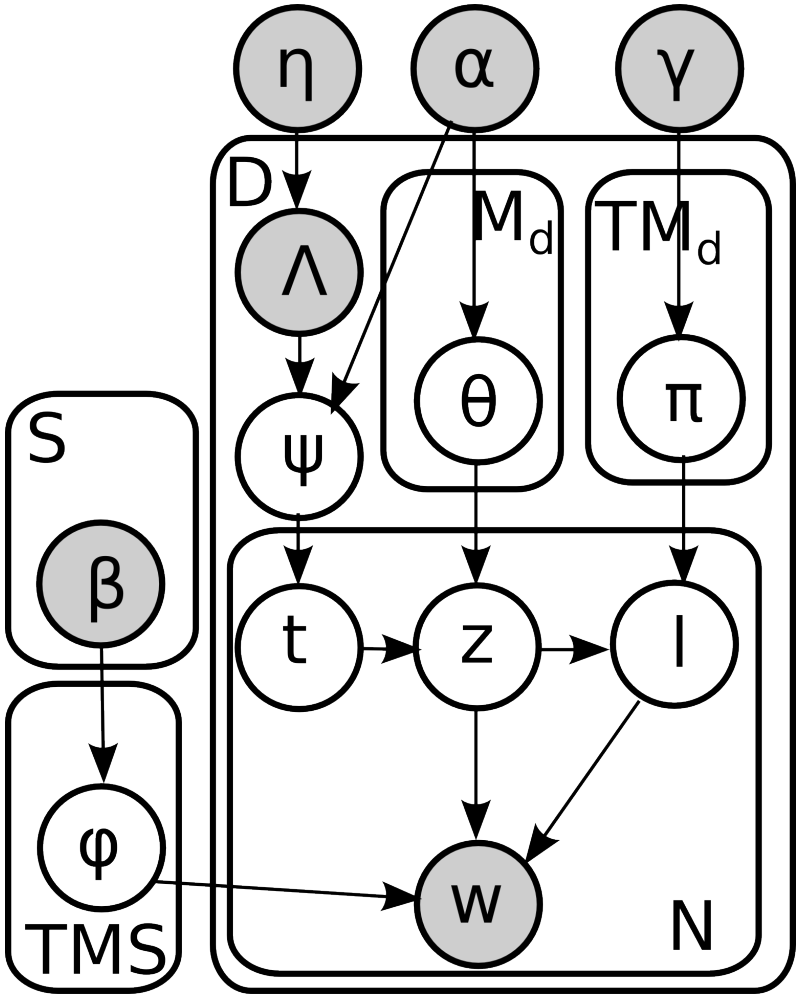
- тема зависит от сентимента;
- сентимент берётся из распределений сентимента π_d;
- у слов тоже есть метки сентимента λ;
- а слова в документах зависят от пар «тема–сентимент».

ASUM



Aspect and Sentiment Unification Model (ASUM): примерно то же, что JST, но делим обзор на предложения и предполагаем, что одно предложение всегда говорит об одном аспекте (SentenceLDA).

USTM



User-aware Sentiment Topic Model (USTM): а теперь добавим данные о конкретном пользователе, выраженные как теги документов a_j, взятые из соответствующих распределений ψ_d.

Наши последние результаты

- Все эти модели для определения окрашенных слов модифицируют их априорные распределения β, т.е. фактически подставляют готовый словарь окрашенных слов.
- Но на практике не всегда словарь по данному аспекту есть, а когда есть, его всё равно неплохо бы расширить.
- Поэтому идея такая:
  - инициализируем β по имеющемуся словарю;
  - EM-схема – повторяем до сходимости:
    - немножко обучаем тематическую модель с фиксированными β;
    - переобучаем β по обучившимся темам.

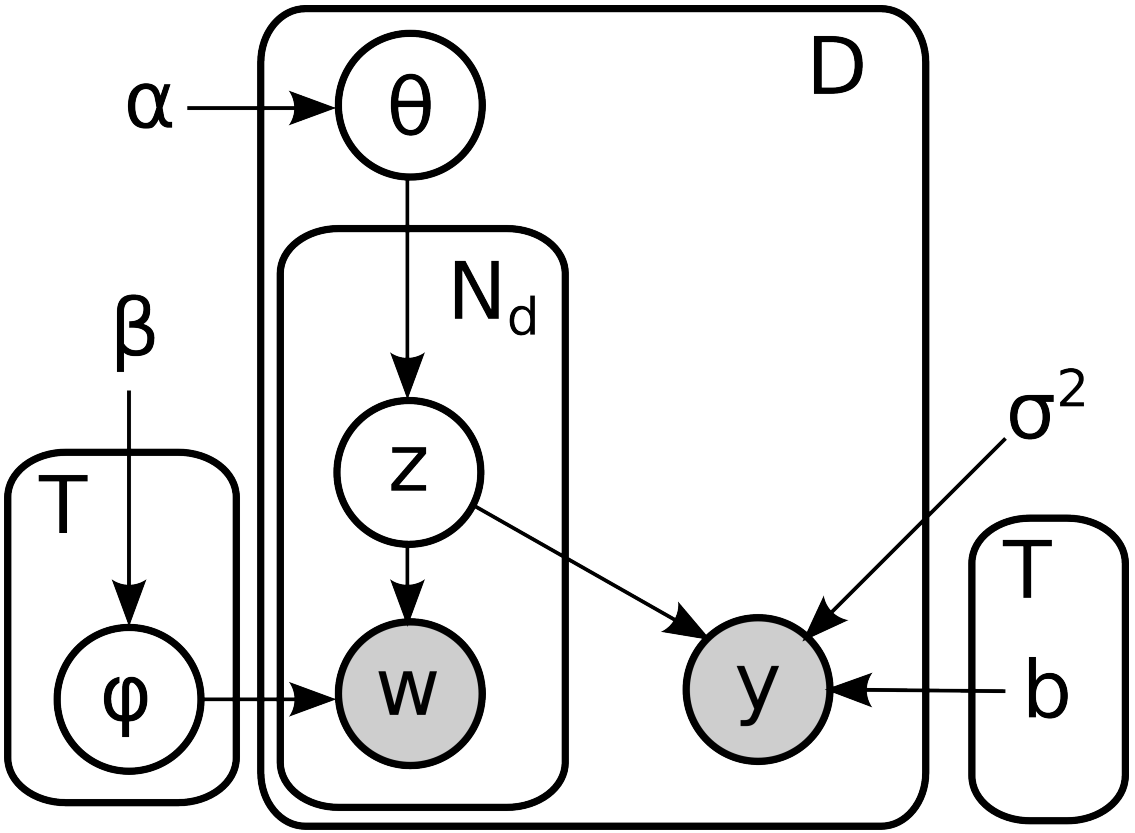
SVD-LDA

- LDA для рекомендательных систем: как нам рекомендовать тексты?
- Можно просто обучить темы и построить «профили интересов».
- Но можно и лучше: можно обучить темы так, чтобы темы лучше соответствовали тому, что мы хотим рекомендовать.
- Это делается при помощи варианта модели Supervised LDA.

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.

SVD-LDA



- В качестве связей можно представить себе SVD. Но фактор получается слишком сложный, для каждого сэмпла нужно бегать по всему рекомендательному датасету. Мы разработали приближённую схему сэмплирования.

TwitterLDA

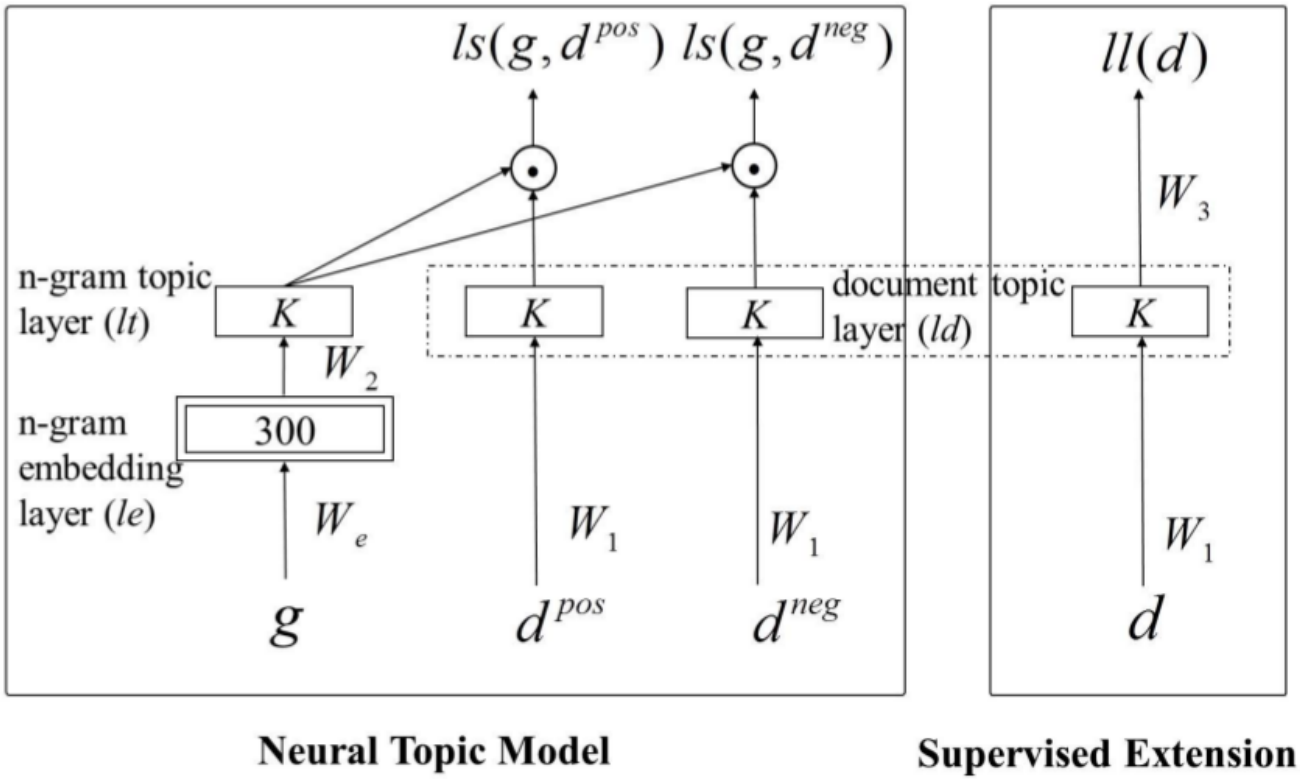
- TwitterLDA – тематическая модель для коротких текстов.
- Вряд ли один твит может содержать сразу много разных тем.
- Но при этом у твитов есть авторы, и у авторов-то уже есть любимые темы.
- Поэтому TwitterLDA устроен так:
  - распределение θ – не у текста, а у автора;
  - из него сэмплируется тема сразу на весь твит;
  - и потом из этой темы сэмплируются все слова.
- Похоже на SentenceLDA.

Neural topic models

- Совсем недавние работы – как скрестить topic modeling и deep learning?
- Одна идея – изменить понятие темы:
  - раньше было мультиномиальное распределение, и все слова были независимы;
  - теперь пусть будет распределение в семантическом пространстве word2vec;
  - в [Sridhar, 2015] тема – это смесь гауссианов в семантическом пространстве.

Neural topic models

- Другая идея – neural topic models [Cao et al., 2015]:
  - рассмотрим базовое уравнение тематических моделей $p(w \mid d) = \sum_t p(w \mid t)p(t \mid d) = \phi(w)^\top \theta(d)$;
  - и представим $\phi(w)$ и $\theta(d)$ нейронными сетями:



Выводы

- Тематические модели берут на вход корпус текстов и без учителя обучают распределения слов в темах и тем в документах.
- Фактически получается сжатое представление матрицы слова×документы как произведение матриц слова×темы и темы×документы.
- Можно просто обучить стандартный LDA и потом темы использовать для ваших целей как features, как сжатое представление текстов.
- А можно использовать одно из многочисленных расширений LDA, которое, теоретически :), должно работать лучше.

Thank you!

Thank you for your attention!

Многомерная модель

- Но есть одна тонкость в деталях реализации наивного байесовского классификатора.
- Сейчас мы рассмотрим два разных подхода к naive Bayes, которые дают разные результаты: мультиномиальный (multinomial) и многомерный (multivariate).

Многомерная модель

- В многомерной модели документ – это вектор бинарных атрибутов, показывающих, встретилось ли в документе то или иное слово.
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что встретилось каждое слово из документа и вероятности того, что не встретилось каждое (словарное) слово, которое не встретилось.
- Получается модель многомерных испытаний Бернулли. Наивное предположение в том, что события «встретилось ли слово» предполагаются независимыми.

Многомерная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|V|$, состоящий из битов B_{it}; $B_{it} = 1$ iff слово w_t встречается в документе d_i.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = \prod_{t=1}^{|V|} \left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right).$$

- Для обучения такого классификатора нужно обучить вероятности $p(w_t \mid c_j)$.

Многомерная модель

- Обучение – дело нехитрое: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем биты B_{it} (знаем документы).
- Тогда можно подсчитать оптимальные оценки вероятностей того, что то или иное слово встречается в том или ином классе (при помощи лапласовой оценки):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} B_{it} p(c_j \mid d_i)}{2 + \sum_{i=1}^{|D|} p(c_j \mid d_i)}.$$

Многомерная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$c = \arg\max_j p(c_j) p(d_i \mid c_j) =$$

$$= \arg\max_j \left(\frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i) \right) \prod_{t=1}^{|V|} (B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))) =$$

$$= \arg\max_j \left(\log(\sum_{i=1}^{|D|} p(c_j \mid d_i)) + \sum_{t=1}^{|V|} \log (B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j) \right.$$

Мультиномиальная модель

- В мультиномиальной модели документ – это последовательность событий. Каждое событие – это случайный выбор одного слова из того самого «bag of words».
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что мы достали из мешка те самые слова, которые встретились в документе. Наивное предположение в том, что мы достаём из мешка разные слова независимо друг от друга.
- Получается мультиномиальная генеративная модель, которая учитывает количество повторений каждого слова, но не учитывает, каких слов *нет* в документе.

Мультиномиальная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|d_i|$, состоящий из слов, каждое из которых «вынуто» из словаря с вероятностью $p(w_t \mid c_j)$.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}},$$

  где N_{it} – количество вхождений w_t в d_i.
- Для обучения такого классификатора тоже нужно обучить вероятности $p(w_t \mid c_j)$.

Мультиномиальная модель

- Обучение: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем вхождения N_{it}.
- Тогда можно подсчитать апостериорные оценки вероятностей того, что то или иное слово встречается в том или ином классе (не забываем сглаживание – правило Лапласа):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} p(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} p(c_j \mid d_i)}.$$

Мультиномиальная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$c = \arg\max_j p(c_j) p(d_i \mid c_j) =$$

$$= \arg\max_j \left(\frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i) \right) p(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}} =$$

$$= \arg\max_j \left(\log \left(\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) + \sum_{t=1}^{|V|} N_{it} \log p(w_t \mid c_j) \right).$$

LDA

- Мы пока для простоты фиксируем число тем k, считаем, что β – это просто набор параметров $\beta_{ij} = p(w^j = 1 \mid z^i = 1)$, которые нужно оценить, и не беспокоимся о распределении на N.
- Совместное распределение тогда выглядит так:

$$p(\theta, z, w, N \mid \alpha, \beta) = p(N \mid \xi) p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta).$$

- В отличие от обычной кластеризации с априорным распределением Дирихле, мы тут не выбираем кластер один раз, а затем накидываем слова из этого кластера, а для каждого слова выбираем по распределению θ, по какой теме оно будет набросано.

Вывод в LDA

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

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

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

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

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

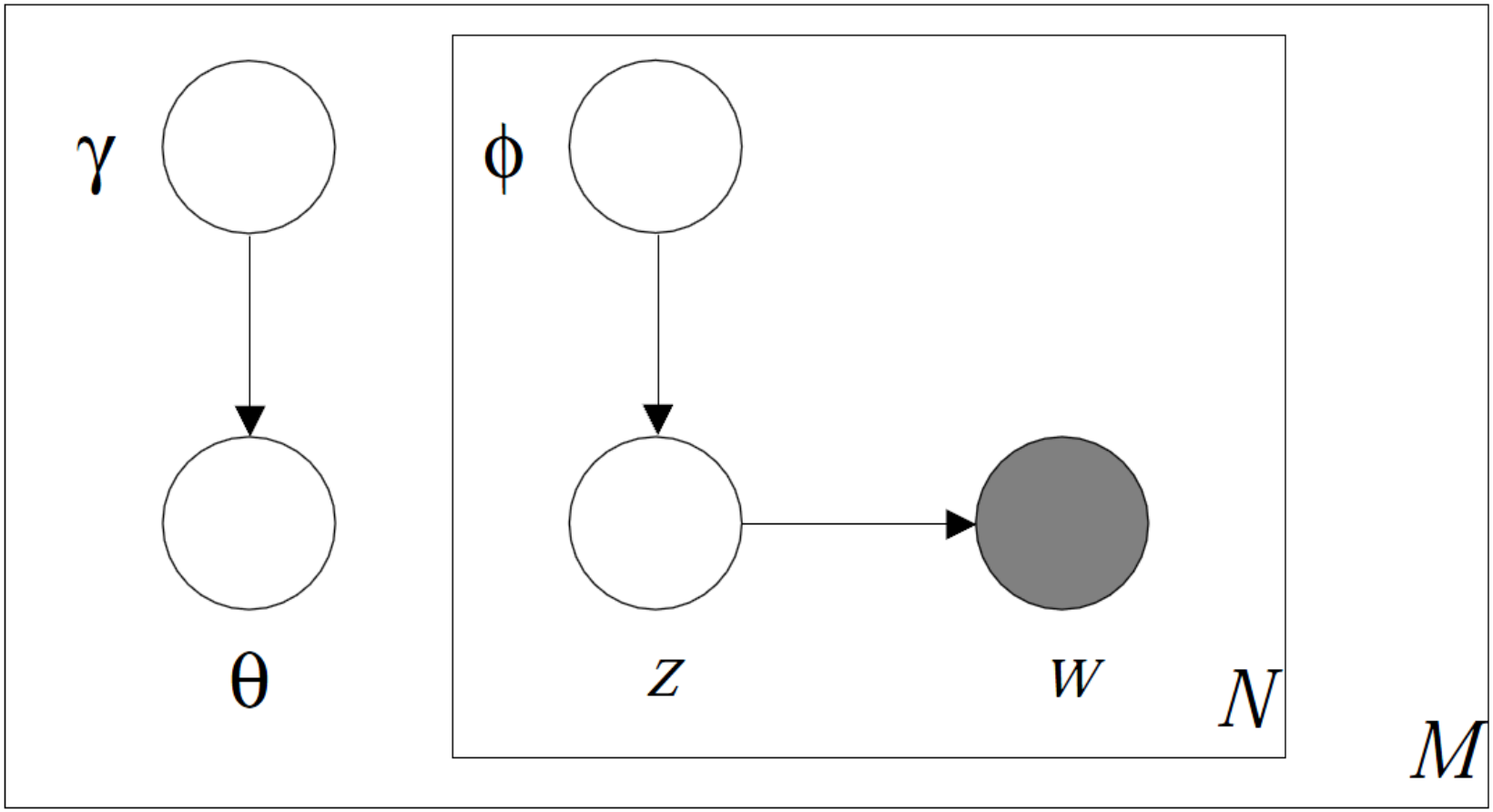
Вывод в LDA

- Вариационное приближение – рассмотрим семейство распределений

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

- 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 $\boldsymbol{w}$.

LDA: вариационное приближение



LDA: вариационный вывод

- Теперь можно искать минимум KL-расстояния:

$$(\gamma^*, \phi^*) = \arg \min_{(\gamma,\phi)} \mathrm{KL}(q(\theta, z \mid \boldsymbol{w}, \gamma\phi) \| p(\theta, \boldsymbol{z} \mid \boldsymbol{w}, \alpha, \beta)).$$

- Для этого сначала воспользуемся уже известной оценкой из неравенства Йенсена:

$$\log p(\boldsymbol{w} \mid \alpha, \beta) = \log \int_\theta \sum_{\boldsymbol{z}} p(\theta, \boldsymbol{z}, \boldsymbol{w} \mid \alpha, \beta) d\theta =$$

$$= \log \int_\theta \sum_{\boldsymbol{z}} \frac{p(\theta, \boldsymbol{z}, \boldsymbol{w} \mid \alpha, \beta) q(\theta, \boldsymbol{z})}{q(\theta, \boldsymbol{z})} d\theta \geq$$

$$\geq E_q [\log p(\theta, \boldsymbol{z}, \boldsymbol{w} \mid \alpha, \beta)] - E_q [\log q(\theta, \boldsymbol{z})] =: \mathcal{L}(\gamma, \phi; \alpha, \beta).$$

LDA: вариационный вывод

- Распишем произведения:

$$\mathcal{L}(\gamma, \phi; \alpha, \beta) = E_q\left[p(\theta \mid \alpha)\right] + E_q\left[p(\boldsymbol{z} \mid \theta)\right] + E_q\left[p(\boldsymbol{w} \mid \boldsymbol{z}, \beta)\right] - \\ - E_q\left[\log q(\theta)\right] - E_q\left[\log q(\boldsymbol{z})\right].$$

- Свойство распределения Дирихле: если $X \sim \mathrm{Di}(\alpha)$, то

$$E[\log(X_i)] = \Psi(\alpha_i) - \Psi(\sum_i \alpha_i),$$

где $\Psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ – дигамма-функция.

- Теперь можно выписать каждый из пяти членов.

LDA: вариационный вывод

$$\mathcal{L}(\gamma, \phi; \alpha, \beta) = \log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1) \left[\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right) \right] +$$
$$+ \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \left[\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right) \right] +$$
$$+ \sum_{n=1}^{N} \sum_{i=1}^{k} \sum_{j=1}^{V} w_n^j \phi_{ni} \log \beta_{ij} -$$
$$- \log \Gamma(\sum_{i=1}^{k} \gamma_i) + \sum_{i=1}^{k} \log \Gamma(\gamma_i) - \sum_{i=1}^{k} (\gamma_i - 1) \left[\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right) \right] -$$
$$- \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \log \phi_{ni}.$$

LDA: вариационный вывод

- Теперь осталось только брать частные производные этого выражения.
- Сначала максимизируем его по ϕ_{ni} (вероятность того, что n-е слово было порождено темой i); надо добавить λ-множители Лагранжа, т.к. $\sum_{j=1}^{k} \phi_{nj} = 1$.
- В итоге получится:

$$\phi_{ni} \propto \beta_{iv} e^{\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)},$$

где v – номер того самого слова, т.е. единственная компонента $w_n^v = 1$.

LDA: вариационный вывод

- Потом максимизируем по γ_i, i-й компоненте апостериорного Дирихле-параметра.
- Получится

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}.$$

- Соответственно, для вывода нужно просто пересчитывать ϕ_{ni} и γ_i друг через друга, пока оценка не сойдётся.
- Фактически мы получили алгоритм вывода pLSA с неким видом регуляризатора (как раз сглаживающим байесовским).

LDA: оценка параметров

- Теперь давайте попробуем оценить параметры α и β по корпусу документов $\mathcal{D}$.
- Мы хотим найти α и β, которые максимизируют

$$\ell(\alpha, \beta) = \sum_{d=1}^{M} \log p(\boldsymbol{w}_d \mid \alpha, \beta).$$

- Подсчитать $p(\boldsymbol{w}_d \mid \alpha, \beta)$ мы не можем, но у нас есть нижняя оценка $\mathcal{L}(\gamma, \phi; \alpha, \beta)$, т.к.

$$p(\boldsymbol{w}_d \mid \alpha, \beta) =$$
$$= \mathcal{L}(\gamma, \phi; \alpha, \beta) + \mathrm{KL}(q(\theta, z \mid \boldsymbol{w}_d, \gamma\phi) \| p(\theta, \boldsymbol{z} \mid \boldsymbol{w}_d, \alpha, \beta)).$$

LDA: оценка параметров

- EM-алгоритм:
  1. найти параметры $\{\gamma_d, \phi_d \mid d \in \mathcal{D}\}$, которые оптимизируют оценку (как выше);
  2. зафиксировать их и оптимизировать оценку по α и β.

LDA: оценка параметров

- Для β это тоже делается нехитро:

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni} w_n^j.$$

- Для α_i получается система уравнений, которую можно решить методом Ньютона.

LDA: сэмплирование по Гиббсу

- Теперь давайте попробуем другим методом, который мы знаем: сэмплированием по Гиббсу.
- В случае LDA сэмплирование получается совсем простое и относительно быстрое, но чтобы его получить, надо немножко постараться.
- Дело в том, что некоторые переменные можно будет сразу проинтегрировать.

LDA: сэмплирование по Гиббсу

- Правдоподобие датасета:

$$p(\boldsymbol{w}, \boldsymbol{z}, \theta, \boldsymbol{\Phi} \mid \alpha, \beta) =$$
$$= p(\boldsymbol{\Phi} \mid \beta) p(\theta \mid \alpha) p(\boldsymbol{z} \mid \theta) p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{\Phi}) =$$
$$= \prod_t p(\phi_t \mid \beta) \prod_d p(\theta_d \mid \alpha) \prod_d \prod_{w \in d} p(z_w \mid \theta_d) \prod_d \prod_{w \in d} p(w \mid \phi_{z_w}).$$

- Обозначим через n_{tdw} то, сколько раз слово w в документе d принадлежит теме t; соответственно n_{t*w}, n_{td*} и так далее.

LDA: сэмплирование по Гиббсу

- Для сэмплирования по Гиббсу надо посчитать

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

$$= \int\int p(\boldsymbol{w}, \boldsymbol{z}, \theta, \boldsymbol{\phi} \mid \alpha, \beta) d\theta d\boldsymbol{\phi} =$$

$$= \int\int p(\boldsymbol{\phi} \mid \beta) p(\theta \mid \alpha) p(\boldsymbol{z} \mid \theta) p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{\phi}) d\theta d\boldsymbol{\phi} =$$

$$= \int p(\boldsymbol{z} \mid \theta) p(\theta \mid \alpha) d\theta \times \int p(\boldsymbol{\phi} \mid \beta) p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{\phi}) d\boldsymbol{\phi} =$$

$$= \int \prod_d p(\theta_d \mid \alpha) \prod_d \prod_{w \in d} p(z_w \mid \theta_d) d\theta \times$$

$$\times \int \prod_t p(\phi_t \mid \beta) \prod_d \prod_{w \in d} p(w \mid \phi_{z_w}) d\boldsymbol{\phi}.$$

LDA: сэмплирование по Гиббсу

- И дальше можно разбить интегралы:
- $\ldots = \int \prod_d p(\theta_d \mid \alpha) \prod_d \prod_{w \in d} p(z_w \mid \theta_d) d\boldsymbol{\theta} \times \int \prod_t p(\phi_t \mid \beta) \prod_d \prod_{w \in d} p(w \mid \phi_{z_w}) d\boldsymbol{\Phi} =$

$$= \prod_d \int p(\theta_d \mid \alpha) \prod_d \prod_{w \in d} p(z_w \mid \theta_d) d\theta_d \times$$

$$\times \prod_t \int p(\phi_t \mid \beta) \prod_d \prod_{w \in d} p(w \mid \phi_{z_w}) d\phi_t =$$

$$= \prod_d \int \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{dt}^{\alpha_k - 1} \prod_{w \in d} \theta_{dz_w} d\theta_d \times$$

$$\times \prod_t \int \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{tw}^{\beta_w - 1} \prod_w \phi_{tw}^{n_{t*w}} d\phi_t = \ldots$$

LDA: сэмплирование по Гиббсу

- Теперь соберём степени при θ и ϕ:
- $$\ldots = \prod_d \int \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{dt}^{\alpha_k - 1} \prod_{w \in d} \theta_{dz_w} d\theta_d \times$$
$$\times \prod_t \int \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{tw}^{\beta_w - 1} \prod_w \phi_{tw}^{n_{t*w}} d\phi_t =$$

$$\prod_d \int \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{dt}^{\alpha_k - 1} \prod_t \theta_{dt}^{n_{td*}} d\theta_d \times$$
$$\times \prod_t \int \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{tw}^{\beta_w - 1} \prod_w \phi_{tw}^{n_{t*w}} d\phi_t =$$
$$= \prod_d \int \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{dt}^{\alpha_t + n_{td*} - 1} d\theta_d \times$$
$$\times \prod_t \int \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{tw}^{\beta_w + n_{t*w} - 1} d\phi_t = \ldots$$

LDA: сэмплирование по Гиббсу

- Теперь ключевой момент – можно взять интегралы:

$$\int \prod_t \theta_{dt}^{\alpha_t + n_{td*} - 1} d\theta_d = \frac{\prod_t \Gamma(n_{td*} + \alpha_t)}{\Gamma(\sum_t n_{td*} + \alpha_t)},$$

$$\int \prod_w \phi_{tw}^{\beta_w + n_{t*w} - 1} d\phi_t = \frac{\prod_w \Gamma(n_{t*w} + \beta_w)}{\Gamma(\sum_w n_{t*w} + \beta_w)}.$$

LDA: сэмплирование по Гиббсу

- В итоге получается

$$\ldots = \prod_d \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \frac{\prod_t \Gamma(n_{td*} + \alpha_t)}{\Gamma(\sum_t n_{td*} + \alpha_t)} \times \times \prod_t \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \frac{\prod_w \Gamma(n_{t*w} + \beta_w)}{\Gamma(\sum_w n_{t*w} + \beta_w)} \propto \ldots$$

- ...выкидываем то, что зависит только от α и β...

$$\ldots \propto \prod_d \frac{\prod_t \Gamma(n_{td*} + \alpha_t)}{\Gamma(\sum_t n_{td*} + \alpha_t)} \prod_t \frac{\prod_w \Gamma(n_{t*w} + \beta_w)}{\Gamma(\sum_w n_{t*w} + \beta_w)} = \ldots$$

LDA: сэмплирование по Гиббсу

- ...вытаскиваем то, что зависит от текущего сэмпла (a, b) (док-т a, слово b)...

$$\dots \prod_{d \neq a} \frac{\prod_t \Gamma(n_{td*} + \alpha_t)}{\Gamma(\sum_t n_{td*} + \alpha_t)} \times \frac{\prod_t \Gamma(n_{ta*} + \alpha_t)}{\Gamma(\sum_t n_{ta*} + \alpha_t)} \times$$
$$\times \prod_t \frac{\prod_{w \neq b} \Gamma(n_{t*w} + \beta_w) \times \Gamma(n_{t*b} + \beta_b)}{\Gamma(\sum_w n_{t*w} + \beta_w)} \propto \dots$$

- и удаляем всё, что не зависит от (a, b):

$$\dots \propto \frac{\prod_t \Gamma(n_{ta*} + \alpha_t)}{\Gamma(\sum_t n_{ta*} + \alpha_t)} \prod_t \frac{\Gamma(n_{t*b} + \beta_b)}{\Gamma(\sum_w n_{t*w} + \beta_w)}.$$

LDA: сэмплирование по Гиббсу

- $\ldots = \frac{\prod_t \Gamma(n_{ta*}+\alpha_t)}{\Gamma(\sum_t n_{ta*}+\alpha_t)} \prod_t \frac{\Gamma(n_{t*b}+\beta_b)}{\Gamma(\sum_w n_{t*w}+\beta_w)} = \ldots$
- Теперь выделим из сумм текущий сэмпл; пусть $n_{tdw}^{-(a,b)}$ – это те же счётчики, но за вычетом позиции (a, b). Тогда

$$\ldots = \frac{\prod_{t \neq z_b} \Gamma(n_{ta*}^{-(a,b)} + \alpha_t)\Gamma(n_{z_b a*}^{-(a,b)} + \alpha_{z_b} + 1)}{\Gamma(1 + \sum_t (n_{ta*}^{-(a,b)} + \alpha_t))} \times$$

$$\times \prod_{t \neq z_b} \frac{\Gamma(n_{t*b}^{-(a,b)} + \beta_b)}{\Gamma(\sum_w n_{t*w} + \beta_w)} \frac{\Gamma(n_{z_b*b}^{-(a,b)} + \beta_b + 1)}{\Gamma(1 + \sum_w (n_{z_b*w}^{-(a,b)} + \beta_w))} = \ldots$$

LDA: сэмплирование по Гиббсу

- ...и теперь вытащим единички из-под Γ ($\Gamma(x+1) = x\Gamma(x)$), а остальное соберём обратно:

$$\ldots = \frac{\prod_{t \neq z_b} \Gamma(n_{ta*}^{-(a,b)} + \alpha_t)\Gamma(n_{z_b a*}^{-(a,b)} + \alpha_{z_b})(n_{z_b a*}^{-(a,b)} + \alpha_{z_b})}{\Gamma(1 + \sum_t (n_{ta*}^{-(a,b)} + \alpha_t))} \times$$

$$\times \prod_{t \neq z_b} \frac{\Gamma(n_{t*b}^{-(a,b)} + \beta_b)}{\Gamma(\sum_w n_{t*w} + \beta_w)} \frac{\Gamma(n_{z_b*b}^{-(a,b)} + \beta_b)(n_{z_b*b}^{-(a,b)} + \beta_b)}{\Gamma(\sum_w (n_{z_b*w}^{-(a,b)} + \beta_w)) \sum_w (n_{z_b*w}^{-(a,b)} + \beta_w)} =$$

$$= \frac{(n_{z_b a*}^{-(a,b)} + \alpha_{z_b}) \prod_t \Gamma(n_{ta*}^{-(a,b)} + \alpha_t)}{\Gamma(1 + \sum_t (n_{ta*}^{-(a,b)} + \alpha_t))} \times$$

$$\times \prod_t \frac{\Gamma(n_{t*b}^{-(a,b)} + \beta_b)}{\Gamma(\sum_w n_{t*w} + \beta_w)} \frac{n_{z_b*b}^{-(a,b)} + \beta_b}{\sum_w (n_{z_b*w}^{-(a,b)} + \beta_w)} \propto \ldots$$

LDA: сэмплирование по Гиббсу

- ...и теперь это всё можно выкинуть, потому что они не зависят от текущих сэмплов...

$$\ldots \propto \frac{\left(n_{z_b a *}^{-(a,b)} + \alpha_{z_b}\right)\left(n_{z_b * b}^{-(a,b)} + \beta_b\right)}{\sum_w \left(n_{z_b * w}^{-(a,b)} + \beta_w\right)} =$$

$$= \frac{\left(n_{z_b a *}^{-(a,b)} + \alpha_{z_b}\right)\left(n_{z_b * b}^{-(a,b)} + \beta_b\right)}{n_{z_b **}^{-(a,b)} + \sum_w \beta_w}.$$

LDA: сэмплирование по Гиббсу

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

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

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

где $n_{-w,t}^{(d)}$ – число слов в документе d, выбранных по теме t, а $n_{-w,t}^{(w)}$ – 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 $\boldsymbol{z}_{-w}$.

LDA: сэмплирование по Гиббсу

- Из сэмплов затем можно оценить переменные модели

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

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

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

- Сэмплирование по Гиббсу обычно проще расширить на новые модификации LDA, но вариационный подход быстрее и часто стабильнее.

Author-Topic model

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

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

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

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 $\boldsymbol{z}_{-w}$, $\mathbf{x}_{-w}$.