2021.10.10
市民講座

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

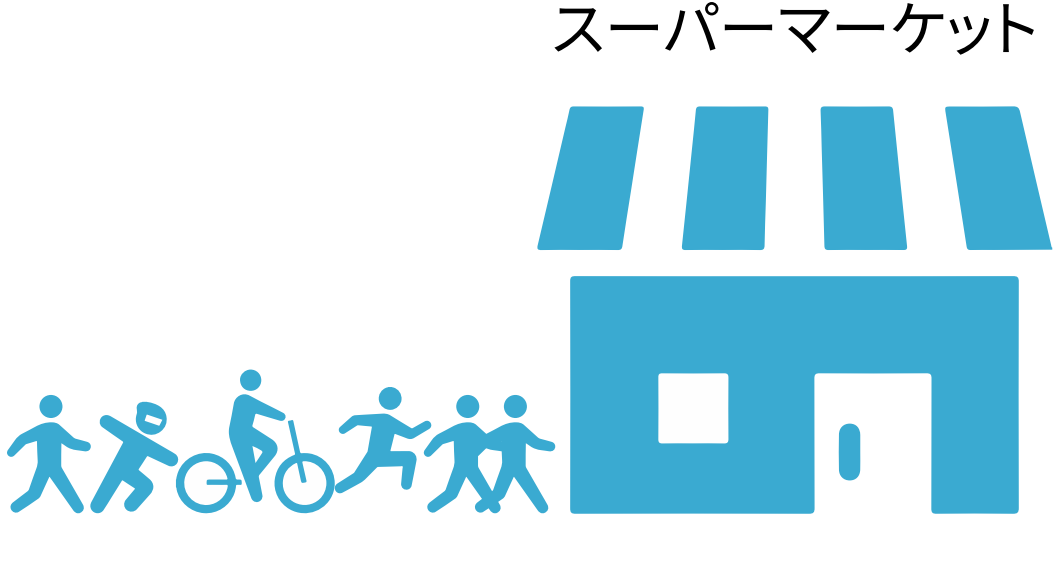
データに隠れた規則性を探せ！
―パターンが織りなす構造とAI―

杉山麿人（国立情報学研究所）

購買データ解析の例

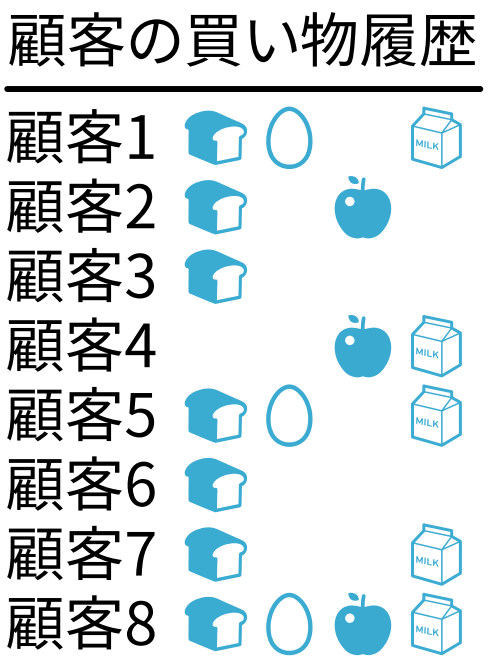
顧客の買い物履歴

顧客1
顧客2
顧客3
顧客4
顧客5
顧客6
顧客7
顧客8



どの商品が売れている？

顧客の買い物履歴

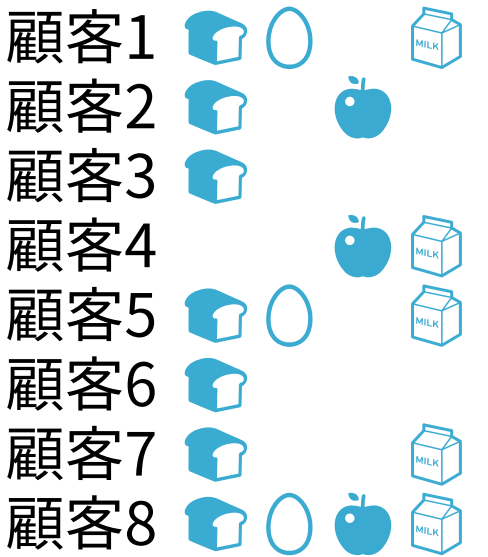


商品の購入**頻度**

{🍞}: 7/8=0.875
{🥚}: 3/8=0.375
{🍎}: 3/8=0.375
{🥛}: 5/8=0.625

商品の組合せもチェック

顧客の買い物履歴

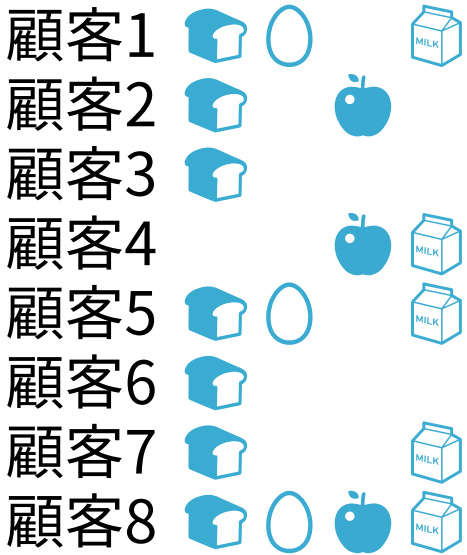


商品の購入**頻度**

{🍞}: 7/8=0.875　{🍞, 🥛}: 4/8=0.5
{🥯}: 3/8=0.375　{🥯, 🥛}: 3/8=0.375
{🍎}: 3/8=0.375　{🍞, 🥯}: 3/8=0.375
{🥛}: 5/8=0.625

商品の組合せ＝パターンのマイニング

顧客の買い物履歴



商品の購入**頻度**

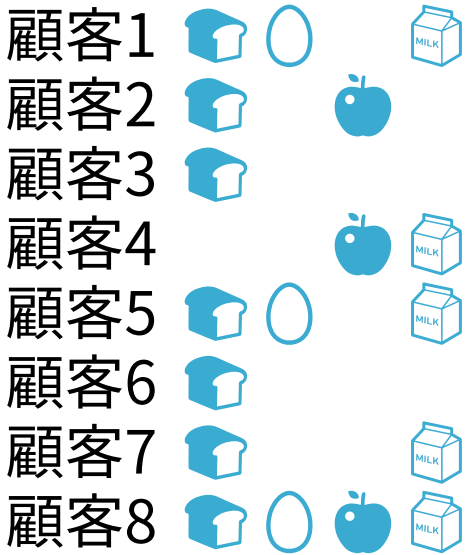
{🍞}: 7/8=0.875
{🥚}: 3/8=0.375
{🍎}: 3/8=0.375
{🥛}: 5/8=0.625

商品の組合せを**パターン**と呼ぶ

{🍞,🥛}: 4/8=0.5
{🥚,🥛}: 3/8=0.375
{🍞,🥚}: 3/8=0.375
{🍞,🥛,🥚}: 3/8 =0.375

頻出しているパターンを見つける

顧客の買い物履歴



商品の購入**頻度**

{🍞}: 7/8=0.875
{🥚}: 3/8=0.375
{🍎}: 3/8=0.375
{🥛}: 5/8=0.625

(例:頻度≥0.5)

商品の組合せを**パターン**と呼ぶ

{🍞,🥛}: 4/8=0.5
{🥚,🥛}: 3/8=0.375
{🍞,🥚}: 3/8=0.375

{🍞,🥛,🥚}: 3/8 =0.375

単純な全列挙は「組合せ爆発」を起こす

- 頻度が高いパターン（商品の組合せ）を全部見つけたい → 計算が大変！

商品の数	パターン数	おおよその計算時間
20	$\underbrace{2 \times 2 \times \cdots \times 2}_{20\text{個}} = 2^{20}$	0.00059 秒

単純な全列挙は「組合せ爆発」を起こす

- 頻度が高いパターン（商品の組合せ）を全部見つけたい → 計算が大変！

商品の数	パターン数	おおよその計算時間
20	$\underbrace{2 \times 2 \times \cdots \times 2}_{20\text{個}} = 2^{20}$	0.00059 秒
40	2^{40}	10.2 分

単純な全列挙は「組合せ爆発」を起こす

- 頻度が高いパターン（商品の組合せ）を全部見つけたい → 計算が大変！

商品の数	パターン数	おおよその計算時間
20	$\underbrace{2 \times 2 \times \cdots \times 2}_{20\text{個}} = 2^{20}$	0.00059 秒
40	2^{40}	10.2 分
50	2^{50}	7 日

単純な全列挙は「組合せ爆発」を起こす

- 頻度が高いパターン（商品の組合せ）を全部見つけたい → 計算が大変！

商品の数	パターン数	おおよその計算時間
20	$\underbrace{2 \times 2 \times \cdots \times 2}_{20\text{個}} = 2^{20}$	0.00059 秒
40	2^{40}	10.2 分
50	2^{50}	7 日
70	2^{70}	19,000 年

単純な全列挙は「組合せ爆発」を起こす

- 頻度が高いパターン（商品の組合せ）を全部見つけたい → 計算が大変！

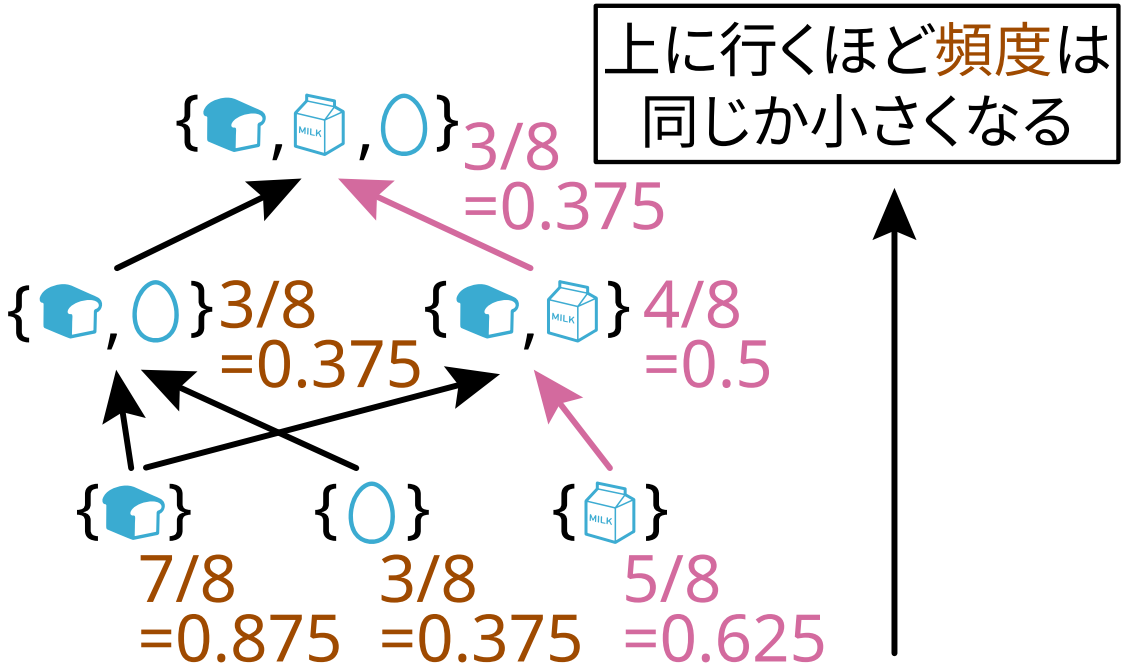
商品の数	パターン数	おおよその計算時間
20	$\underbrace{2 \times 2 \times \cdots \times 2}_{20\text{個}} = 2^{20}$	0.00059 秒
40	2^{40}	10.2 分
50	2^{50}	7 日
70	2^{70}	19,000 年
100	2^{100}	200 億年

（宇宙の年齢が約137億年…）

「頻度が持つ性質」を使って計算を高速化する

顧客の買い物履歴

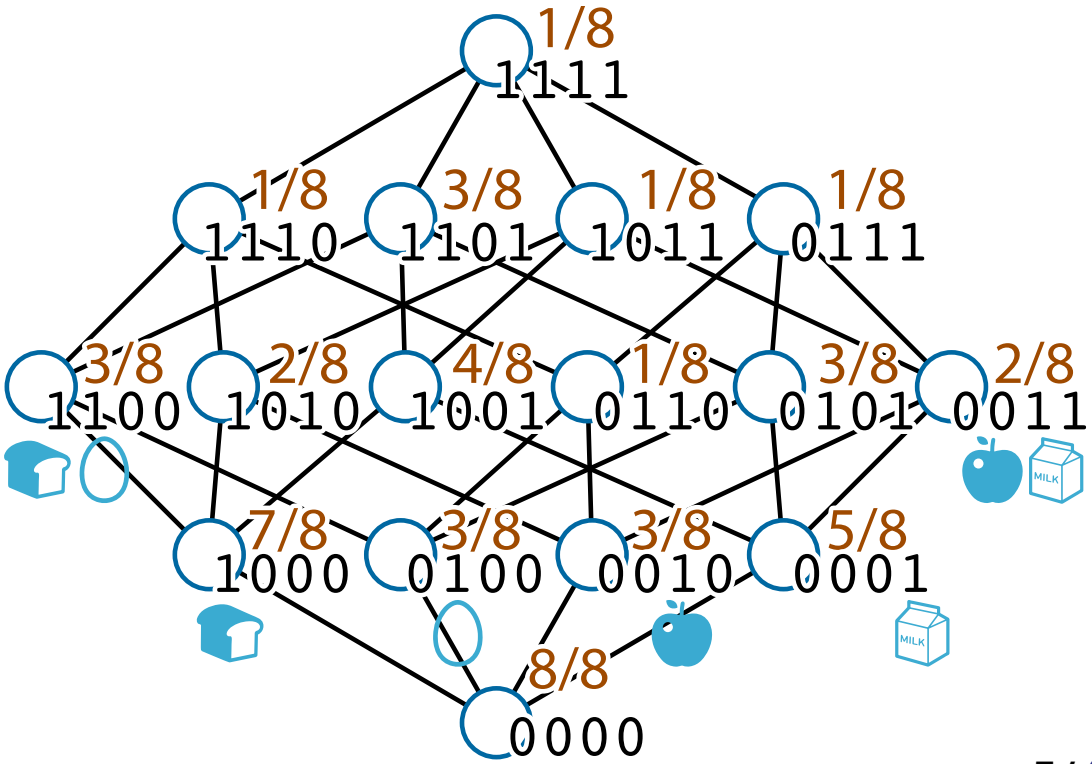
顧客1
顧客2
顧客3
顧客4
顧客5
顧客6
顧客7
顧客8



すべてのパターンからなる空間を考える

データ集合：

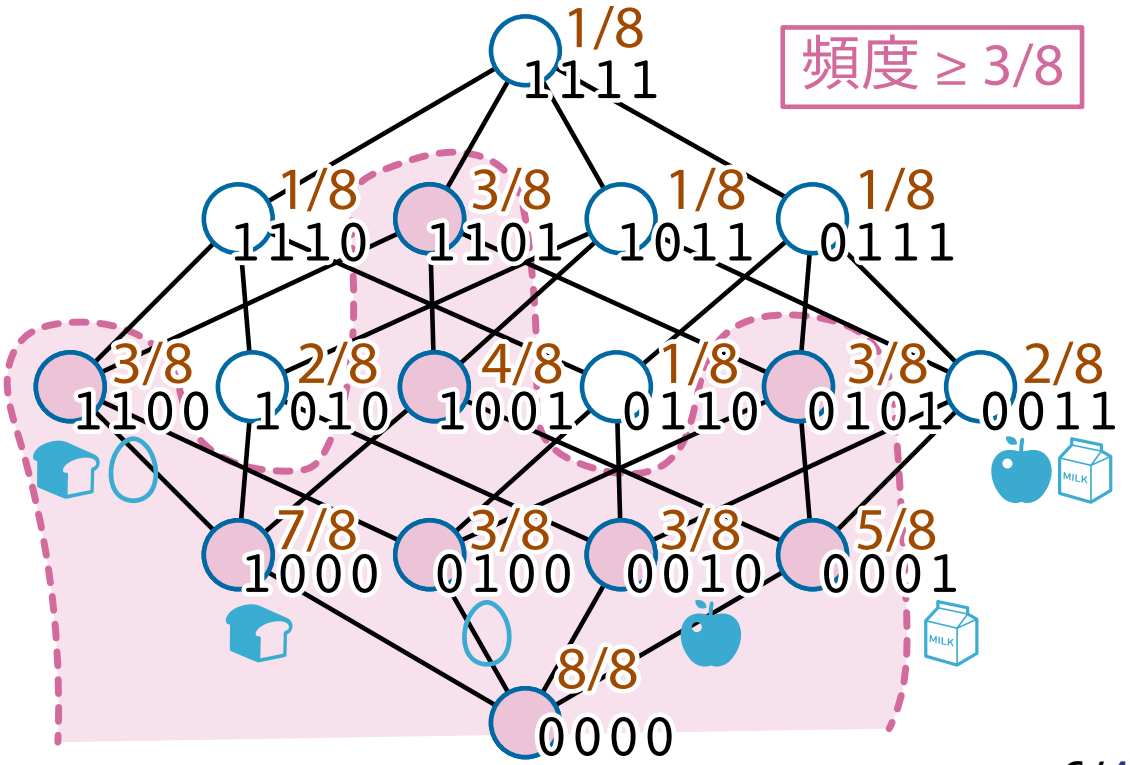
	A	B	C	D
ID 1:	1	1	0	1
ID 2:	1	0	1	0
ID 3:	1	0	0	0
ID 4:	0	0	1	1
ID 5:	1	1	0	1
ID 6:	1	0	0	0
ID 7:	1	0	0	1
ID 8:	1	1	1	1



頻出なパターンは下に固まる

データ集合：

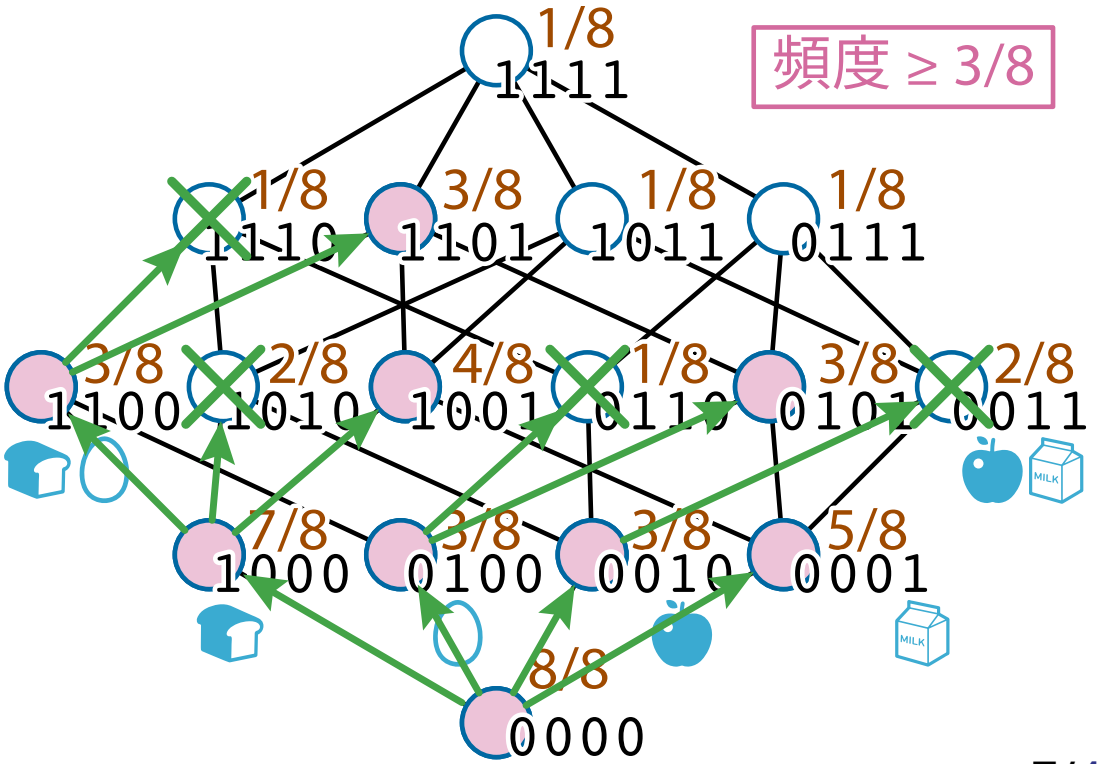
	A	B	C	D
ID 1:	1	1	0	1
ID 2:	1	0	1	0
ID 3:	1	0	0	0
ID 4:	0	0	1	1
ID 5:	1	1	0	1
ID 6:	1	0	0	0
ID 7:	1	0	0	1
ID 8:	1	1	1	1



下からチェックして閾値を下回ったらストップ

データ集合：

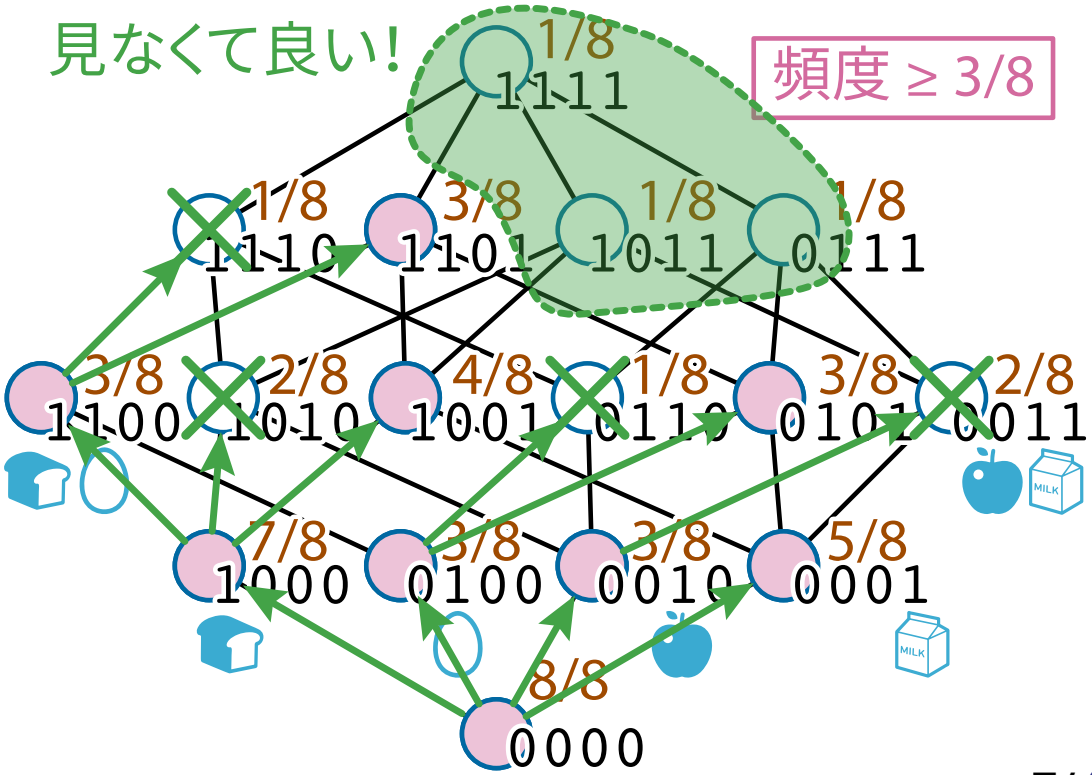
	A	B	C	D
ID 1:	1	1	0	1
ID 2:	1	0	1	0
ID 3:	1	0	0	0
ID 4:	0	0	1	1
ID 5:	1	1	0	1
ID 6:	1	0	0	0
ID 7:	1	0	0	1
ID 8:	1	1	1	1



チェック不要のパターンがある！

データ集合：

	A	B	C	D
ID 1:	1	1	0	1
ID 2:	1	0	1	0
ID 3:	1	0	0	0
ID 4:	0	0	1	1
ID 5:	1	1	0	1
ID 6:	1	0	0	0
ID 7:	1	0	0	1
ID 8:	1	1	1	1

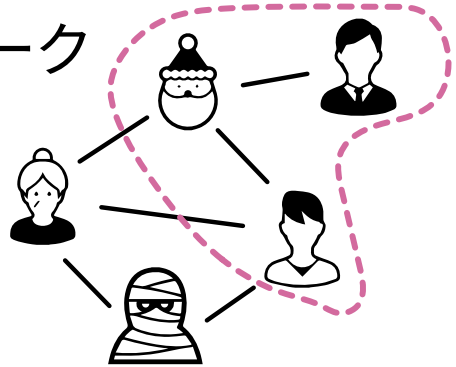


Apriori による高速化

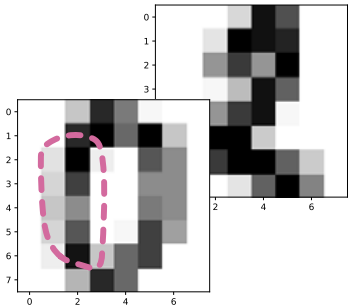
- 下から順にチェックしていって途中で止める方法を Apriori という
  - [Agrawal & Srikant, 1994]
- 商品数100のとき200億年かかっていた計算が，Aprioriを使うと数秒（ときには一秒以下）で計算できる
  - 得られる結果は全く同じ
  - パターンが持つ性質をうまく使うことで問題が解決できた
  - 世界最速のアルゴリズムは LCM （by 宇野先生@NII）
- 「組合せ爆発」の面白さ（恐ろしさ）については：
  - 『フカシギの数え方』おねえさんといっしょ！
  - https://www.youtube.com/watch?v=Q4gTV4r0zRs

様々なパターン（組合せ的な構造を持った対象）

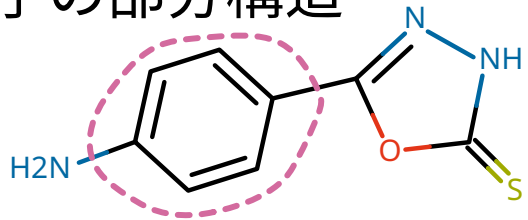
ソーシャルネットワークの部分グラフ



画像の一部分



分子の部分構造

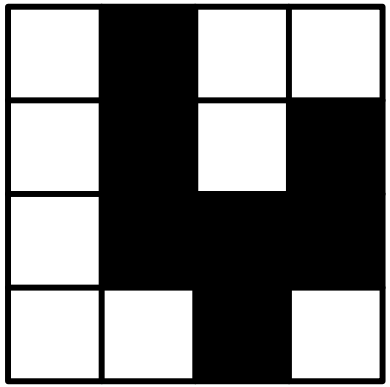


DNAの部分列

AGGTTTCCCT...
TCCAAAGGGA...

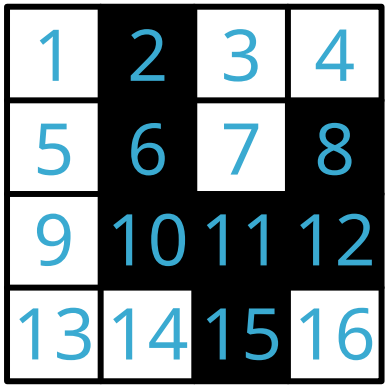
画像のベクトル化とパターン

4×4の
白黒画像



画像のベクトル化とパターン

4×4の
白黒画像

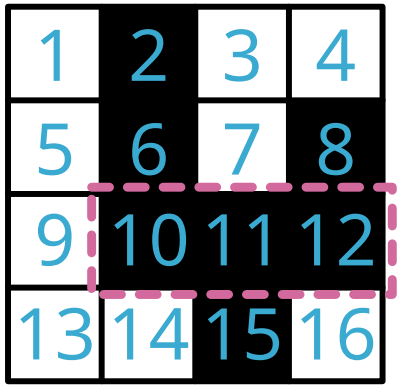


→ バイナリベクトル表現

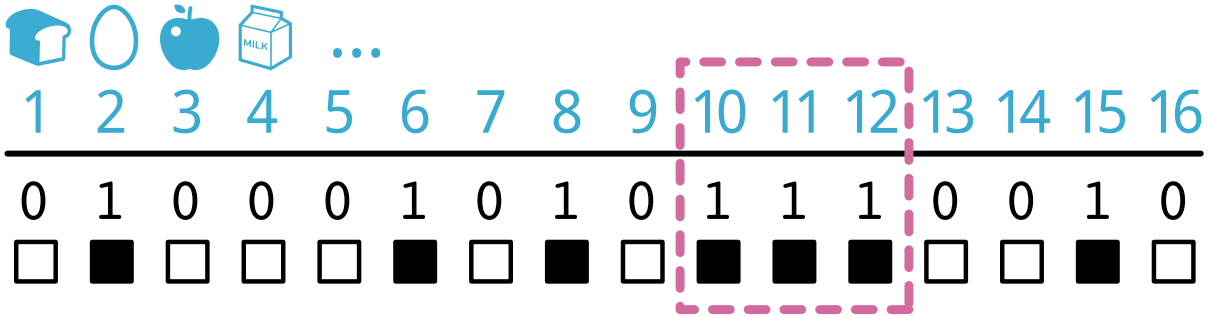
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	0	0	0	1	0	1	0	1	1	1	0	0	1	0

画像のベクトル化とパターン

4×4の
白黒画像



→ バイナリベクトル表現



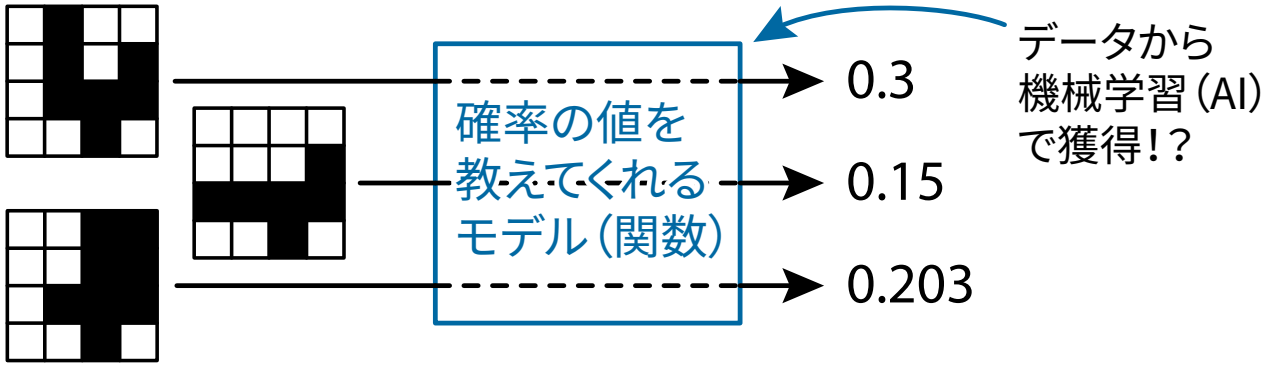
バイナリベクトル表現を用いることで,
購買データも画像データも同様に
扱うことができる(各画素と各商品が対応)

パターン解析の肝

- データ解析の現場では，パターンは**数え切れないほど無数にある**
  - 例：スーパーに商品が5,000種類 → パターン数は $2^{5000} \approx 1\underbrace{000\ldots\ldots000}_{0\text{が}1505\text{個}}$
  - 例：28x28の画像 → パターン数は $2^{784} \approx 1\underbrace{000\ldots\ldots000}_{0\text{が}236\text{個}}$
  - 有限ではあるが，とにかく数が多いので，どんなに高性能なスーパーコンピュータを使っても全部を調べることはできない
- パターンを相手にするデータ解析では，
  いかに**全体を見ないで一部分だけで答えを出せるか**が勝負！

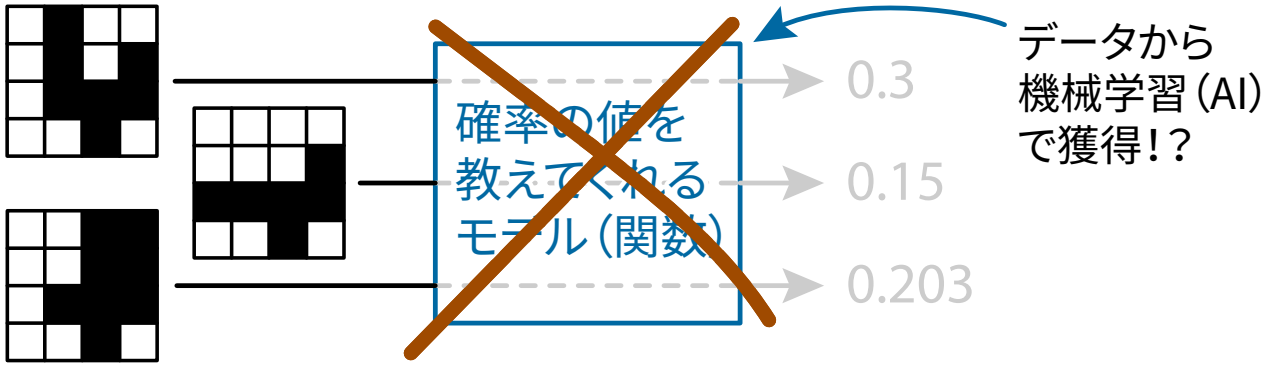
パターンが出現する「傾向」を分析する

- パターンの出現具合（確率）を推定したい



課題：確率の計算は不可能

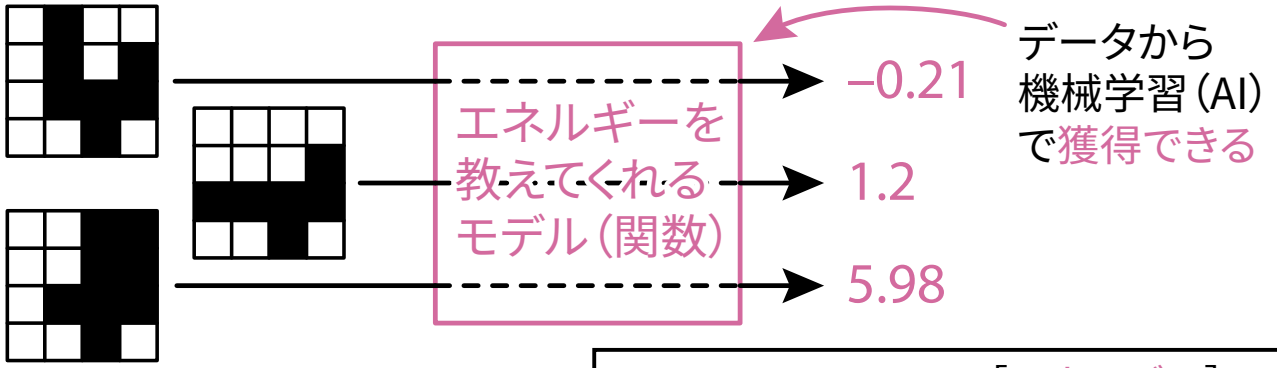
- パターンの出現具合（確率）を推定したい



- 「確率」値を得るには正規化が必要なので，これは残念ながら不可能
  - パターン全体を見る必要があるため

解決策：エネルギーを利用する

- パターンの出現具合（確率）を推定したい



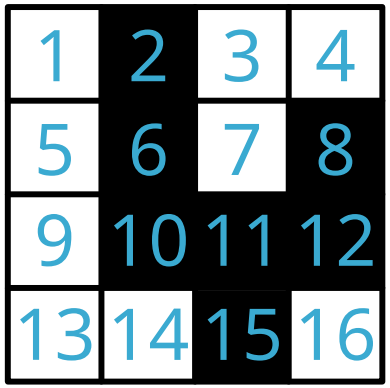
- 「エネルギー」を代わりに使う：

$$\text{確率} = \frac{e^{-[\text{エネルギー}]}}{\text{全パターンの } e^{-[\text{エネルギー}]} \text{ の和}}$$

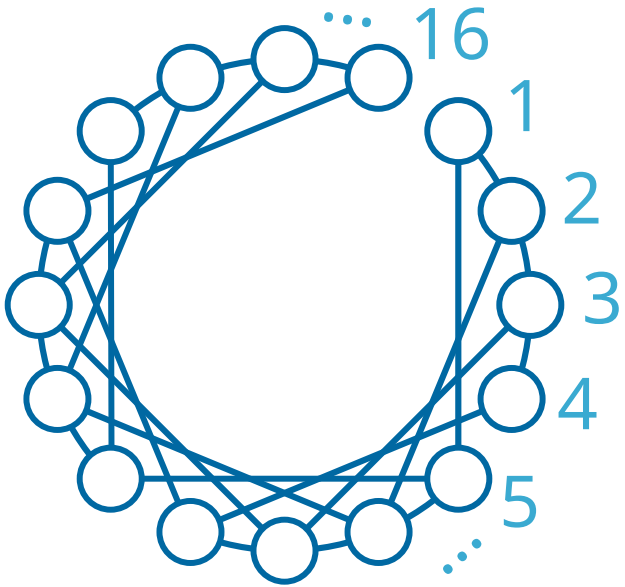
  - $-\infty$ から ∞ の値をとる
  - 小さいエネルギーは高い確率を意味する

マルコフ確率場でエネルギーを表現

4×4の白黒画像



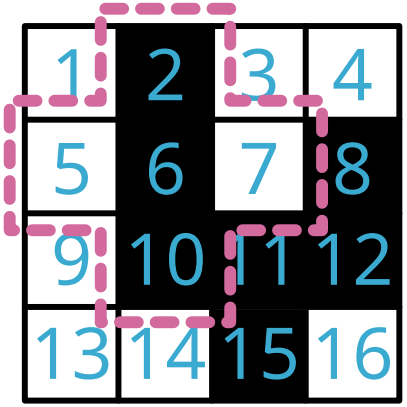
マルコフ確率場（ボルツマンマシン）



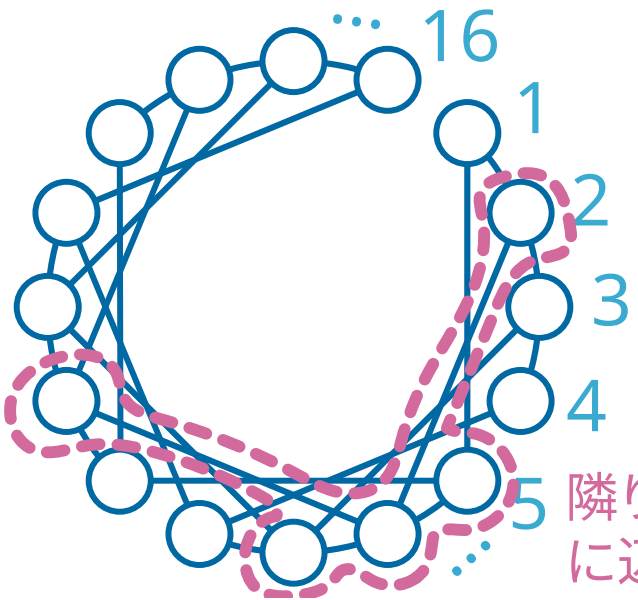
頂点と辺に
パラメータが
載っている

マルコフ確率場でエネルギーを表現

4×4の白黒画像



マルコフ確率場（ボルツマンマシン）

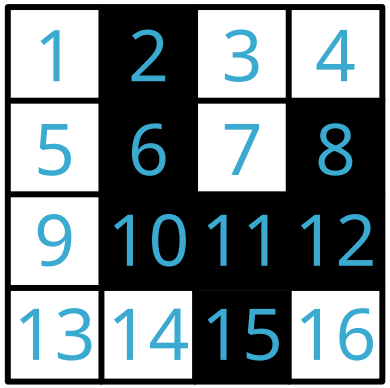


頂点と辺に
パラメータが
載っている

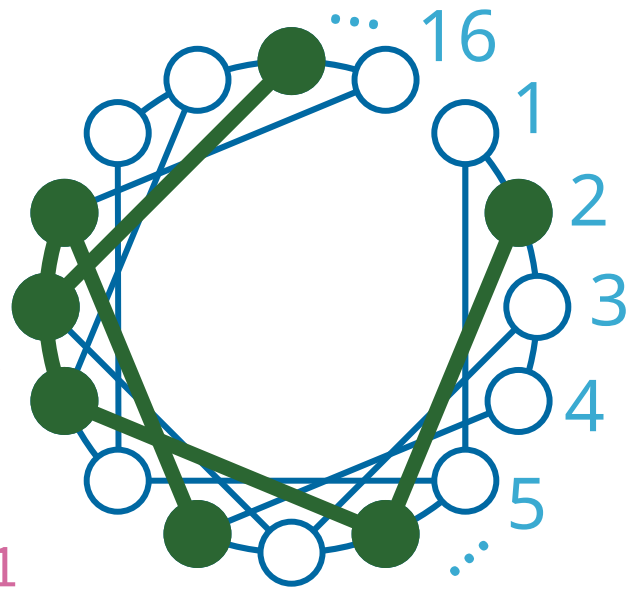
隣り合うピクセル間
に辺を張っている

マルコフ確率場でエネルギーを表現

4×4の白黒画像



マルコフ確率場（ボルツマンマシン）

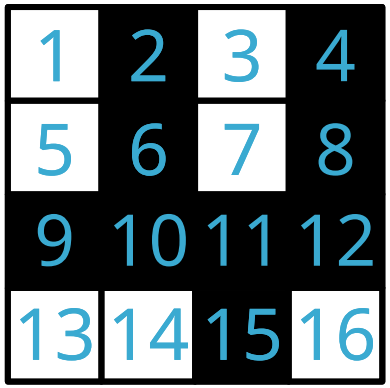


頂点と辺に
パラメータが
載っている

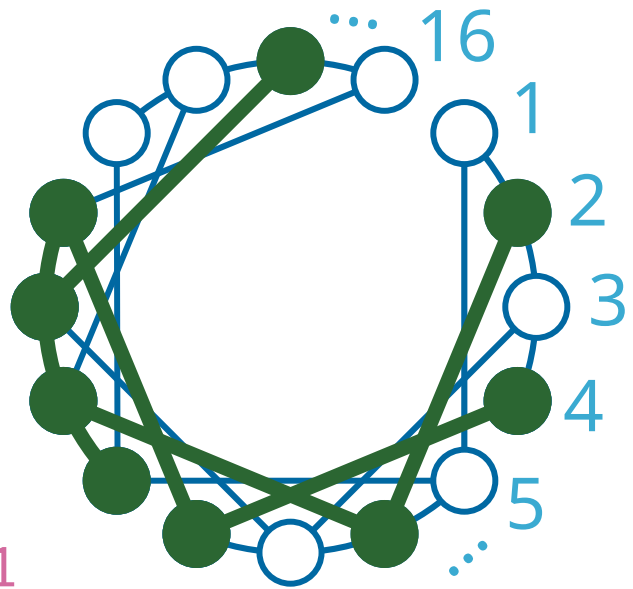
エネルギー
= 対応した箇所の
パラメータの総和×−1

マルコフ確率場によるエネルギー計算の例（1/2）

4×4の白黒画像



マルコフ確率場（ボルツマンマシン）

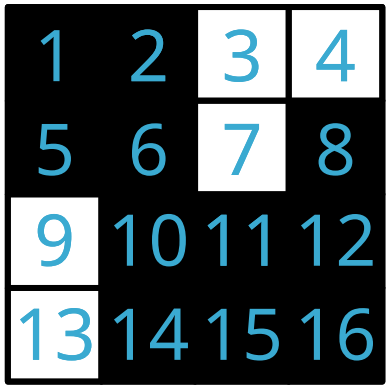


頂点と辺に
パラメータが
載っている

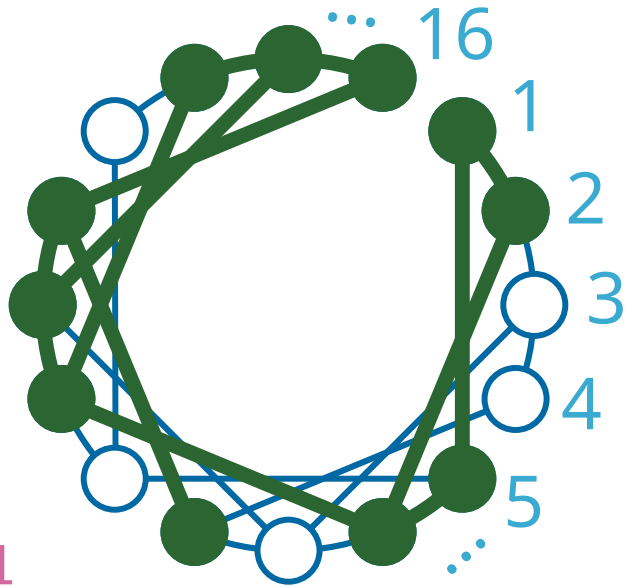
エネルギー
= 対応した箇所の
パラメータの総和×−1

マルコフ確率場によるエネルギー計算の例（2/2）

4×4の白黒画像



マルコフ確率場（ボルツマンマシン）



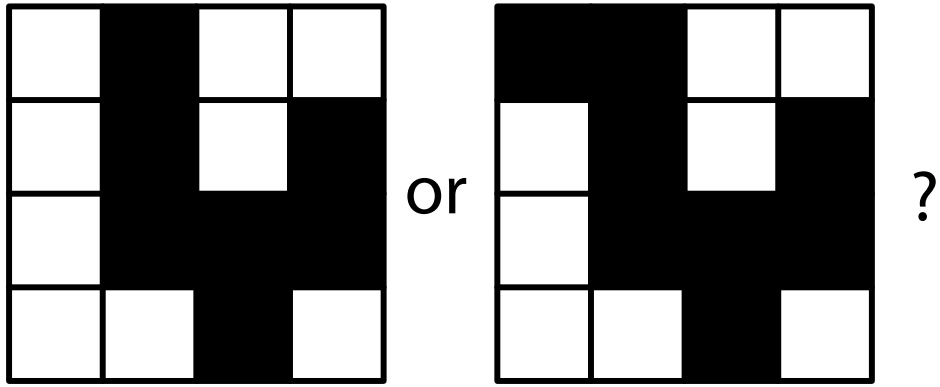
頂点と辺に
パラメータが
載っている

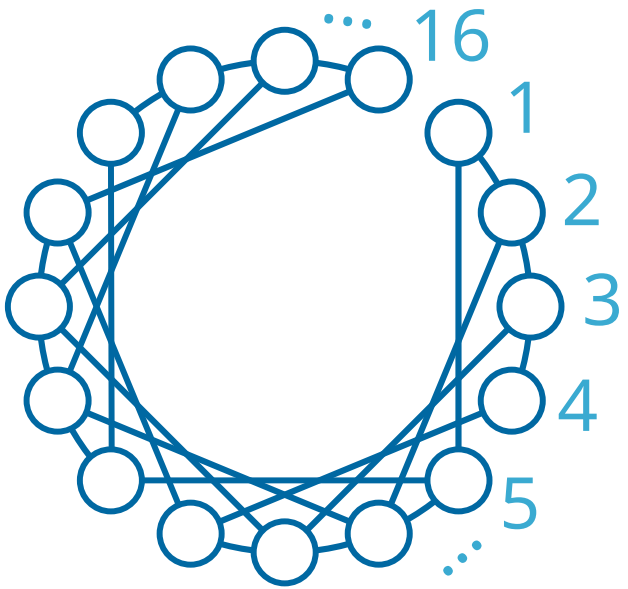
エネルギー
= 対応した箇所の
パラメータの総和×−1

ギブスサンプリング

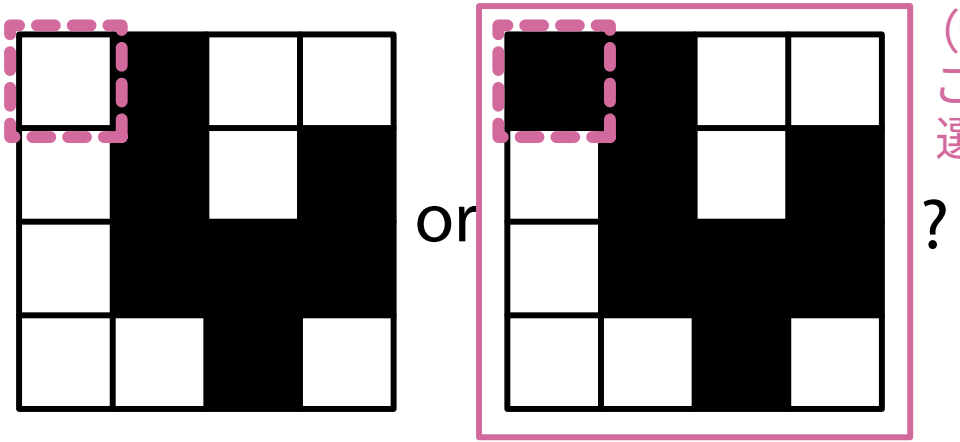
- マルコフ確率場は，パターン空間の確率分布を表現している
  - しかし，確率の値を取り出して見てみることはできない
  - 分かるのはエネルギーの値だけ
- エネルギーをうまく使えば，パターンの**サンプリング**ならできる！
  - 基本戦略：パターンの値を一つずつ更新していく
  - 各更新の際の（条件付き）確率であれば計算できる！
    - 正規化定数が分母と分子に出てくるので消える
- この手法は**ギブスサンプリング**と呼ばれる
  - MCMC（マルコフ連鎖モンテカルロ）法の一種

ギブスサンプリングの例（1/6）

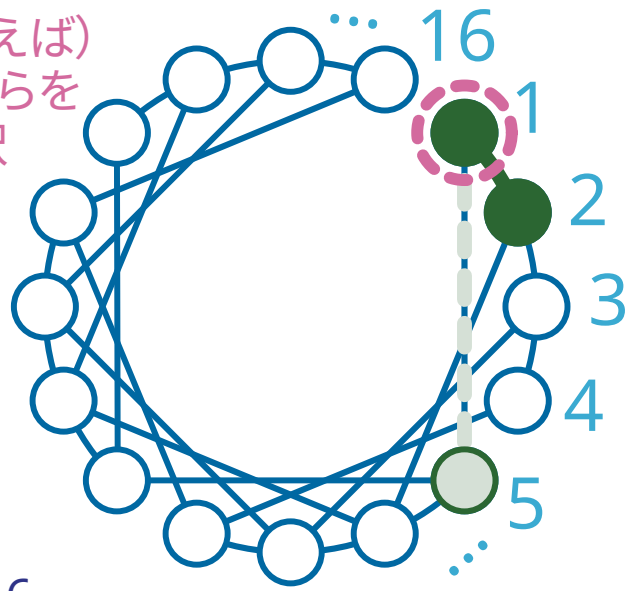




ギブスサンプリングの例（1/6）

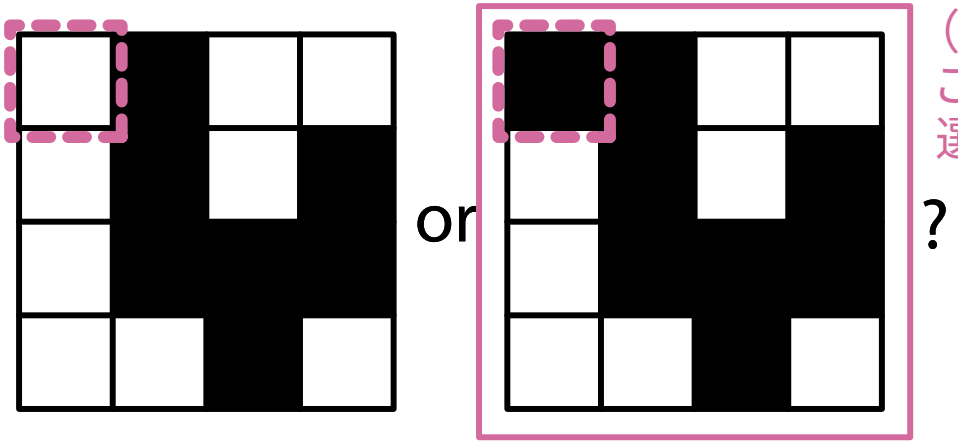


（例えば）
こちらを
選択



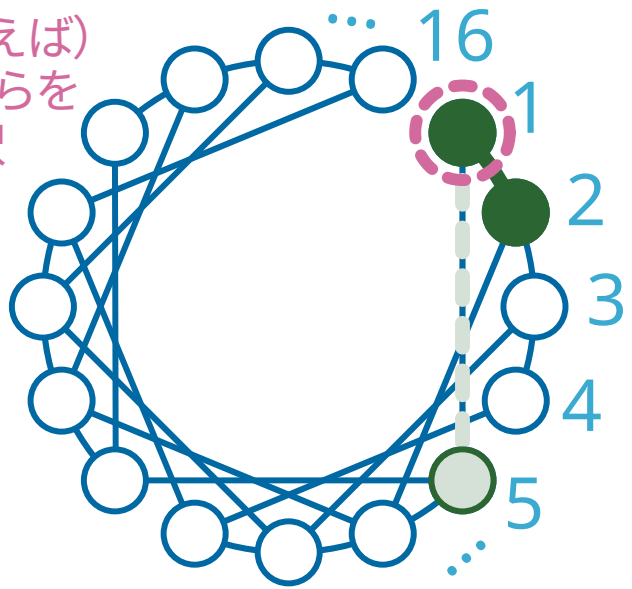
ポイント：
右か左の画像のどちらかを選ぶ，という状況なら，
例えば，右になる確率0.4，左になる確率1–0.4 = 0.6
のように，マルコフ確率場から求まるエネルギーを使って確率が計算できる

ギブスサンプリングの例（2/6）

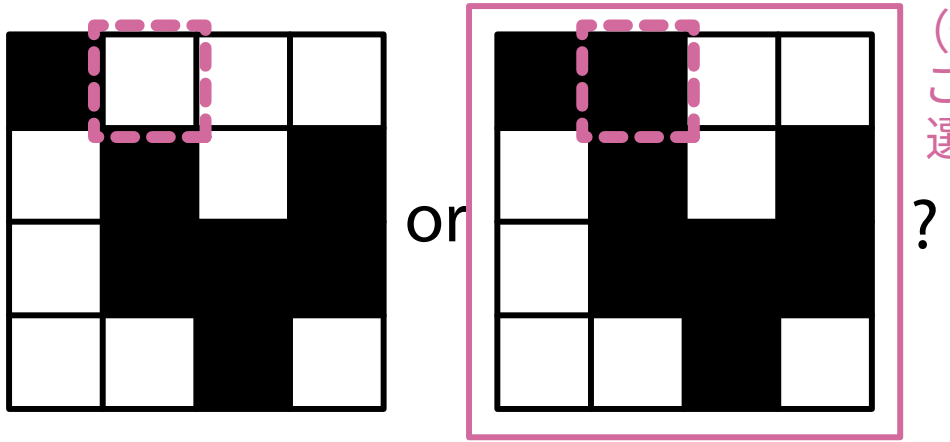


0100010101110010
1100010101110010

（例えば）
こちらを
選択



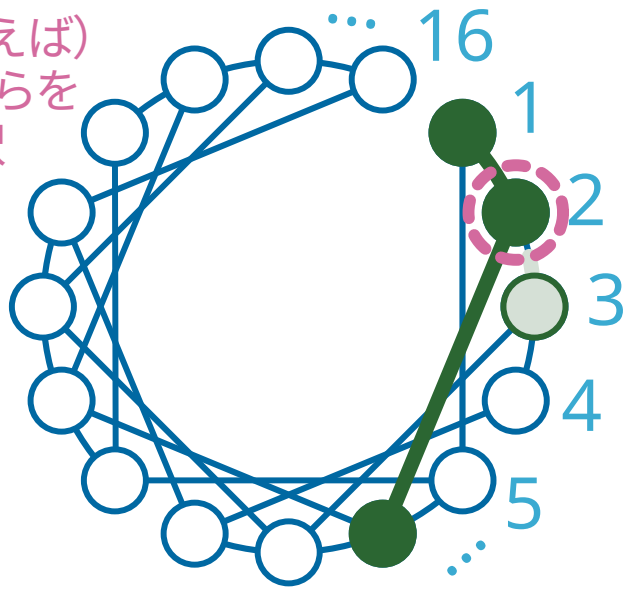
ギブスサンプリングの例（3/6）



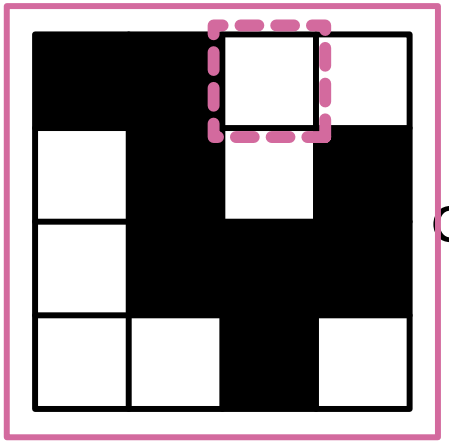
1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0

1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 0

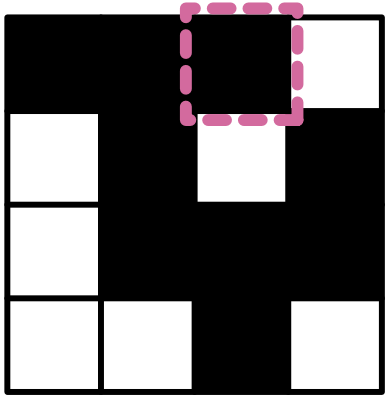
（例えば）
こちらを
選択



ギブスサンプリングの例（4/6）



or

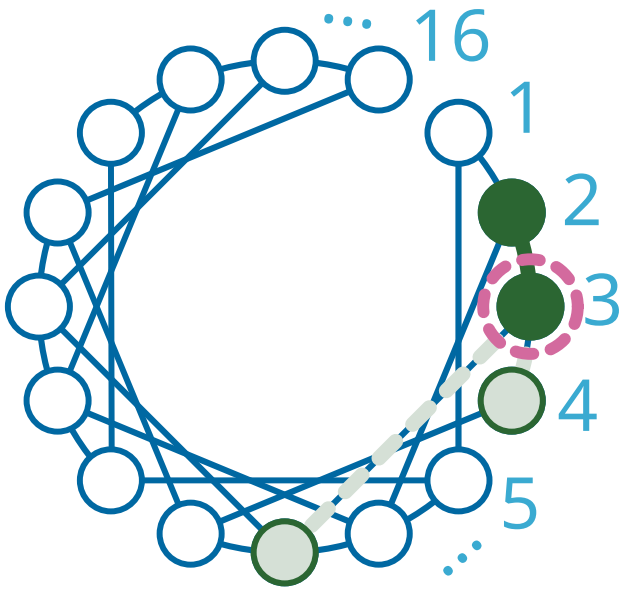


?

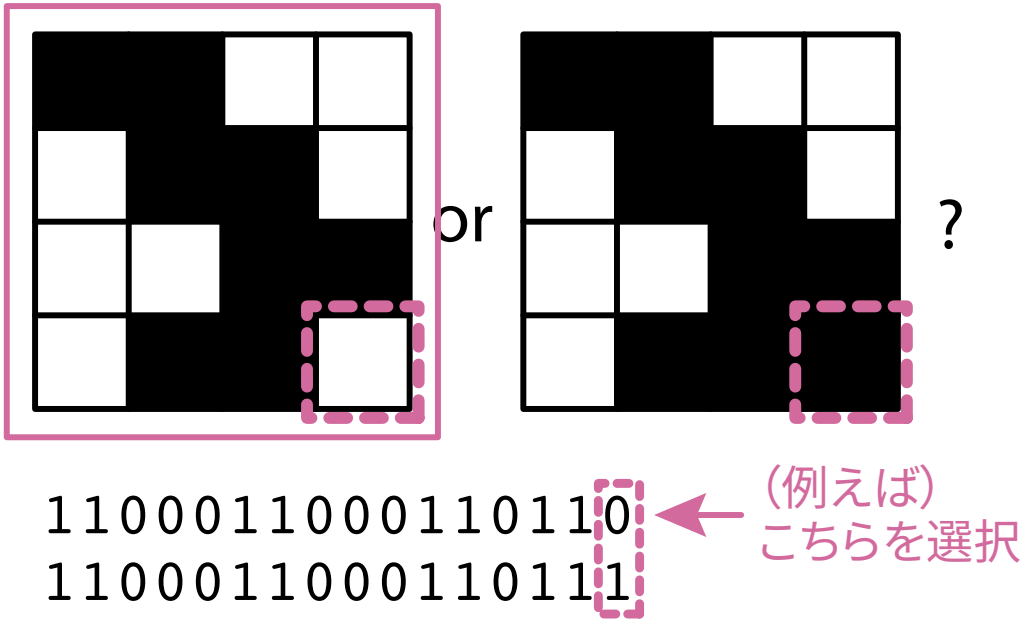
1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 0

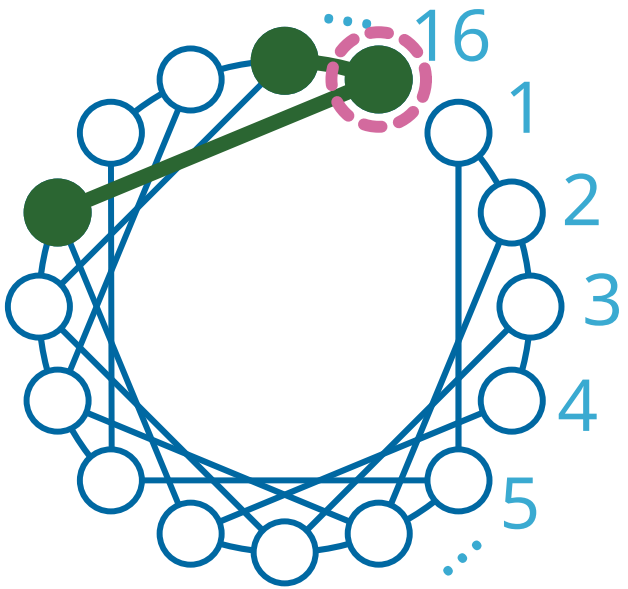
1 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0

（例えば）
こちらを選択

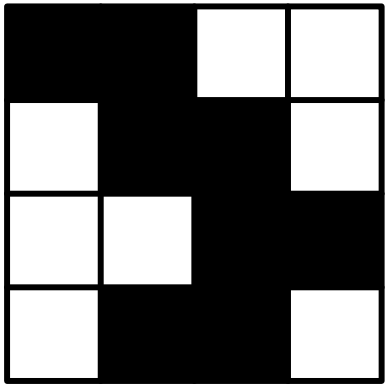


ギブスサンプリングの例（5/6）





ギブスサンプリングの例（6/6）



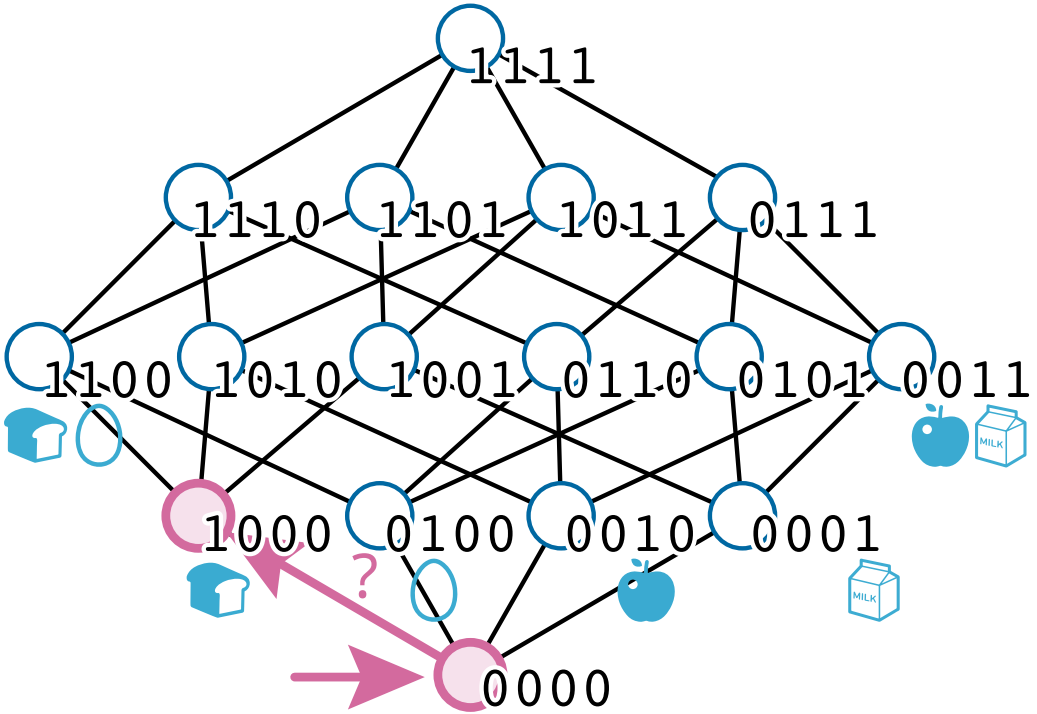
1100011000110110

これを出力する

- 以上の操作を，何度も繰り返すことで，いくつもサンプルが生成できる
  （実際には，サンプルをいくつか捨ててから出力したりする）
- この例では，2^{16} = 65,536 パターンからのサンプリングを，
  最大 16 + ${}_{16}C_2$ = 136 個のパラメータの組合せで実現

パターン空間でのギブスサンプリングの様子（1/5）

0000 or 1000 ?

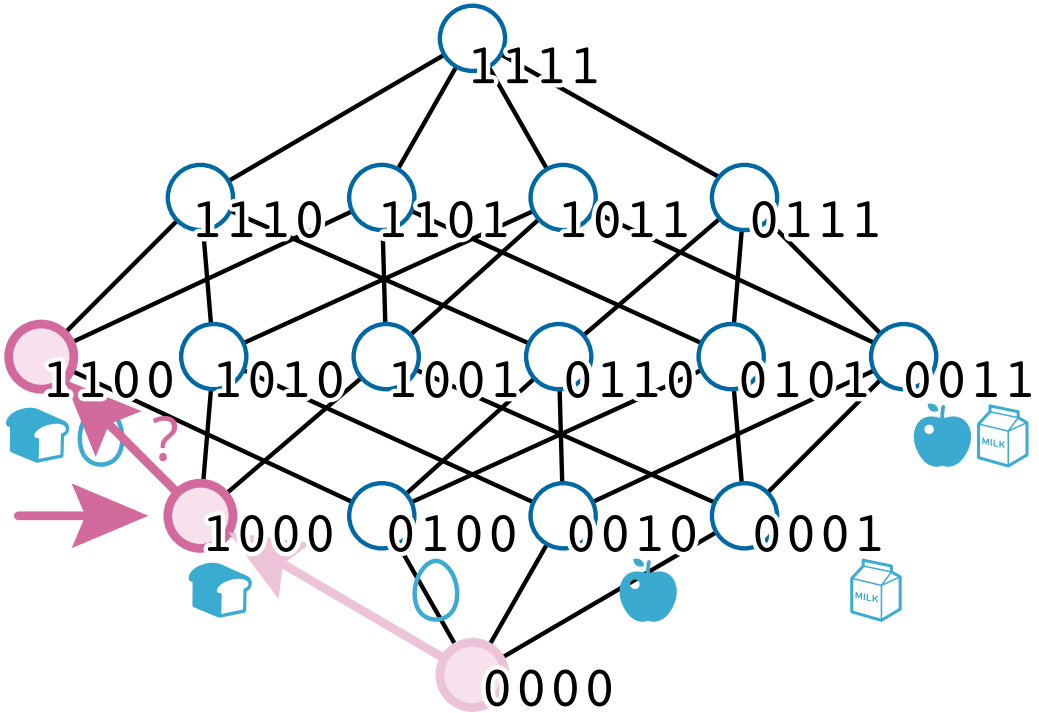


パターン空間でのギブスサンプリングの様子（2/5）

0000 or 1000 ?

↓

1000 or 1100 ?



パターン空間でのギブスサンプリングの様子（3/5）

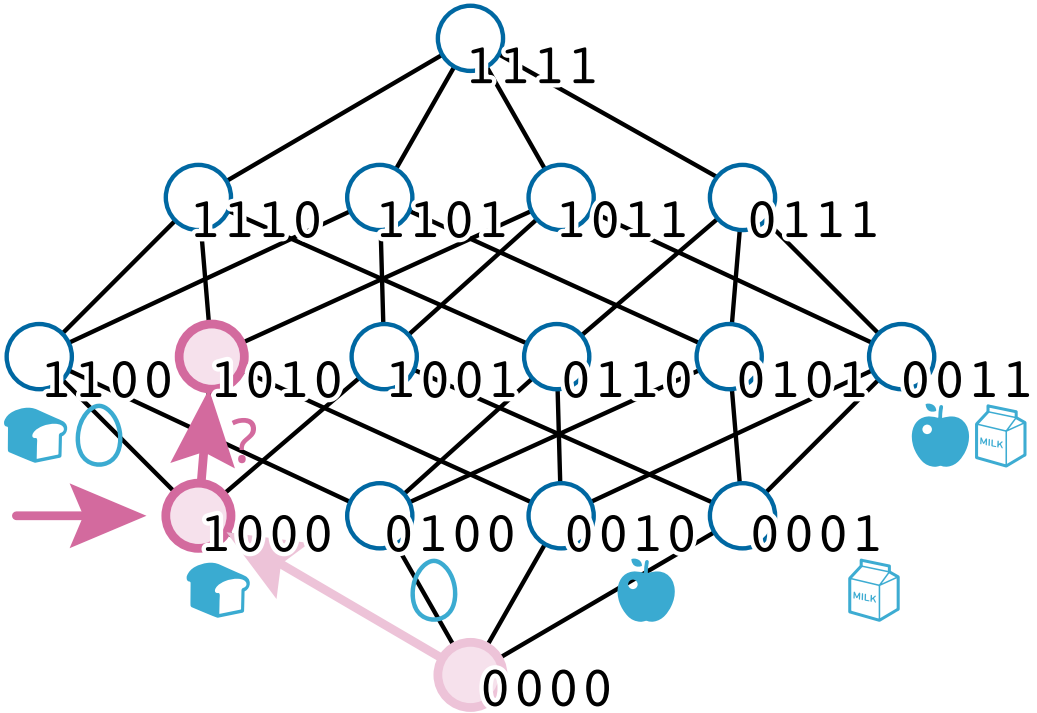
0000 or 1000 ?

↓

1000 or 1100 ?

↓

1000 or 1010 ?



パターン空間でのギブスサンプリングの様子（4/5）

0000 or 1000 ?

↓

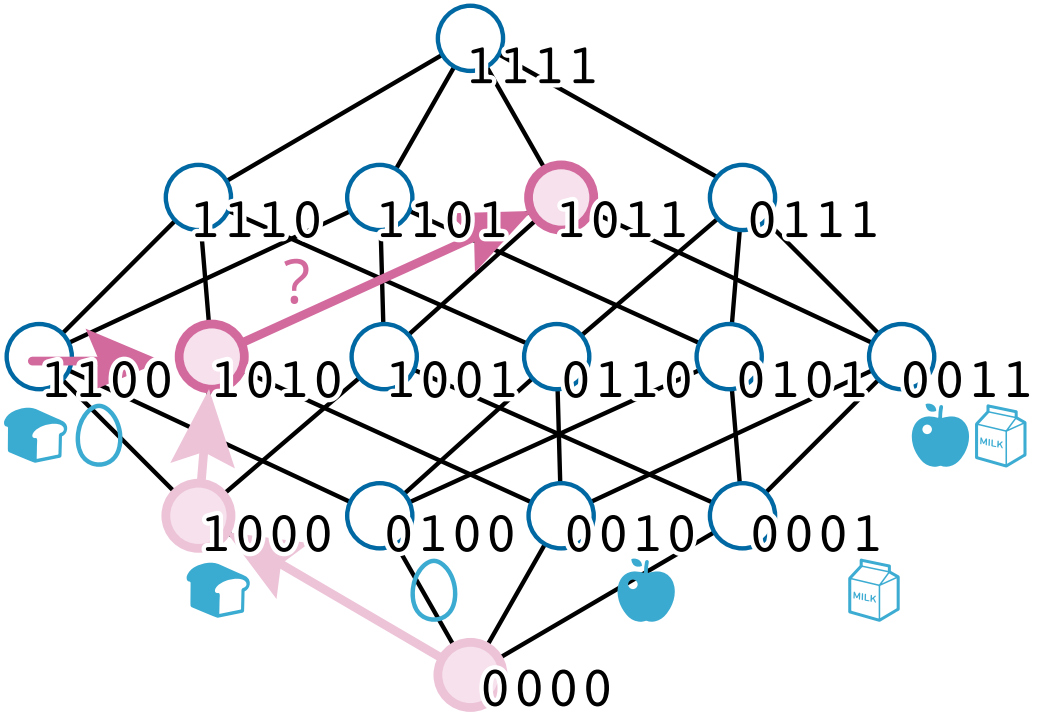
1000 or 1100 ?

↓

1000 or 1010 ?

↓

1010 or 1011 ?



パターン空間でのギブスサンプリングの様子（5/5）

0000 or 1000 ?

↓

1000 or 1100 ?

↓

1000 or 1010 ?

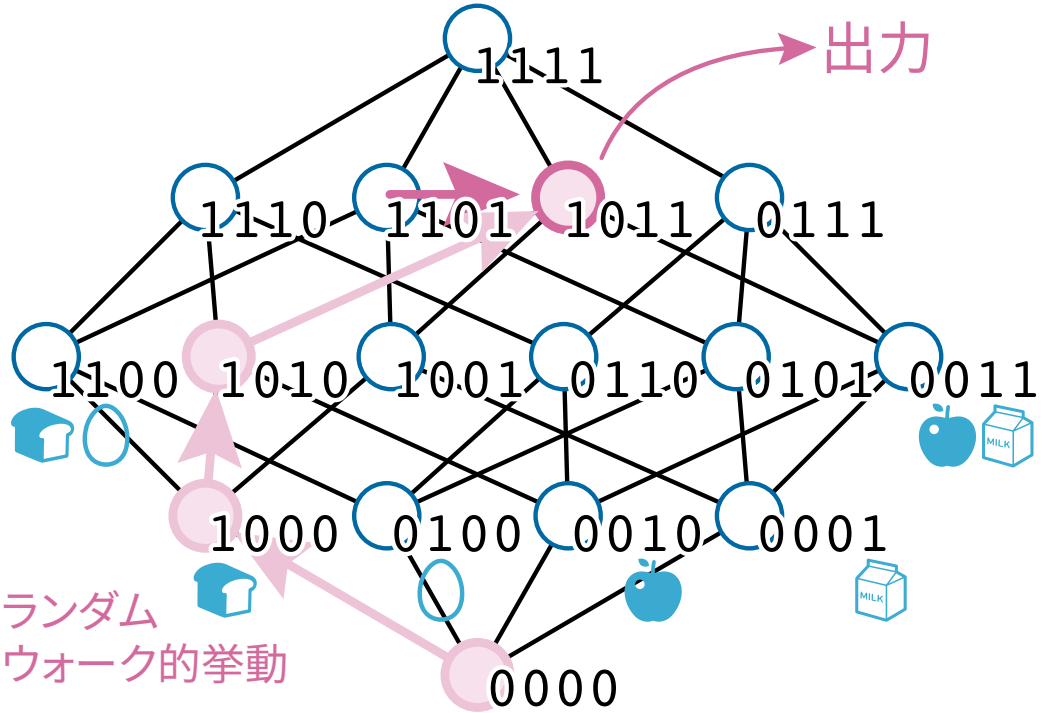
↓

1010 or 1011 ?

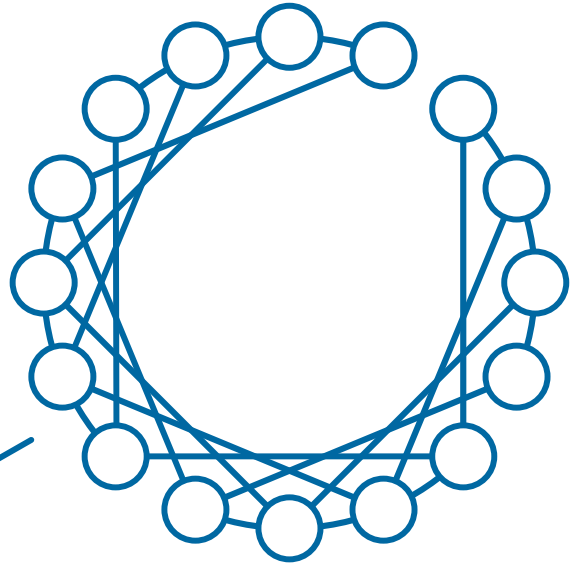
↓

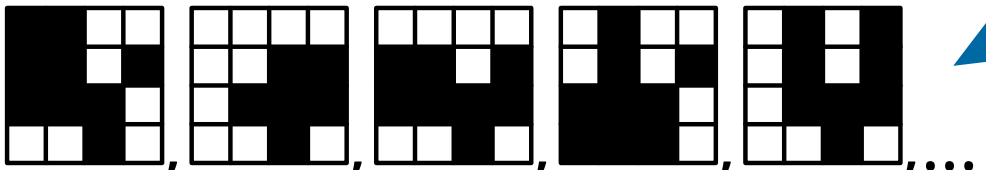
1011

出力



マルコフ確率場の学習

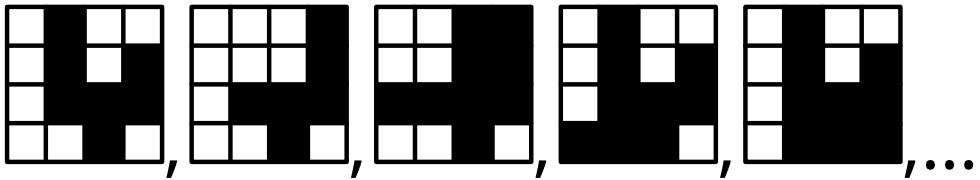




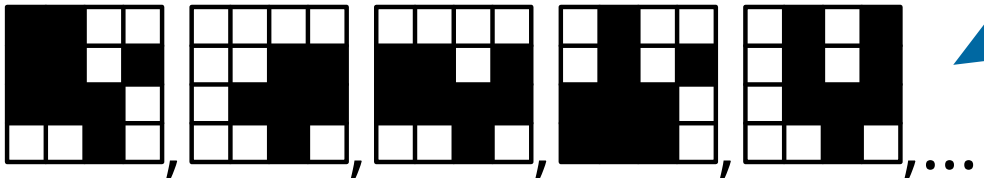
サンプリングで得られたパターン

勾配法でパラメータを調整する

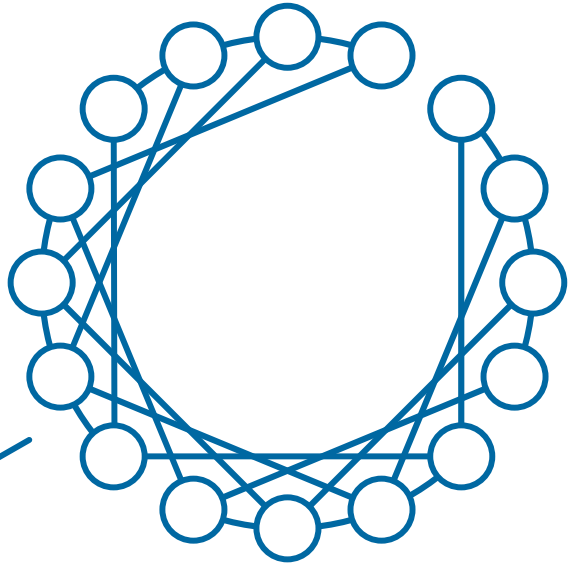
データとして持ってるパターン



頻度が等しくなるように
パラメータを少しずつ
調整する（勾配法）



サンプリングで得られたパターン

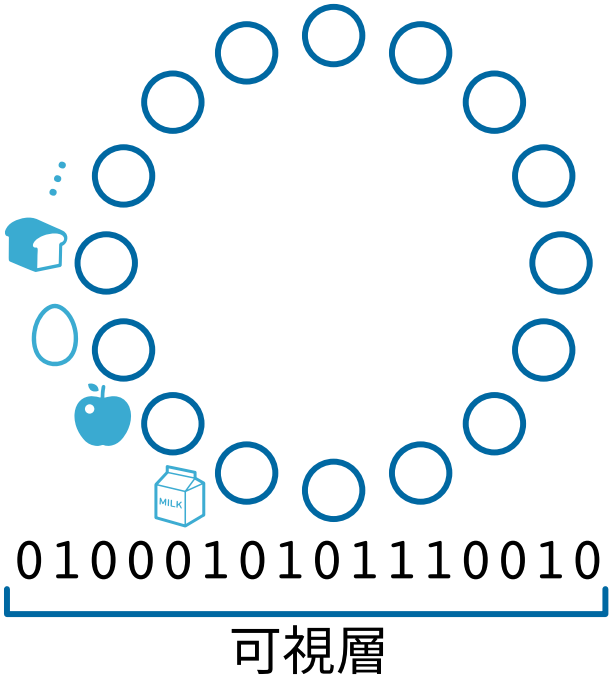


1)サンプリング, 2)頻度の比較,
3)パラメータ調整, を繰り返す

深層学習へ

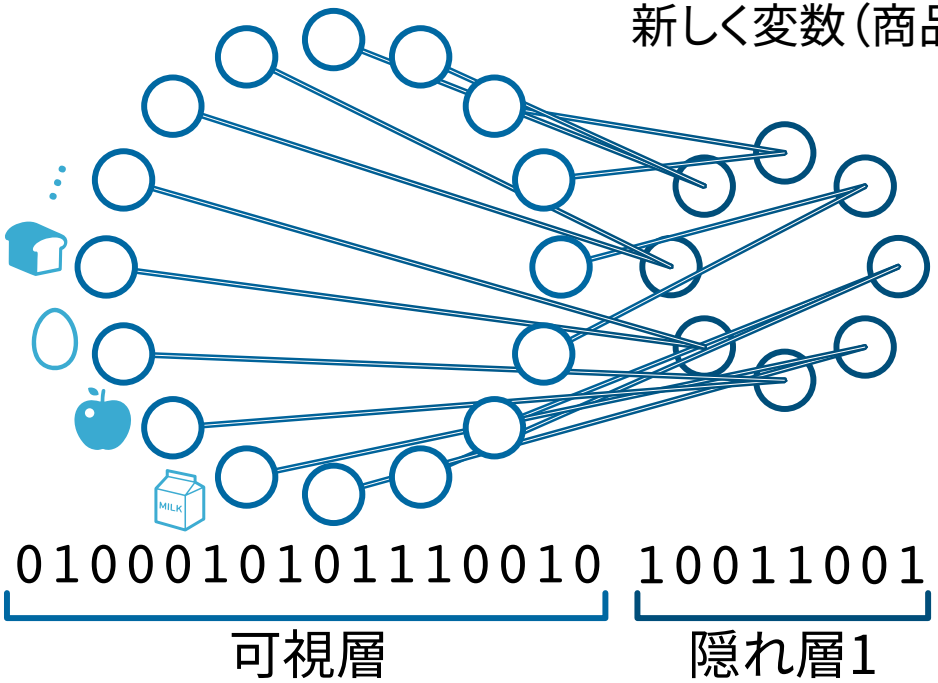
- これまで，いかに**必要とする（見ないといけない）パターンの数を減らすか**，ということに注力してきた
  - マイニングでは，下から順にチェックすることで，全探索を避けた
  - サンプリングでは，2つの比較を順に繰り返す，という戦略で全空間を見ずにそこからのサンプリングを実現した
- 深層学習的アプローチでは，これとは逆に，**パターンの数をさらに増やして**モデルの表現能力を向上させ，より精度の高い推定を目指す
  - 具体的には，隠れ変数を導入する
  - 購買データでは，スーパーにない商品を追加する
  - 画像データでは，新たに画素を追加する

可視変数からなるマルコフ確率場

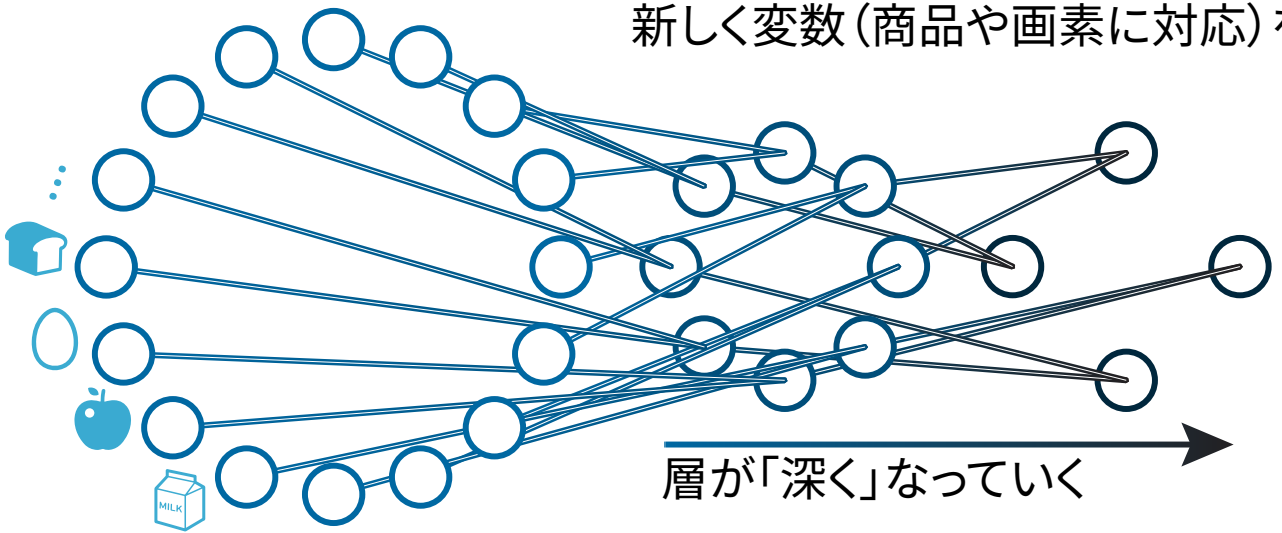


隠れ層を追加

新しく変数（商品や画素に対応）を追加する

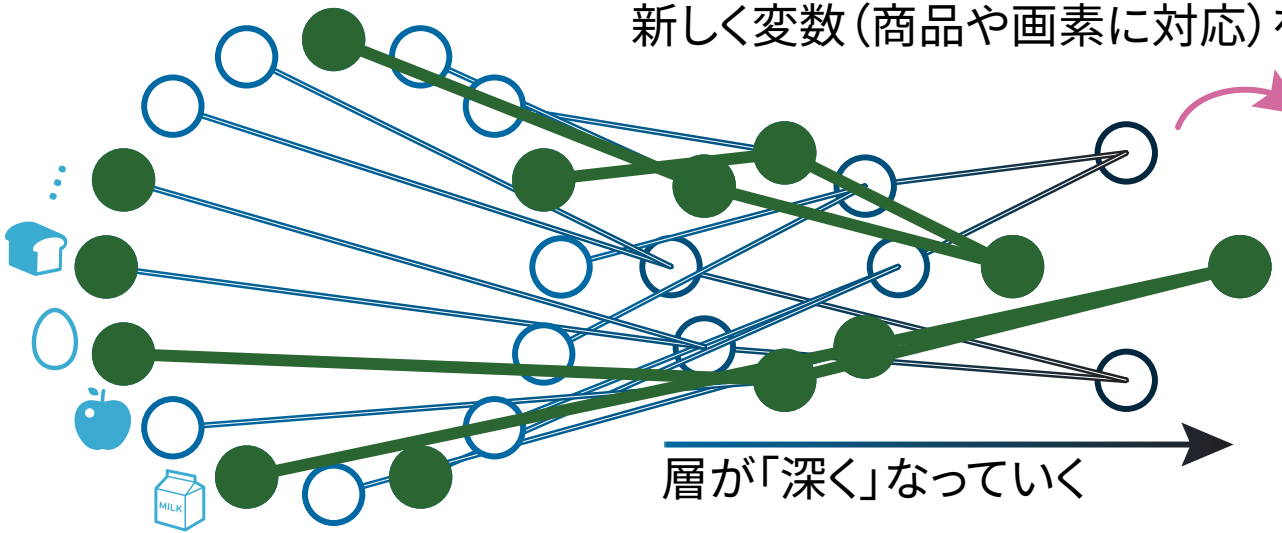


隠れ層をさらに追加（「深層」モデル）

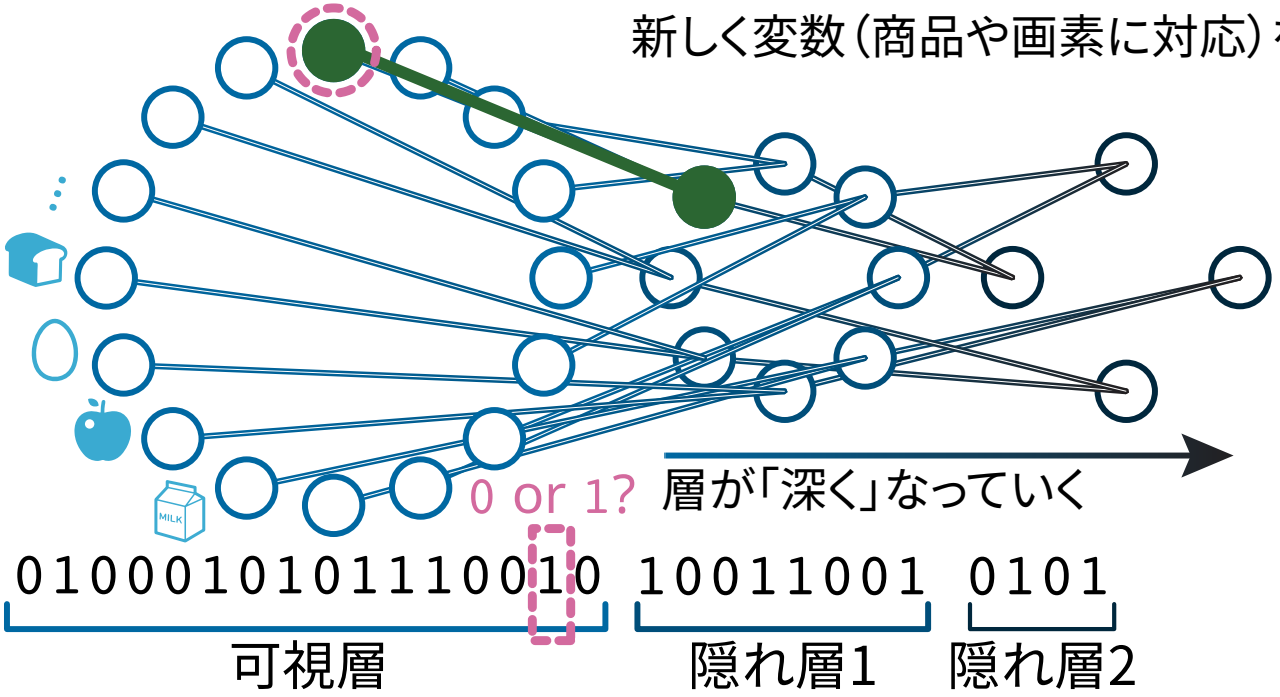


各パターンが
2^{16} = 65,535
パターンに増えた
→ 表現力の向上

エネルギーの計算は同じ手続きでOK

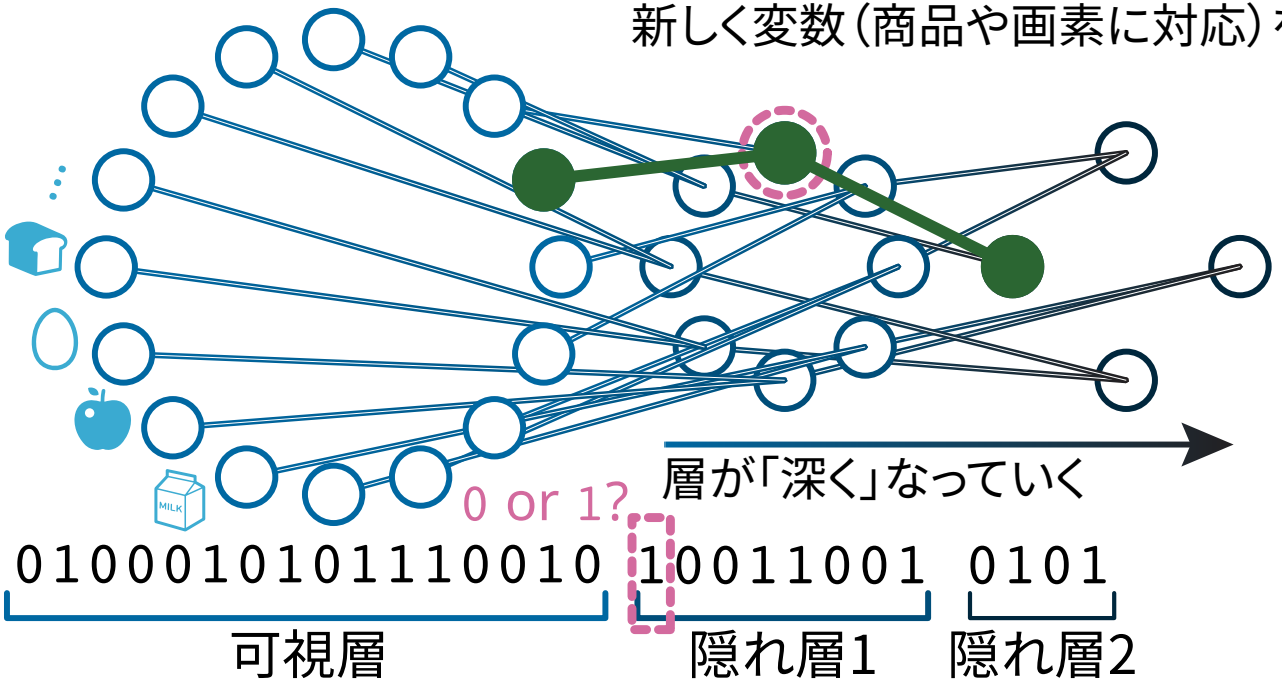


ギブスサンプリングも同じ手続きでOK



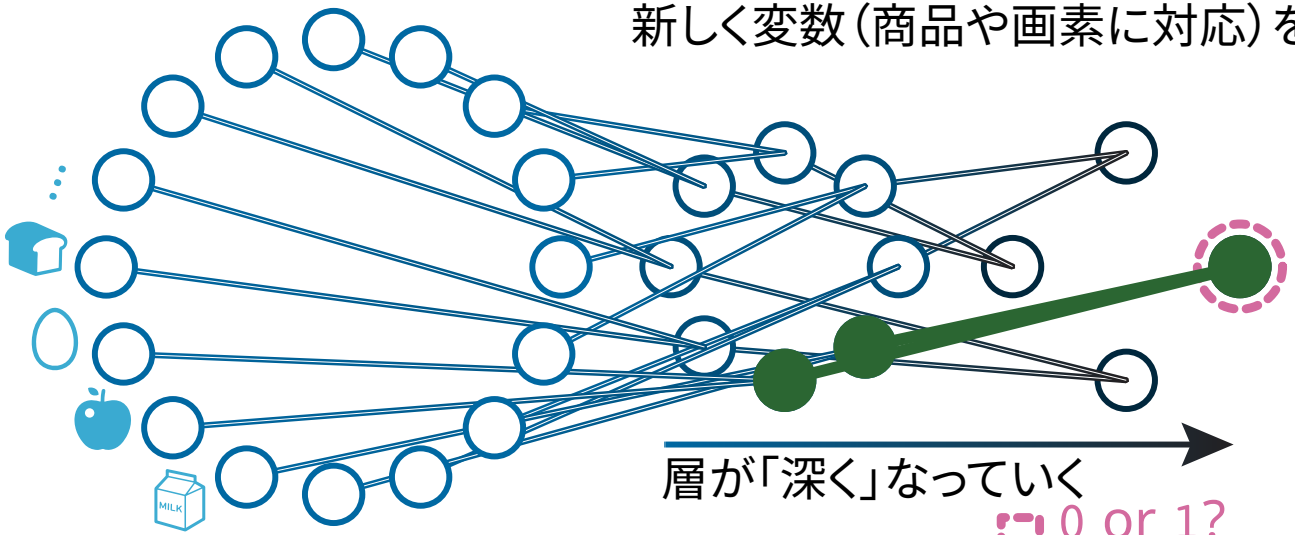
各パターンが
2^{16} = 65,535
パターンに増えた
→ 表現力の向上

隠れ変数についても同様にサンプリングできる



各パターンが
2^{16} = 65,535
パターンに増えた
→ 表現力の向上

隠れ変数についても同様にサンプリングできる



各パターンが
2^{16} = 65,535
パターンに増えた
→ 表現力の向上

まとめ

- パターン解析の難しさ：数が多すぎて全部見れない
- パターンが持つ構造を利用すれば，一部分を見るだけでも色々できる
  - マイニング，サンプリング，確率分布の学習
- パターンの数をわざと増やすこともある（深層学習）

- 私たちの国立情報学研究所での研究：
  - 統計的な検定を使ったパターンマイニングで信頼性向上
  - エネルギーベースモデルの表現方法を変えることで，適用範囲を拡大
  - 変数を無限に増やしていったときの性能を理論的に解析
  - パターンの結合による化合物の生成

参考文献（さらに興味のある方へ）

- パターンマイニングについては：
  - Zaki, M. J., Meira Jr., W.: **Data Mining and Analysis: Fundamental Concepts and Algorithms**, Cambridge Univ. Press [Link]
  - 杉山, **統計的有意性を担保するパターンマイニング技術**, オペレーションズ・リサーチ誌, 2017 [Link]
- マルコフ確率場での学習やギブスサンプリングについては：
  - 人工知能学会監修，**深層学習**，近代科学社
  - Wainwright, M. J., Jordan, M. I.: **Graphical Models, Exponential Families, and Variational Inference**, Foundations and Trends in Machine Learning, 2008 [Link]