自然言語処理のための Gibbs サンプリング
Version 0.03

村脇 有吾

最終更新: 2012 年 11 月 5 日

今日の内容

- 多項分布から Dirichlet 過程までの続き
- 例によって図は板書する方向
- Gibbs サンプリングの理論的背景は端折る
- 引き続き例は LDA と Unigram 単語分割モデル
- 目標は Type-based サンプリングの理解

前回の復習 1/2

Dirichlet-多項分布の生成確率は以下で求まる (Dirichlet 過程の場合もほとんど同じ)。

$$p(\boldsymbol{x}|\boldsymbol{\alpha}) = \frac{\Gamma(A)}{\Gamma(n(\boldsymbol{x}) + A)} \prod_{k=1}^{K} \frac{\Gamma(n_k(\boldsymbol{x}) + \alpha_k)}{\Gamma(\alpha_k)}$$

LDA や Unigram 単語分割モデルは、Dirichlet(過程)-多項分布の組み合わせにすぎない。

前回の復習 2/2

D 個の文書、K 個のトピックからなる LDA で、単語列 W とトピック割り当て Z の生成確率は、$D + K$ 個の Dirichlet-多項分布の組み合わせで表される。

$$
\begin{aligned}
p(\boldsymbol{W}, \boldsymbol{Z}|\alpha, \boldsymbol{\beta}) &= \prod_{i=1}^{D} \left(\frac{\Gamma(\alpha_{(\cdot)})}{\Gamma(n_{i,(\cdot)}^{(\cdot)} + \alpha_{(\cdot)})} \prod_{k=1}^{K} \frac{\Gamma(n_{i,(\cdot)}^{k} + \alpha_k)}{\Gamma(\alpha_k)} \right) \\
&\quad \times \prod_{k=1}^{K} \left(\frac{\Gamma(\beta_{(\cdot)})}{\Gamma(n_{(\cdot),(\cdot)}^{k} + \beta_{(\cdot)})} \prod_{v} \frac{\Gamma(n_{(\cdot),v}^{k} + \beta_v)}{\Gamma(\beta_v)} \right)
\end{aligned}
$$

Unigram 単語分割モデルの場合、分割なしのコーパス $\boldsymbol{C}$、これに対応する単語列 $\boldsymbol{W}$ の生成確率は

$$
p(\boldsymbol{C}, \boldsymbol{W}|\alpha, G_0) = \frac{\Gamma(\alpha)}{\Gamma(n_{(\cdot)}(\boldsymbol{W}) + \alpha)} \prod_{l \in voc(\boldsymbol{W})} \frac{\Gamma(n_l(\boldsymbol{W}) + \alpha G_0(l))}{\Gamma(\alpha G_0(l))}
$$

ここで $voc(\boldsymbol{W})$ は $\boldsymbol{W}$ に含まれる単語ラベルの集合を返す関数。G_0 は任意の文字列からなる単語に適当な確率を与える分布。

Inference: 最適化問題

モデルを設定し、観測データが与えられたとき、モデルに一番フィットする隠れ状態を求めたい。例えば、Unigram 単語分割の場合、$\mathrm{argmax}_{\boldsymbol{W}}\, p(\boldsymbol{W}|\boldsymbol{C}, \alpha, G_0) = \mathrm{argmax}_{\boldsymbol{W}}\, p(\boldsymbol{C}, \boldsymbol{W}|\alpha, G_0)$ を求めたい。どうすればよいか。

- 解析的に求める (できるなら教えて!)
- $\boldsymbol{W}$ を総当りで試して max を求める (組合せ爆発により実際上不可能)
- 近似的に探索 (Gibbs サンプリングはこれ)

Gibbs サンプリングの実行の雰囲気

1. 適当に $\boldsymbol{W}$ を初期化 ($\boldsymbol{W}_1$ とする)
2. $\boldsymbol{W}_i$ を局所的に変更した $\boldsymbol{W}_{i+1}$ を確率的に得る

$p(\boldsymbol{W}_i|\boldsymbol{C}, \alpha, G_0)$ は上がったり下がったりするが、長期的には上がっていく。

そもそもサンプリングとは何か

確率分布からサンプルを得る。例えば、サイコロを振って出た目を記録する。実装的には、rand 関数の戻り値 r $(0 \leq r < 1)$ を適当に加工して目的の値を得る。

例えば $\theta = (0.1, 0.6, 0.3)$ からのサンプル x は以下の手続きで得られる。

$$x = \begin{cases} 1 & 0 \leq r < 0.1 \\ 2 & 0.1 \leq r < 0.7 \\ 3 & 0.7 \leq r < 1 \end{cases}$$

Dirichlet-多項分布の予測分布からのサンプリング

Dirichlet-多項分布の予測確率は以下で与えられる (復習):

$$p(x' = k' | \boldsymbol{x}, \boldsymbol{\alpha}) = \frac{n_{k'}(\boldsymbol{x}) + \alpha_{k'}}{n(\boldsymbol{x}) + \alpha_{(\cdot)}}$$

いま $\boldsymbol{x} = (\text{赤}, \text{赤}, \text{黄}, \text{赤}, \text{赤})$, $\boldsymbol{\alpha} = (\alpha_{\text{赤}}, \alpha_{\text{黄}}, \alpha_{\text{緑}}) = (1, 1, 1)$ とすると、

$$\begin{aligned}
p(x' = \text{赤} | \boldsymbol{x}, \boldsymbol{\alpha}) &= \frac{4 + 1}{5 + 3} = 0.625 \\
p(x' = \text{黄} | \boldsymbol{x}, \boldsymbol{\alpha}) &= \frac{1 + 1}{5 + 3} = 0.25 \\
p(x' = \text{緑} | \boldsymbol{x}, \boldsymbol{\alpha}) &= \frac{0 + 1}{5 + 3} = 0.125
\end{aligned}$$

したがって $\boldsymbol{\theta} = (0.625, 0.25, 0.125)$ からサンプルを得ると、それがこの予測分布からのサンプルになっている。

Gibbs サンプリング

Dirichlet(過程)-多項分布の組み合わせ $p(\boldsymbol{Z}) = p(z_1, \ldots, z_N)$ から $\boldsymbol{Z}$ を直接サンプリングするのは難しい。ただし、z_i を $p(z_i|\boldsymbol{Z}^{-i})$ からをサンプリングするのは容易。ここで $\boldsymbol{Z}^{-i}$ は $\boldsymbol{Z}$ から z_i を除いたもの。そこで、$p(z_i|\boldsymbol{Z}^{-i})$ からのサンプリングを用いて、$\boldsymbol{Z}$ のサンプルを得たい。これを実現するのが Gibbs サンプリング。

1. Z を適当に初期化
2. 各ステップ $\tau = 1, \ldots, T$ で以下を行う
   - z_i を $z_i \sim p(z_i|\boldsymbol{Z}^{-i})$ からのサンプルで更新
   - 新たな $\boldsymbol{Z}$ を用いて同様に z_{i+1} を更新
   - これをすべての $i = 1, \ldots N$ に対して行う

これを充分に繰り返すと $\boldsymbol{Z}_\tau$ は真の確率分布 $p(\boldsymbol{Z})$ からのサンプルとなる。理論的背景については PRML 11.3 を参照。

交換可能性 (Exchangeability) 再び

Gibbs サンプリングのキモは、$p(z_i|\mathbf{Z}^{-i})$ から容易にサンプリングできることである。Dirichlet-多項分布 (とその組み合わせ) の場合、交換可能性がなりたつことから容易にサンプリングできる。
$(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_N)$ を観測する確率は、順番を入れ換えた $(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_N, z_i)$ を観測する確率と等しい。
$p(z_i|\mathbf{Z}^{-i})$ は $(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_N)$ を観測したもとでの z_i の予測確率を表している。

LDA のサンプリング 1/2

$$
\begin{aligned}
& p(z_i = k' | \boldsymbol{Z}^{-i}, \boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
\propto \quad & p(z_i = k', \boldsymbol{Z}^{-i}, \boldsymbol{W} | \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
= \quad & \prod_{j=1}^{D} \left(\frac{\Gamma(\alpha_{(\cdot)})}{\Gamma(n_{j,(\cdot)}^{(\cdot)}(\boldsymbol{Z}^{-i}) + I(w_i \in \boldsymbol{D}_j) + \alpha_{(\cdot)})} \right. \\
& \left. \prod_{k=1}^{K} \frac{\Gamma(n_{j,(\cdot)}^{k}(\boldsymbol{Z}^{-i}) + I(k' = k \,\&\, w_i \in \boldsymbol{D}_j) + \alpha_k)}{\Gamma(\alpha_k)} \right) \\
& \times \prod_{k=1}^{K} \left(\frac{\Gamma(\beta_{(\cdot)})}{\Gamma(n_{(\cdot),(\cdot)}^{k}(\boldsymbol{Z}^{-i}) + I(k' = k) + \beta_{(\cdot)})} \right. \\
& \left. \prod_{v} \frac{\Gamma(n_{(\cdot),v}^{k}(\boldsymbol{Z}^{-i}) + I(k' = k \,\&\, w_i = v) + \beta_v)}{\Gamma(\beta_v)} \right)
\end{aligned}
$$

i はコーパス全体での通し番号。$\boldsymbol{D}_j$ は j 番目の文書の単語の集合。
ほとんどの項は z_i と無関係なので無視できる。後半へ続く。

LDA のサンプリング 2/2

$$
\begin{aligned}
& p(z_i = k' | \boldsymbol{Z}^{-i}, \boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
\propto\ & \frac{\Gamma(n^{(\cdot)}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + \alpha_{(\cdot)})}{\Gamma(n^{(\cdot)}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + 1 + \alpha_{(\cdot)})} \times \frac{\Gamma(n^{k'}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + 1 + \alpha_{k'})}{\Gamma(n^{k'}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + \alpha_{k'})} \\
& \times \frac{\Gamma(n^{k'}_{(\cdot),(\cdot)}(\boldsymbol{Z}^{-i}) + \beta_{(\cdot)})}{\Gamma(n^{k'}_{(\cdot),(\cdot)}(\boldsymbol{Z}^{-i}) + 1 + \beta_{(\cdot)})} \times \frac{\Gamma(n^{k'}_{(\cdot),v'}(\boldsymbol{Z}^{-i}) + 1 + \beta_{v'})}{\Gamma(n^{k'}_{(\cdot),v'}(\boldsymbol{Z}^{-i}) + \beta_{v'})} \\
=\ & \frac{n^{k'}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + \alpha_{k'}}{n^{(\cdot)}_{j',(\cdot)}(\boldsymbol{Z}^{-i}) + \alpha_{(\cdot)}} \times \frac{n^{k'}_{(\cdot),v'}(\boldsymbol{Z}^{-i}) + \beta_{v'}}{n^{k'}_{(\cdot),(\cdot)}(\boldsymbol{Z}^{-i}) + \beta_{(\cdot)}}
\end{aligned}
$$

ここで $v' = w_i$, j' は w_i の属す文書のインデックスとする。
第 1 項は文書 $\boldsymbol{D}_j$ に属すトピックの Dirichlet-多項分布に k' を追加するときの予測確率、第 2 項はトピック k' に属す単語の Dirichlet-多項分布に v' を追加するときの予測確率となっている (実際には第 1 項の分母は k' によらないので無視できる)。

LDA のサンプリングの直感的な説明

$$\frac{n^{k'}_{j',(\cdot)}(\boldsymbol{Z}^{-i})+\alpha_{k'}}{n^{(\cdot)}_{j',(\cdot)}(\boldsymbol{Z}^{-i})+\alpha_{(\cdot)}} \times \frac{n^{k'}_{(\cdot),v'}(\boldsymbol{Z}^{-i})+\beta_{v'}}{n^{k'}_{(\cdot),(\cdot)}(\boldsymbol{Z}^{-i})+\beta_{(\cdot)}}$$

- z_i の値は、第 1 項と第 2 項の積を大きくするような k に (確率的に) なりやすい
- 第 1 項を大きくするには、文書 $\boldsymbol{D}_j$ の中で頻出する k を選べばよい
- 第 2 項を大きくするには、w_i が頻出するトピック k を選べばよい
- サンプリングを繰り返すと、$\boldsymbol{Z}$ の値は、これらを全体的に満たすようになっていく

LDA の実装

以下のデータを管理する。

- $\boldsymbol{W}$ と現在の $\boldsymbol{Z}$
- $n^k_{j,(\cdot)}$ (文書 D_j 中のトピック k の割り当て数)
- $(n^{(\cdot)}_{j,(\cdot)}$ (文書 D_j 中の単語総数))
- $n^k_{(\cdot),v}$ (トピック k の単語 v の頻度)
- $n^k_{(\cdot),(\cdot)}$ (トピック k の単語総数)

z_i のサンプリングは以下の手順で行う。

1. 現在の z_i に対応するカウントを decrement
2. $p(z_i|\boldsymbol{Z}^{-i}, \boldsymbol{W}, \alpha, \beta)$ (に比例する値) を K 通り計算
3. 乱数を振って新たな z_i を決め、$\boldsymbol{Z}$ を更新
4. 新たな z_i に対応するカウントを increment

焼きなまし法 (Simulated annealing)

サンプリングで得られるのは、あくまで確率分布からのサンプルであり、明示的に最大値を求めているわけではない。LDA の場合、$p(\boldsymbol{Z}|\boldsymbol{W}, \alpha, \beta)$ からのサンプルであり、この値を小さくするような $\boldsymbol{Z}$ も (非常に小さな確率で) 得られる。ただ、LDA などの確率分布は非常に peaky であり、単にサンプリングを繰り返しているだけで極大値付近に滞留することが多い。

真面目に最大値を近似的に求めるための拡張として焼きなまし法がある (詳細省略)。

log probability の追跡

サンプリングがうまく行っているかを確認するには、以下の値を追跡する。

$$
\begin{aligned}
p(\boldsymbol{W}, \boldsymbol{Z}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \quad = \quad & \prod_{i=1}^{D} \left(\frac{\Gamma(\alpha_{(\cdot)})}{\Gamma(n_{i,(\cdot)}^{(\cdot)} + \alpha_{(\cdot)})} \prod_{k=1}^{K} \frac{\Gamma(n_{i,(\cdot)}^{k} + \alpha_k)}{\Gamma(\alpha_k)} \right) \\
& \times \prod_{k=1}^{K} \left(\frac{\Gamma(\beta_{(\cdot)})}{\Gamma(n_{(\cdot),(\cdot)}^{k} + \beta_{(\cdot)})} \prod_{v} \frac{\Gamma(n_{(\cdot),v}^{k} + \beta_v)}{\Gamma(\beta_v)} \right)
\end{aligned}
$$

この値が下がり続けていたりしたらバグっている可能性が高い。

実装上のヒント: $p(\boldsymbol{W}, \boldsymbol{Z}|\boldsymbol{\alpha}, \boldsymbol{\beta})$ は非常に小さな値なので、そのままでは underflow する。代わりに $log p(\boldsymbol{W}, \boldsymbol{Z}|\boldsymbol{\alpha}, \boldsymbol{\beta})$ を求める。

$log\Gamma$ は C の lgamma で実装されている。Perl の場合は外部ライブラリを使う (e.g. Math::Cephes の lgam)。

Unigram 単語分割モデルのサンプリング 1/3

準備として、単語列 $\boldsymbol{W} = (w_1, w_2, \ldots)$ と等価な 2 値変数の列 $\boldsymbol{B} = (b_1, b_2, b_3, \ldots)$ を考える。各変数は文字の間の分割状態に対応し、1 なら境界あり、0 なら境界なしを表す。例えば、単語列 (京都, 大学, だ) は $(0, 1, 0, 1)$ に対応する (文頭、文末は固定なので無視)。

Unigram 単語分割モデルのサンプリングは $\boldsymbol{B}$ に対して行う。例えば、2 番目の値を flip して $\boldsymbol{B} = (0, 0, 0, 1)$ とすると、$\boldsymbol{W} =$ (京都大学, だ) になる。

b_i のサンプリングは、$\boldsymbol{W}$ の対応する局所領域を 2 単語 (京都, 大学) とするか、1 単語 (京都大学) とするかに対応する。以下ではサンプリング箇所を . で示し、京都.大学 と表記する。

Unigram 単語分割モデルのサンプリング 2/3

b_i に対応する局所領域を $a.b$ (e.g. 京.都) としたとき、分割しない確率は以下のように、$\boldsymbol{W}^{-ab}$ を観測したもとでの単語 ab の予測確率に比例する。

$$
\begin{aligned}
& P(b_i = 0 | \boldsymbol{B}^{-i}, \boldsymbol{C}, \alpha, G_0) \\
= \quad & P(ab | \boldsymbol{W}^{-ab}, \boldsymbol{C}, \alpha, G_0) \\
\propto \quad & P(ab, \boldsymbol{W}^{-ab}, \boldsymbol{C} | \alpha, G_0) \\
= \quad & \frac{\Gamma(\alpha)}{\Gamma(n_{(\cdot)}(\boldsymbol{W}^{-ab}) + 1 + \alpha)} \\
& \times \prod_{l \in voc(\boldsymbol{W}^{-ab}) \cup \{ab\}} \frac{\Gamma(n_l(\boldsymbol{W}^{-ab}) + I(l = ab) + \alpha G_0(l))}{\Gamma(\alpha G_0(l))} \\
\propto \quad & \frac{\Gamma(n_{(\cdot)}(\boldsymbol{W}^{-ab}) + \alpha)}{\Gamma(n_{(\cdot)}(\boldsymbol{W}^{-ab}) + 1 + \alpha)} \times \frac{\Gamma(n_{ab}(\boldsymbol{W}^{-ab}) + 1 + \alpha G_0(ab))}{\Gamma(n_{ab}(\boldsymbol{W}^{-ab}) + \alpha G_0(ab))} \\
= \quad & \frac{n_{ab}(\boldsymbol{W}^{-ab}) + \alpha G_0(ab)}{n_{(\cdot)}(\boldsymbol{W}^{-ab}) + \alpha}
\end{aligned}
$$

Unigram 単語分割モデルのサンプリング 3/3

分割する確率も同様だが、a = b のときに注意が必要。

$$
\begin{aligned}
& P(b_i = 1 | \boldsymbol{B}^{-i}, \boldsymbol{C}, \alpha, G_0) \\
= \quad & P(a, b | \boldsymbol{W}^{-ab}, \boldsymbol{C}, \alpha, G_0) \\
\propto \quad & \frac{n_a(\boldsymbol{W}^{-ab}) + \alpha G_0(a)}{n_{(\cdot)}(\boldsymbol{W}^{-ab}) + \alpha} \times \frac{n_b(\boldsymbol{W}^{-ab}) + I(a = b) + \alpha G_0(b)}{n_{(\cdot)}(\boldsymbol{W}^{-ab}) + 1 + \alpha}
\end{aligned}
$$

$\boldsymbol{W}^{-ab}$ を観測したもとで a を観測し、次に $(\boldsymbol{W}^{-ab}, a)$ を観測したもとで b を観測する予測確率となっている (LDA の場合と同様、第 1 項の分母は無視できる)。

Unigram 単語分割モデルのサンプリングの直感的な説明

$$\frac{n_{ab}(\boldsymbol{W}^{-ab})+\alpha G_0(ab)}{n_{(\cdot)}(\boldsymbol{W}^{-ab})+\alpha} \text{ vs. } \frac{n_a(\boldsymbol{W}^{-ab})+\alpha G_0(a)}{n_{(\cdot)}(\boldsymbol{W}^{-ab})+\alpha} \times \frac{n_b(\boldsymbol{W}^{-ab})+I(a=b)+\alpha G_0(b)}{n_{(\cdot)}(\boldsymbol{W}^{-ab})+1+\alpha}$$

- 分母に注目すると、1 以下の数を 2 つかける (分割する) より、1 つだけの (分割しない) 方が大きくなる。極端な話、1 文を 1 単語にすれば掛け算せずにすんで大きな値になる。
- 分子の $n_x(\boldsymbol{W}^{-ab})$ は、x が $\boldsymbol{W}^{-ab}$ の中で出てきた回数だから、これを大きくしようと思ったら、なるべく区切った方がよい。極端な話、1 文字 1 単語にすれば、再利用されまくる。
- Unigram 単語分割モデルは両者のバランスをとっている。

Unigram 単語分割モデルの実装

以下のデータを管理する。

- 現在の単語列 $\boldsymbol{W}$
- n_l (各単語の頻度)
- $n_{(\cdot)}$ (全単語数)

z_i のサンプリングは以下の手順で行う。

1. 現在の b_i に対応するカウントを decrement ($b_i = 0$ のときは 1 単語、$b_i = 1$ のときは 2 単語)
2. $P(b_i|\boldsymbol{B}^{-i}, \boldsymbol{C}, \alpha, G_0)$ を 2 通り計算
3. 乱数を振って新たな b_i を決め、$\boldsymbol{W}$ を更新
4. 新たな b_i に対応するカウントを increment

Type-based サンプリング: 動機

普通の Gibbs サンプリングは局所的な探索を繰り返すので、山と山の間に深い谷があるとなかなか移れない。
例えば、Unigram 単語分割モデルにおいて、$\boldsymbol{W}$ 中で a.b (e.g. 京.都) が 100 回出現し、すべてマージすると $p(\boldsymbol{W}|\boldsymbol{C}, \alpha, G_0)$ が最大値になるとする。いま、a.b が 97 回分割されているとする。(a, b) をサンプリングすると、$97 - 1$ 回の出現の影響により $P(b_i = 1|\ldots) \approx 1$ となり、ほぼ必ず分割される。したがって、すべてを分割する状態に収束し、すべてマージする状態に移る可能性は 0 に近い。つまり極所解にとらわれてしまう。

Type-based サンプリング (Liang et al., 2010) では、この 100 個の出現をまとめてサンプリングすることにより、谷を一気に飛び越える。

Type-based サンプリング 1/3

$a.b$ をサンプリングするとき、単語列 a, b および ab は同じ型 (type) に属す。同じ型に属すものの集合を $\boldsymbol{S}$ とする。Type-based サンプリングでは $\boldsymbol{S}$ をまとめてサンプリングする。
$\boldsymbol{S}$ に対応する 2 値変数の集合を $\boldsymbol{b_S}$ とする。$\boldsymbol{b_S}$ に対応する $|\boldsymbol{S}|$ 個の局所領域を $\boldsymbol{W}$ から除いたものを $\boldsymbol{W}^{-\boldsymbol{S}}$ とする。
Type-based サンプリングのキモは、$\boldsymbol{b_S}$ に属す各 b_i が交換可能なことである。久しぶりに (サンプルではなく) 観測数の分布を考え、以下の 2 段階でサンプリングを行う。

1. 最初に観測数の分布から分割数 k $(0 \leq k \leq |\boldsymbol{S}|)$ をサンプル
2. 次に $|\boldsymbol{S}|$ 個の状態に対して k 個の分割をランダムに割り当て

Type-based サンプリング 2/3

分割数 k の分布は以下の通り (ただし $a \neq b$)。

$$
\begin{aligned}
& P(k|\boldsymbol{B}^{-S}, \boldsymbol{C}, \alpha, G_0) \\
= \; & P(k|\boldsymbol{W}^{-S}, \boldsymbol{C}, \alpha, G_0) \\
\propto \; & \binom{|\boldsymbol{S}|}{k} \frac{\Gamma(n_{(\cdot)}(\boldsymbol{W}^{-S}) + \alpha)}{\Gamma(n_{(\cdot)}(\boldsymbol{W}^{-S}) + |\boldsymbol{S}| + k + \alpha)} \times \frac{\Gamma(n_a(\boldsymbol{W}^{-S}) + k + \alpha G_0(a))}{\Gamma(n_a(\boldsymbol{W}^{-S}) + \alpha G_0(a))} \\
& \times \frac{\Gamma(n_b(\boldsymbol{W}^{-S}) + k + \alpha G_0(b))}{\Gamma(n_b(\boldsymbol{W}^{-S}) + \alpha G_0(b))} \times \frac{\Gamma(n_{ab}(\boldsymbol{W}^{-S}) + |\boldsymbol{S}| - k + \alpha G_0(ab))}{\Gamma(n_{ab}(\boldsymbol{W}^{-S}) + \alpha G_0(ab))}
\end{aligned}
$$

$a = b$ のときは、第 3,4 項

$$
\frac{\Gamma(n_a(\boldsymbol{W}^{-S}) + k + \alpha G_0(a))}{\Gamma(n_a(\boldsymbol{W}^{-S}) + \alpha G_0(a))} \times \frac{\Gamma(n_b(\boldsymbol{W}^{-S}) + k + \alpha G_0(b))}{\Gamma(n_b(\boldsymbol{W}^{-S}) + \alpha G_0(b))}
$$

を以下で置き換える

$$
\frac{\Gamma(n_a(\boldsymbol{W}^{-S}) + 2k + \alpha G_0(a))}{\Gamma(n_a(\boldsymbol{W}^{-S}) + \alpha G_0(a))}
$$

Type-based サンプリング 3/3

分割数の分布からのサンプリングにより分割数 k が求まる。次に S に対して k を割り振る。どのような分割の割り当ても等確率なので、ランダムに割り当てる。

Type-based サンプリング: 衝突に注意

特殊な場合には注意が必要である。例えば a_1, a_2, a_3 $(a_1 = a_2 = a_3)$ について、$a_1.a_2$ をサンプリングすることを考える。このとき、$a_2.a_3$ は $a_1.a_2$ と同じ型に属す。しかし、a_2 を共有しているため、$a_1.a_2$ と $a_2.a_3$ は交換可能ではない。仕方がないので、$a_1.a_2$ をサンプリングするときには、$a_2.a_3$ はサンプリング対象から外す。

Type-based サンプリングの実装

以下のデータを管理する。

- 現在の単語列 $\boldsymbol{W}$
- n_l (各単語の頻度)
- $n_{(\cdot)}$ (全単語数)
- 型を管理するデータ構造

サンプリングは以下の手順で行う。

1. b_i について、同じ型で、衝突のないものの集合 $\boldsymbol{S}$ を集める
2. $\boldsymbol{S}$ に対応するカウントを decrement
3. $P(k|\boldsymbol{B}^{-S}, \boldsymbol{C}, \alpha, G_0)$ を $|\boldsymbol{S}| + 1$ 通り計算
4. 乱数を振って k を決め、分割をランダムに割り振る
5. 割り当てられた分割に応じて $\boldsymbol{W}$ を更新
6. 割り当てられた分割に応じてカウントを increment