統計的機械学習理論とボルツマン機械学習

山形大学　大学院理工学研究科
安田 宗樹

統計的機械学習

統計的機械学習は何をするのか？

決定論的ニューラルネットワークによる機械学習

入出力関係をデータから学習する

入力と出力の決定論的な（入力から出力が一意に決まる）

関数関係を探し出すのが目的

統計的機械学習

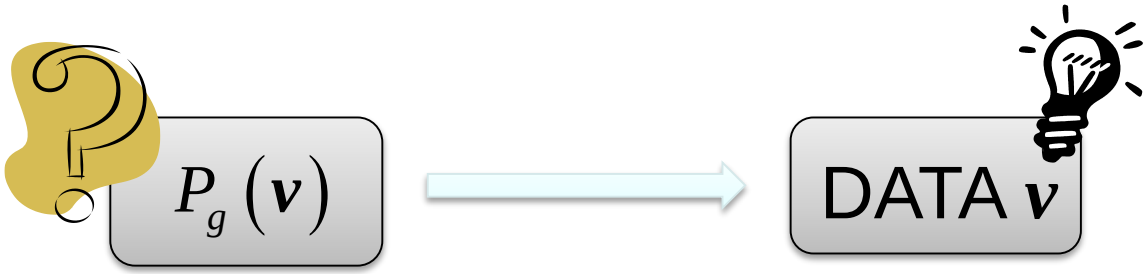
データの確率的な生成規則を学習する

データパターンを生成する背景となっている

確率分布を探し出すのが目的

なぜ確率を使うのか？

データパターンはとある確率規則に従って発生しているとする



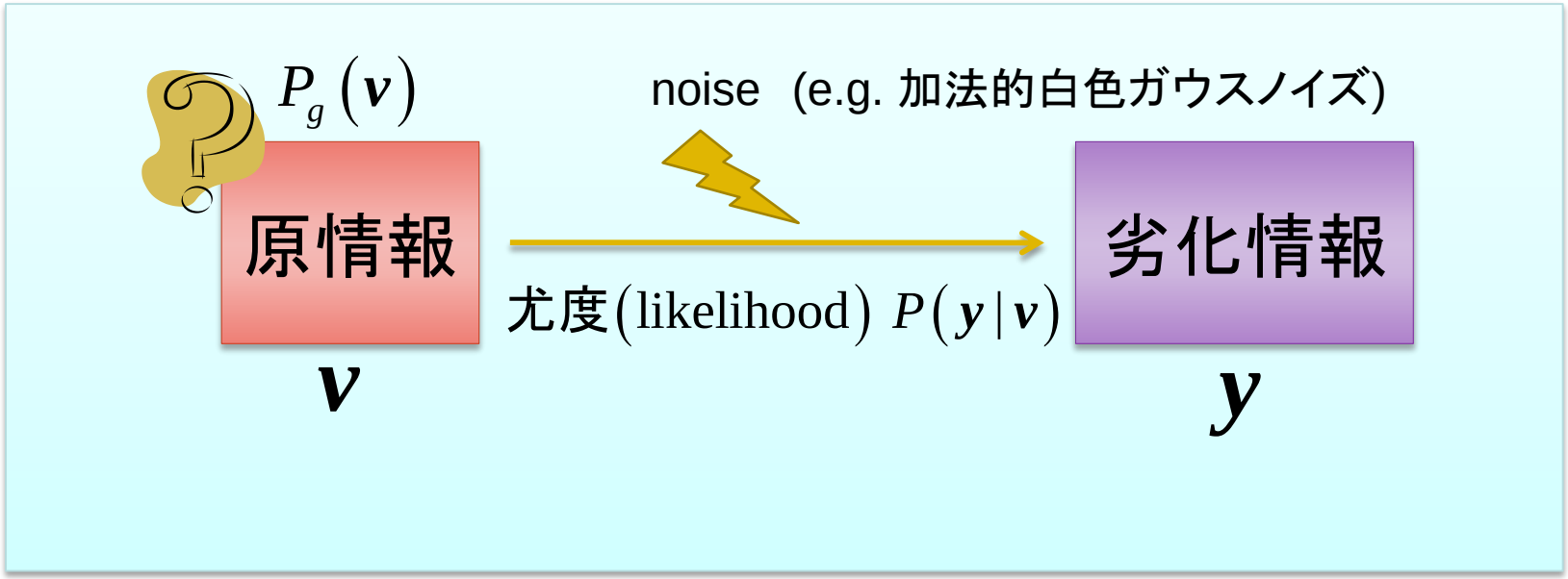
image, text, gene, etc

何故確率が必要？

- 観測ノイズに関わる不可避の**不確実性**
- 知識不足による未知の因果に対する**不確実性**

不確実な現象を表現する数学 ＝ **確率理論**

ベイズ的確率推論

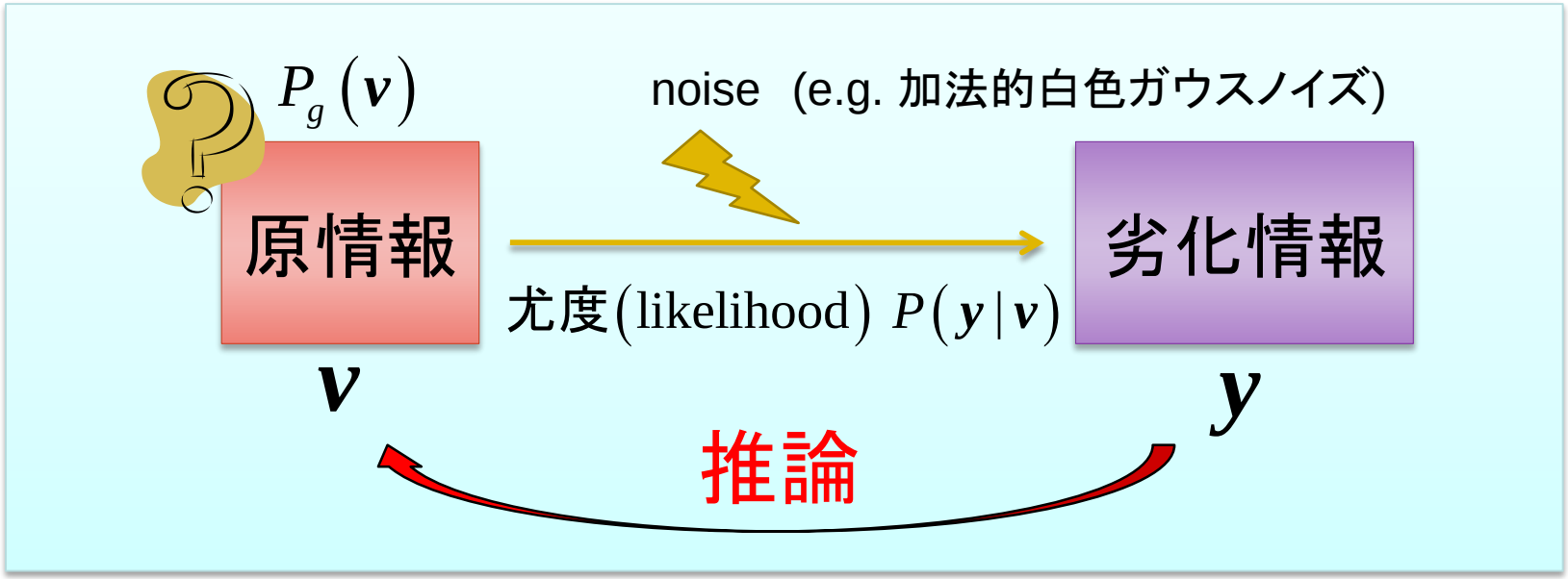


原情報 $\boldsymbol{v}$ が通信路ノイズによって劣化し

劣化情報 $\boldsymbol{y}$ に変化した

我々は $\boldsymbol{y}$ しか知らない

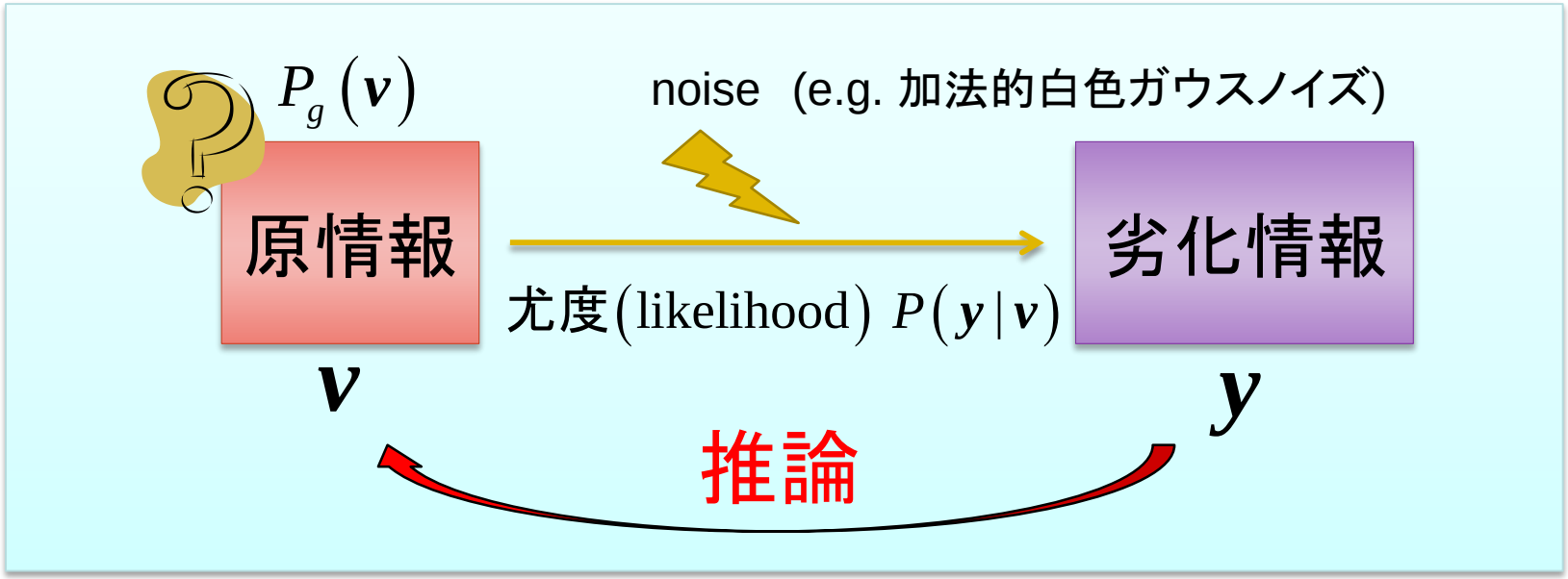
ベイズ的確率推論



我々は $\boldsymbol{y}$ しか知らない

$\boldsymbol{y}$ から $\boldsymbol{v}$ を推定したい！

事後確率を計算する



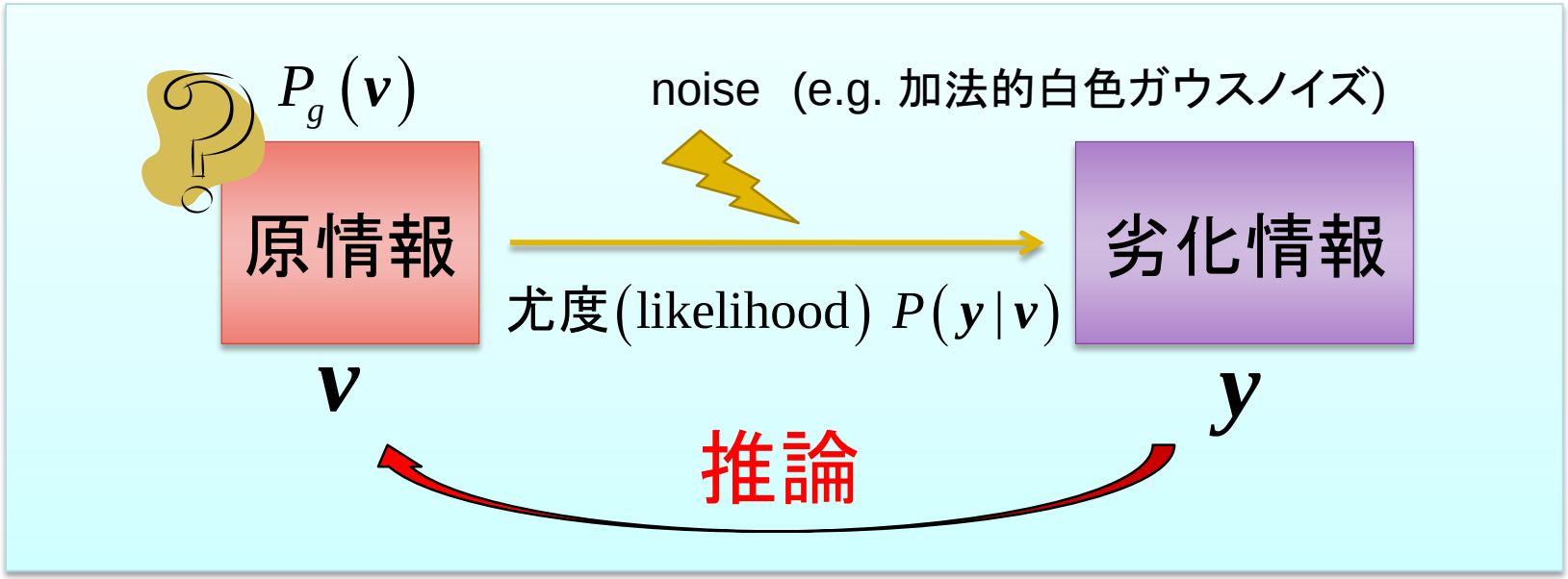
事後確率

ベイズの公式

$$P(\boldsymbol{v}|\boldsymbol{y}) = \frac{P(\boldsymbol{v},\boldsymbol{y})}{P(\boldsymbol{y})} = \frac{P(\boldsymbol{y}|\boldsymbol{v})P_g(\boldsymbol{v})}{P(\boldsymbol{y})}$$

$\boldsymbol{y}$ を知った下での $\boldsymbol{v}$ の確率

事後確率(用語の整理)

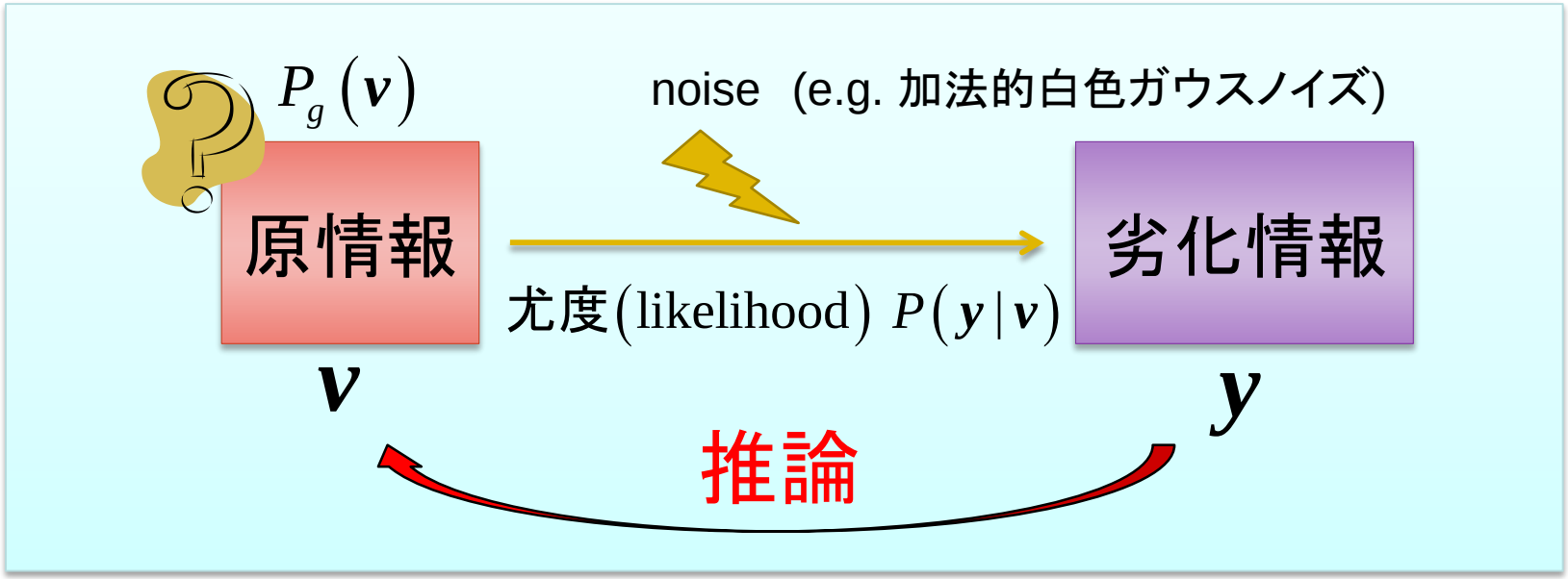


時間的には。。。

$$(\text{事後確率}) = P(\boldsymbol{v}|\boldsymbol{y}) = P(\text{原因}|\text{結果})$$

結果から原因を探るための定式化

最大事後確率(MAP)推定



$$\boldsymbol{v}^* = \arg\max_{\boldsymbol{v}} P(\boldsymbol{v}|\boldsymbol{y}) = \arg\max_{\boldsymbol{v}} P(\boldsymbol{y}|\boldsymbol{v}) P_g(\boldsymbol{v})$$

$\boldsymbol{y}$ から見て確率的にもっともあり得そうな $\boldsymbol{v}$ が原情報だろうと考える

ベイズ的確率推定の基礎

事後確率計算に必要なもの

$$P(\boldsymbol{v}\,|\,\boldsymbol{y}) \propto P(\boldsymbol{y}\,|\,\boldsymbol{v})\,P_g(\boldsymbol{v})$$

$$(\text{事後確率}) \propto (\text{尤度}) \times (\text{事前確率})$$

- 尤度 ・・・ 原情報がどのような過程で変形を受けるかの確率
- 事前確率 ・・・ 原情報がどのような確率規則で生成されるかの確率

事前確率 $P_g(\boldsymbol{v})$ ： 真の情報の生成確率（情報生成の背後にある確率規則）

$$(\text{事前確率}) = (\text{原情報生成の背後規則})$$

$$= (\text{生成モデル（generative model）})$$

事前確率(生成モデル)

事前確率モデル　＝　データの生成モデル(メカニズム)

- ✓ 適切に設計しなければ確率推論そのものがダメになる
- ✓ 一般的にはかなり難しい問題　(メカニズム理解からのモデル構築)
  (ex. 遺伝子発現のメカニズム，自然画像の生成メカニズム)

データ駆動型のモデル構築

- ✓ 生成モデルの詳細は分からないが，そこから発生したデータはもっている
  (例えば画像などは google で検索すればいくらでも手に入る)
- ✓ それらのデータは生成モデルに関する情報をもっているはず
- ✓ その情報を活かして生成モデルを探せないだろうか？

データ主体の生成モデル発掘技術　→　統計的機械学習

観測データのセット D

N given (n dimensional) data points:

$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ dimensional data point}}$$

$$\mathrm{d}_i^{(\mu)} \in \{0,1\}$$

$$\underbrace{0100101\cdots001}_{n} \quad \mathbf{d}^{(1)}$$

$$\underbrace{1011011\cdots111}_{n} \quad \mathbf{d}^{(2)}$$

$$\vdots$$

$$\underbrace{0010011\cdots101}_{n} \quad \mathbf{d}^{(N)}$$

それぞれのデータはある生成モデルから独立に生成されているとする

$$P_g(\boldsymbol{v}) \xrightarrow[\text{generate (i.i.d.)}]{} \mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}$$

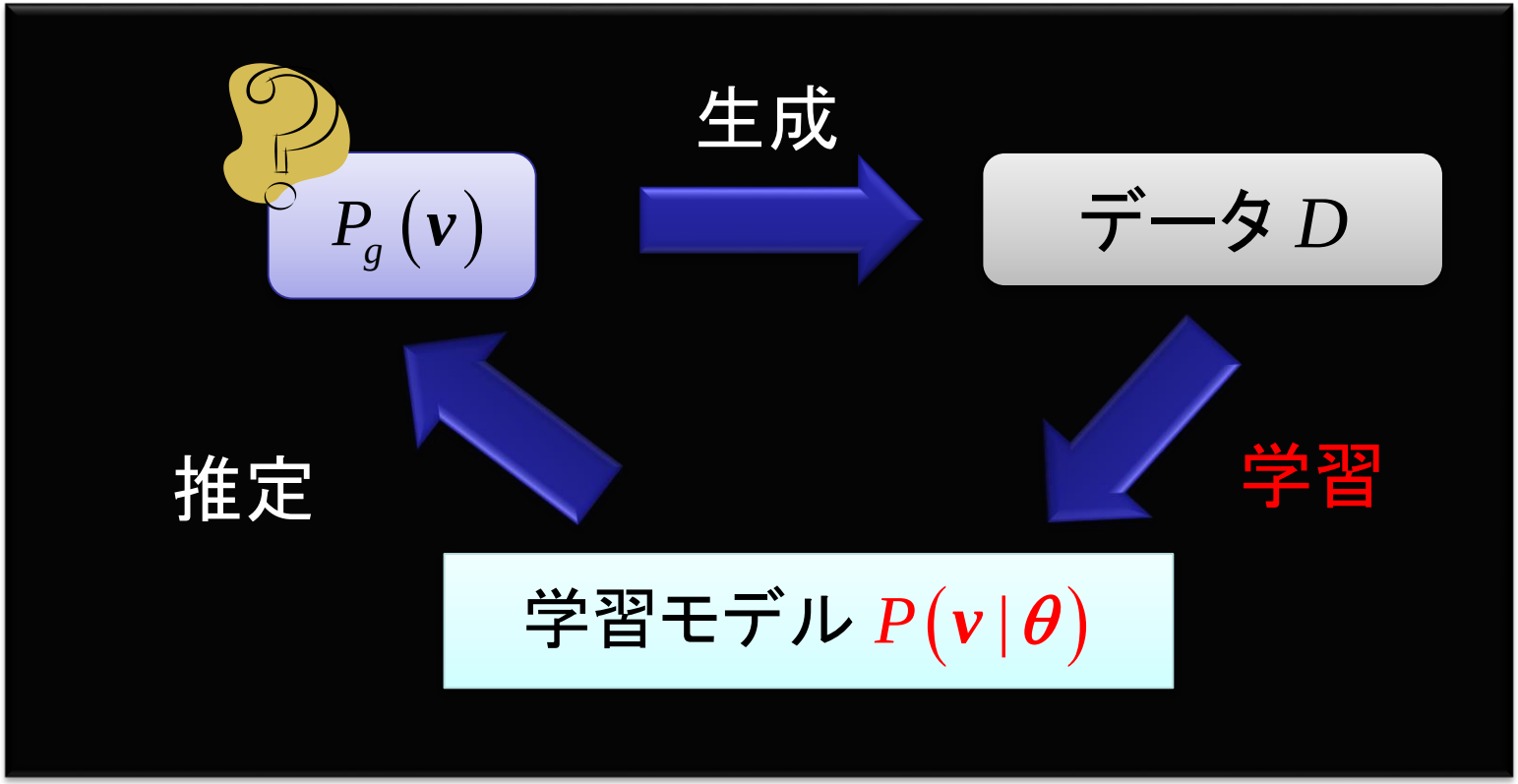
$$\boldsymbol{v} = \{v_1, v_2, \ldots, v_n\}$$

生成モデルは データ生成の確率規則を記述する

$\mathbf{d}^{(\mu)}$ は確率 p_μ で生成される $\Leftrightarrow P_g\left(\mathbf{d}^{(\mu)}\right) = p_\mu$

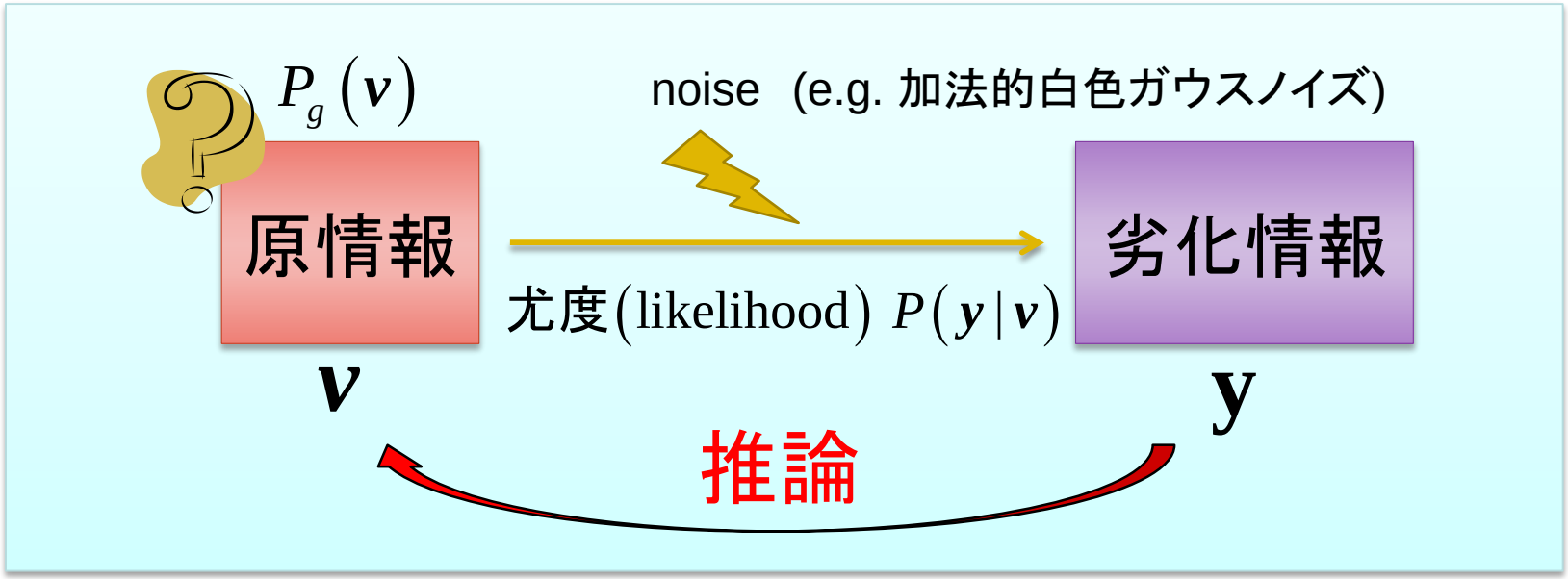
統計的機械学習の枠組み

$P(\boldsymbol{v}|\boldsymbol{\theta})$ ← パラメータ θ をもったある学習モデルを仮定する



- データ D を利用して仮定した学習モデルを学習する
- 学習により（未知の）事前確率に学習モデルを近づける

統計的機械学習とベイズ的確率推論



$$\boldsymbol{v}^* = \arg\max_{\boldsymbol{v}} P(\boldsymbol{y}|\boldsymbol{v})\, P_g(\boldsymbol{v}) \approx \arg\max_{\boldsymbol{v}} P(\boldsymbol{y}|\boldsymbol{v})\, \underbrace{P(\boldsymbol{v}|\boldsymbol{\theta})}$$

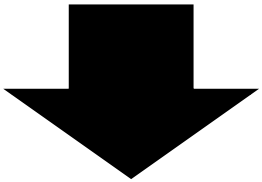
生成モデル　　学習モデル

学習モデルで真の生成モデルを近似して
原情報を推定する！

残された課題

生成モデルを近似する学習モデルを仮定する必要がある

どのようなモデルを仮定すればよいか？？



もちろんそれは課題によって個別に設計されるべき

だが

グラフィカルモデルは中でもわりと一般的に使える
可能な選択肢の一つ

グラフィカルモデル

データをどう考えるか？

（生成モデル）＝（データの確率的生成規則を記述する）

ふつう多次元（e.g. 画像ならピクセルの個数分の次元）

- データというものはふつう独立的ではなく，データ要素間に何らかの関連性（確率的相関構造）をもつ
  → 出現しやすいデータパターンや出現しにくいパターンが存在する
  （A家の夕食がカレーのときは何故かB家の夕食が煮物であることが多い etc）
- データを確率的に扱うだけでなく，そのような関連性も扱うことが重要

データ要素間の関連性が意味のある情報の基本

生成モデルは確率的相関構造をもつ

生成モデルはデータ要素間に関連性を持たせたものであるべき

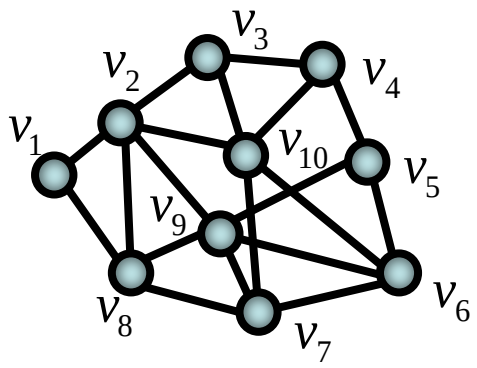
データ要素間の確率的関連性を明に取り入れるモデリング

- マルコフ確率場（Markov random field; MRF）
- ベイジアンネットワーク（Bayesian network; BN）

ともにグラフィカルモデルとよばれるモデル

グラフィカルモデル？

ノードとリンクから構成されるグラフを作る



各ノードがデータの各要素（つまり各確率変数）に対応する

リンクは確率変数間の（直接の）関連性を表わす

関連性を直観的に
導入することができる

作ったグラフ構造に則って確率モデルを作成する

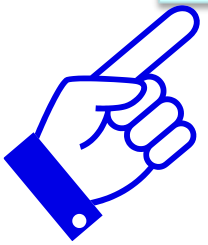
MRF と BN

MRF

無向グラフ上の確率モデル

BN

有向（非循環）グラフ上の確率モデル



MRF と BN の関係

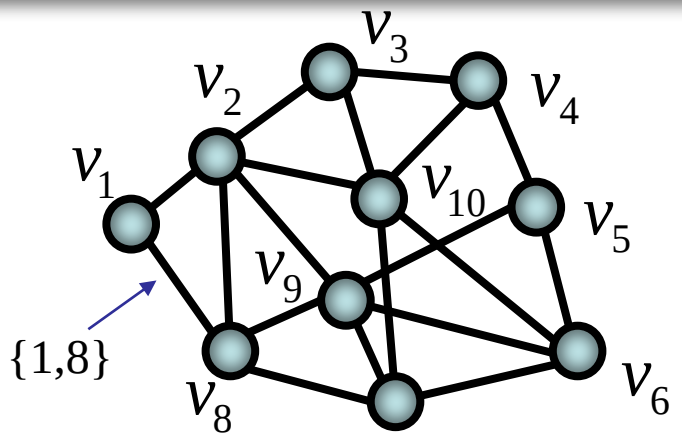
BN から MRF への変換 ・・・ 簡単

MRF から BN への変換 ・・・ NP-hard

マルコフ確率場の導入

$G(V,E)$ ← 無向グラフ

$V=\{1,2,\ldots,10\}$ ← ノードラベルの集合



$E=\{\{1,2\},\{1,8\},\{2,3\},\{2,8\},\{2,9\},\ldots\}$ ←リンクラベルの集合

- 各ノードが1つのデータ要素に対応
  （確率変数が割り当てられる）

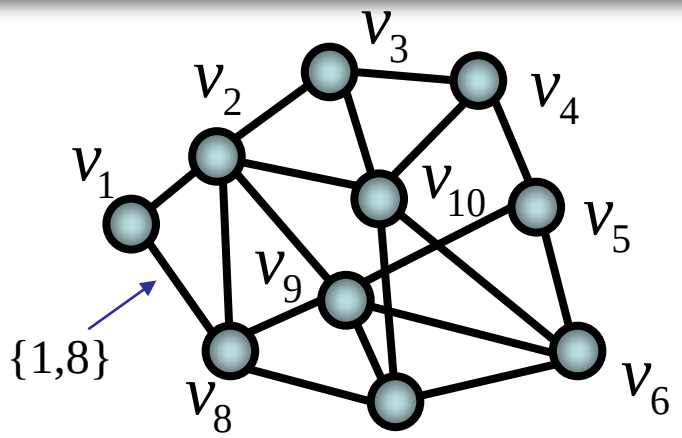
$\boldsymbol{v}=\{v_1,v_2,\ldots,v_{10}\}$

- リンクが変数同士の確率的な関連性を表現
  影響を直接及ぼすノード同士を結んでいる

マルコフ確率場の導入

$G(V,E)$ ← 無向グラフ

$V=\{1,2,\ldots,10\}$ ← ノードラベルの集合



$E=\{\{1,2\},\{1,8\},\{2,3\},\{2,8\},\{2,9\},\ldots\}$ ←リンクラベルの集合

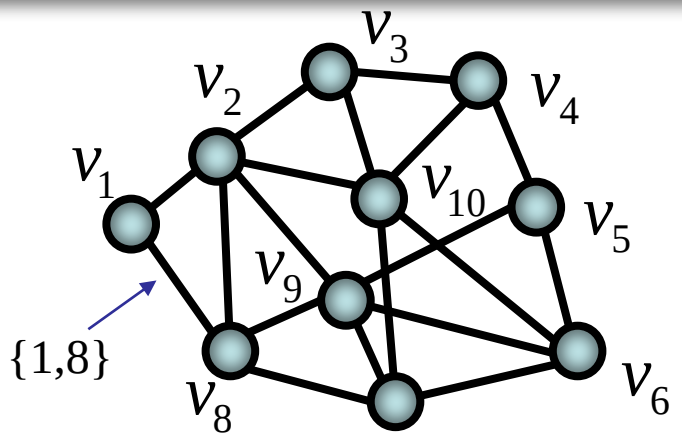
エネルギー関数（コスト関数）

$$C(\boldsymbol{v})=\underbrace{-\sum_{i\in V}\Phi_i(v_i)}_{\text{各ノードごとのコスト}}\underbrace{-\sum_{\{i,j\}\in E}\Psi_{\{i,j\}}(v_i,v_j)}_{\text{各リンクごとのコスト}}$$

マルコフ確率場の導入

$G(V,E)$ ← 無向グラフ

$V=\{1,2,\ldots,10\}$ ← ノードラベルの集合



$E=\{\{1,2\},\{1,8\},\{2,3\},\{2,8\},\{2,9\},\ldots\}$ ←リンクラベルの集合

エネルギー関数の和の意味

$$C(\boldsymbol{v})=\underbrace{-\sum_{i\in V}\Phi_i(v_i)}_{\text{各ノードごとのコスト}}\underbrace{-\sum_{\{i,j\}\in E}\Psi_{\{i,j\}}(v_i,v_j)}_{\text{各リンクごとのコスト}}$$

$\sum_{i\in V}\Phi_i(v_i)=\Phi_1(v_1)+\Phi_2(v_2)+\Phi_3(v_3)+\cdots+\Phi_{10}(v_{10})$ ← 全てのノードに対する和

$\sum_{\{i,j\}\in E}\Psi_{\{i,j\}}(v_i,v_j)=\Psi_{\{1,2\}}(v_1,v_2)+\Psi_{\{1,8\}}(v_1,v_8)+\Psi_{\{2,3\}}(v_2,v_3)+\cdots$ ← 全てのリンクに対する和

マルコフ確率場の導入

エネルギー関数（コスト関数）

$$C(\boldsymbol{v}) = \underbrace{-\sum_{i \in V} \Phi_i(v_i)}_{\text{各ノードごとのコスト}} \underbrace{- \sum_{\{i,j\} \in E} \Psi_{\{i,j\}}(v_i, v_j)}_{\text{各リンクごとのコスト}}$$

マルコフ確率場

$$P(\boldsymbol{v}) = \frac{1}{Z} \exp\left(-C(\boldsymbol{v})\right)$$

規格化定数（分配関数）

すべての変数の結合確率分布

分配関数

確率分布の規格化条件を保証するための定数

$$\sum_{\boldsymbol{v}} P(\boldsymbol{v}) = 1$$

変数の実現可能な値のすべての組み合わせの和は必ず 1 になるという条件

すべての変数が 0 か 1 の2値を取る場合

$$\sum_{\boldsymbol{v}} (\cdots) = \sum_{v_1=0,1} \sum_{v_2=0,1} \cdots \sum_{v_n=0,1} (\cdots)$$

$$\sum_{\boldsymbol{v}} P(\boldsymbol{v}) = \frac{1}{Z} \sum_{\boldsymbol{v}} \exp\left(-C(\boldsymbol{v})\right) = 1 \Leftrightarrow Z = \sum_{\boldsymbol{v}} \exp\left(-C(\boldsymbol{v})\right)$$

マルコフ確率場の導入

- 無向グラフのリンク構造に則ってエネルギー関数を設計
- エネルギー関数の各項の具体的な関数形は課題によって適宜設計する
- エネルギー関数の負を指数の肩に載せて規格化すると出来上がり！

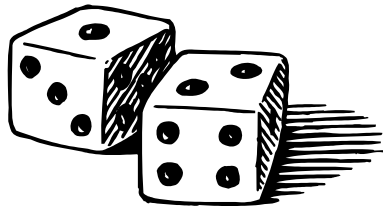
$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{v})\right)$$

エネルギー関数が低い $\boldsymbol{v}$ のとき確率が高くなる

エネルギー関数を低くする $\boldsymbol{v}$ のパターンが頻繁に生成される

マルコフ確率場は結合確率分布

我々は今何を作ったのだろうか？？



$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{v})\right)$$

例えばバイナリの確率変数が 3 個であるような状況を考える $\boldsymbol{v} = \left\{ v_i \in \{0,1\} \mid i = 1,2,3 \right\}$

$\boldsymbol{v}$ は 000, 001, 010, …, 111 の計 $2^3 = 8$ パターンをとる

そして
それぞれのパターンごとにそれぞれ MRF に従う確率値が割り当てられる

（MRF）＝（8 面サイコロ）

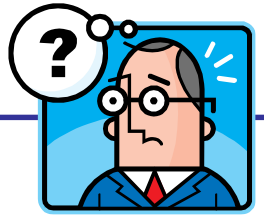
ただし各面の出る確率は等確率でなく MRF が定義する確率に従う

エネルギー関数を低くするパターンの面がよく出るサイコロ

マルコフ確率場からのサンプリング

MRF はサイコロらしいのだが
ではそのサイコロはどのように振ればよいのか？

MRF に従う疑似乱数の取得方法は？



多変数の確率分布においては

ある変数の値が他の変数の値の確率に影響を及ぼすため

一般には各変数ごとに独立に値を決めていくことができない
(ある変数の値を決める際に他の変数の値を気にする必要がある)

そのため疑似乱数のサンプリングが通常より難しい

ギブスサンプリング

MRF はサイコロらしいのだが
ではそのサイコロはどのように振ればよいのか？

MRF に従う疑似乱数の取得方法は？

マルコフ連鎖モンテカルロ（Markov chain Monte Carlo; MCMC）法

の中の

ギブス（Gibbs）サンプリング法

が有名

ギブスサンプリング（空間的マルコフ性）

その他の変数すべてを条件とした
ある確率変数 v_i についての条件付き確率を計算する

$$P(v_i \mid \boldsymbol{v}_{-i}) = \frac{P(\boldsymbol{v})}{P(\boldsymbol{v}_{-i})} \propto \exp\left(\Phi_i(v_i) + \sum_{j \in \partial(i)} \Psi_{\{i,j\}}(v_i, v_j) \right)$$

$\boldsymbol{v}_{-i} \leftarrow v_i$ 以外のすべての変数　　$\partial(i) \leftarrow$ ノード i とリンクでつながるノードの集合

$$P(v_i \mid \boldsymbol{v}_{-i}) = P(v_i \mid \boldsymbol{v}_{\partial(i)})$$

$\boldsymbol{v}_{\partial(i)} \leftarrow v_i$ とリンクでつながっている変数の集合

リンクでつながっている変数のみにしか依存しない！

ギブスサンプリングの方法

全ての変数を適当に初期化

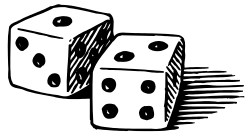
条件付き確率 $P\left(v_i \mid \boldsymbol{v}_{\partial(i)}\right)$ に従ってすべての変数を確率的に更新

$$\boldsymbol{v}^{(1)} \to \boldsymbol{v}^{(2)} \to \boldsymbol{v}^{(3)} \to \cdots \to \boldsymbol{v}^{(K)}$$

$$\boldsymbol{v}^{(K+1)} \to \boldsymbol{v}^{(K+2)} \to \boldsymbol{v}^{(K+3)} \to \cdots \to \boldsymbol{v}^{(2K)}$$

$$\boldsymbol{v}^{(2K+1)} \to \boldsymbol{v}^{(2K+2)} \to \boldsymbol{v}^{(2K+3)} \to \cdots \to \boldsymbol{v}^{(3K)}$$

$$\vdots$$



上記の手続きにより一定間隔（K ステップ間隔）でサンプリングする

サンプル点ごとの相関を切って独立なサンプリングを実現するため

ギブスサンプリングの方法

全ての変数を適当に初期化

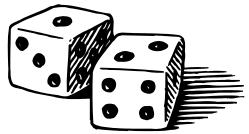
条件付き確率 $P\left(v_i \mid \boldsymbol{v}_{\partial(i)}\right)$ に従ってすべての変数を確率的に更新

$$\boldsymbol{v}^{(1)} \rightarrow \boldsymbol{v}^{(2)} \rightarrow \boldsymbol{v}^{(3)} \rightarrow \cdots \rightarrow \boldsymbol{v}^{(K)}$$

$$\boldsymbol{v}^{(K+1)} \rightarrow \boldsymbol{v}^{(K+2)} \rightarrow \boldsymbol{v}^{(K+3)} \rightarrow \cdots \rightarrow \boldsymbol{v}^{(2K)}$$

$$\boldsymbol{v}^{(2K+1)} \rightarrow \boldsymbol{v}^{(2K+2)} \rightarrow \boldsymbol{v}^{(2K+3)} \rightarrow \cdots \rightarrow \boldsymbol{v}^{(3K)}$$

$$\vdots$$



初期の段階のサンプル点は出現確率が安定しないので捨てて
（緩和期間）

十分回更新した後のサンプル点から採用する

ボルツマンマシン
～MRFの基本モデル～

ボルツマンマシン(Boltzmann machine; BM)とは？

- もっとも基礎的な MRF
- 連想記憶モデルであるホップフィールドモデルの確率的拡張モデル
- 相互結合型の確率的ニューラルネットワーク
- 最近のディープラーニング（深層学習）ブームで再燃焼

$\Phi_i(v_i) \to b_i v_i \quad \Psi_{\{i,j\}}(v_i, v_j) \to w_{ij} v_i v_j$ と単純な関数形で仮定する

ボルツマンマシンのエネルギー関数

$$C(\boldsymbol{v}; \boldsymbol{b}, \boldsymbol{w}) = -\sum_{i \in V} b_i v_i - \sum_{\{i,j\} \in E} w_{ij} v_i v_j$$

$v_i \in \{0,1\}$

バイアスパラメータ

結合パラメータ

$w_{ij} = w_{ji}$（対称性）

ボルツマンマシン [Ackley *et.al.*, 85]

$$P(\boldsymbol{v} \mid \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{b}, \boldsymbol{w})} \exp\left(\sum_{i \in V} b_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_i v_j \right)$$

$$v_i \in \{0,1\}$$

$$C(\boldsymbol{v}; \boldsymbol{b}, \boldsymbol{w}) = -\sum_{i \in V} b_i v_i - \sum_{\{i,j\} \in E} w_{ij} v_i v_j$$

エネルギー関数が低いほど確率が高い

- もし $b_i > 0$ ならば局所的には $v_i = 1$ で逆に $b_i < 0$ なら $v_i = 0$ が確率的に高い
- もし $w_{ij} > 0$ ならば $v_i = 1$ & $v_j = 1$ が局所的には確率が高い
- もし $w_{ij} < 0$ ならば $v_i = 1$ & $v_j = 1$ でない方が局所的には確率が高い

ボルツマンマシン [Ackley *et.al.*, 85]

$$P(\boldsymbol{v} \mid \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{b}, \boldsymbol{w})} \exp\left(\sum_{i \in V} b_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_i v_j \right)$$

$$v_i \in \{0,1\}$$

エネルギー関数中のリンクの項が

変数同士の複雑な関連性

を生み出している

ボルツマンマシン
に対する
統計的機械学習

観測データ

N given (n dimensional) data points:

$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ dimensional data point}}$$

$$\mathrm{d}_i^{(\mu)} \in \{0,1\}$$

$$\underbrace{0100101\cdots001}_{n} \quad \mathbf{d}^{(1)}$$

$$1011011\cdots111 \quad \mathbf{d}^{(2)}$$

$$\vdots$$

$$\underbrace{0010011\cdots101}_{n} \quad \mathbf{d}^{(N)}$$

それぞれのデータはある生成モデルから独立に生成されているとする

$$P_g(\boldsymbol{v}) \xrightarrow[\text{generate (i.i.d.)}]{} \mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}$$

観測データのセットを使って BM を設計する方法を考える

以降 $|V| = n$ を前提として話を進める

データの次元(n 次元)
と BM の変数の数が同じ
という意味

観測データ

N given (n dimensional) data points:

$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ dimensional data point}}$$

$$\mathrm{d}_i^{(\mu)} \in \{0,1\}$$

$$\underbrace{0100101\cdots001}_{n} \quad \mathbf{d}^{(1)}$$

$$1011011\cdots111 \quad \mathbf{d}^{(2)}$$

$$\vdots$$

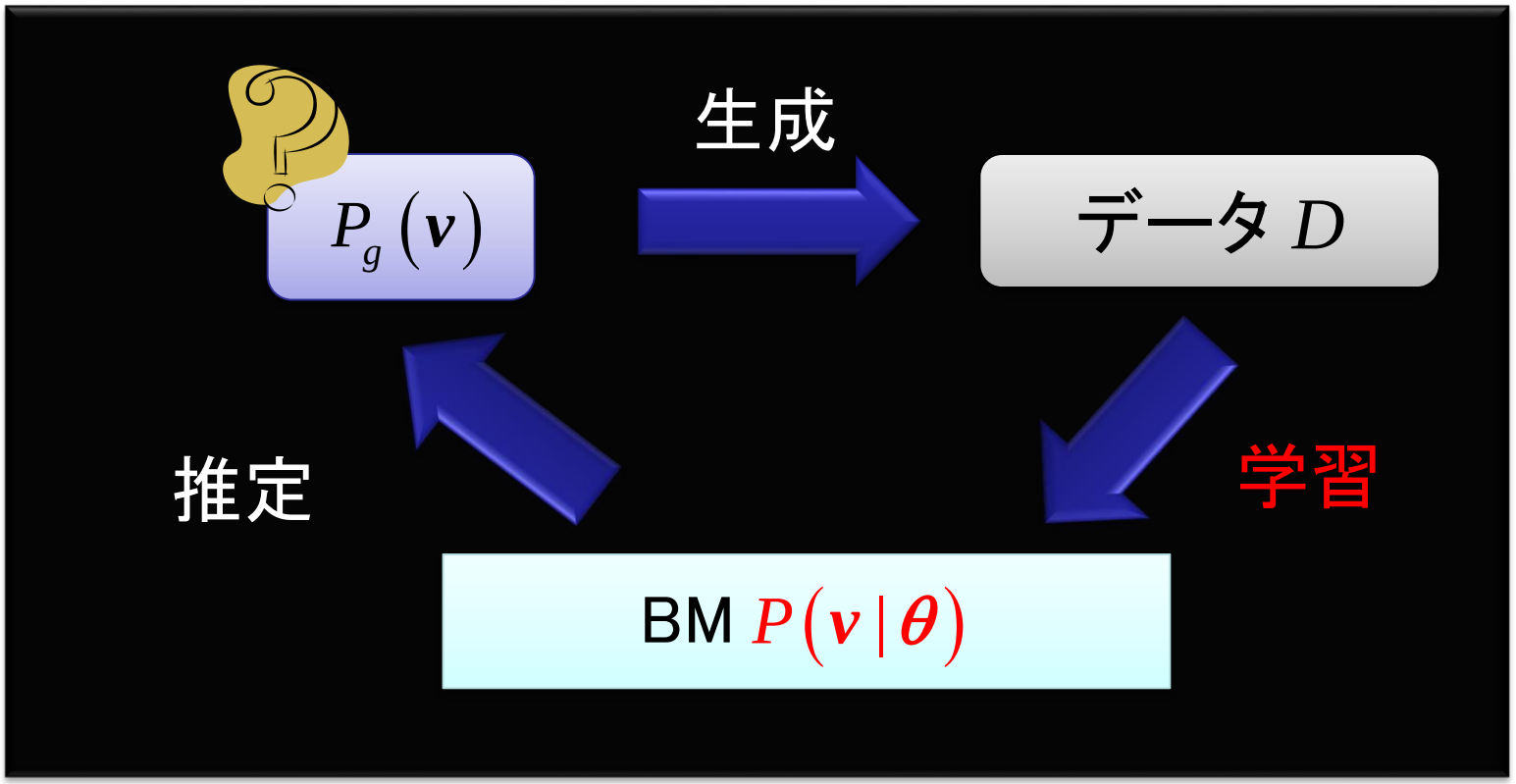
$$0010011\cdots101 \quad \mathbf{d}^{(N)}$$

$d_i^{(\mu)}$ を v_i に対応させる

データの要素一つ一つが
各変数に対応している状況

統計的機械学習の枠組み

$P(\boldsymbol{v}|\boldsymbol{\theta})$ ← パラメータ θ をもった BM を仮定する $\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{w}\}$



- データ D を利用して仮定した BM を学習する
- 学習により（未知の）生成モデルに BM を近づける

最尤推定の考え方

統計的機械学習の基本的考え方

最尤推定（maximum likelihood estimation; MLE）

BM $P(\boldsymbol{v}|\boldsymbol{\theta})$ → データ D

観測データは BM（学習モデル）から i.i.d. で生成されたと考える

実際は未知の事前確率から生成されているのだが
事前確率を学習モデルで近似するという考え方からこのように考える

最尤推定の考え方

統計的機械学習の基本的考え方

最尤推定（maximum likelihood estimation; MLE）

BM $P(\boldsymbol{v}|\boldsymbol{\theta})$ → データ D

観測データを生成するに

もっとも尤もらしい確率になるようにパラメータを調整する

何故なら観測データはそのモデルから生成されたと考えるため

最尤推定(尤度関数)

$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ dimensional data point}}$$

観測データと同じ次元(n次元)の確率変数をもつ BM を用意する

$$P\left(\boldsymbol{v} \mid \boldsymbol{\theta}\right) \qquad \boldsymbol{v} = \left\{v_1, v_2, \ldots, v_n\right\}$$

$P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \boldsymbol{\theta}\right)$ ← MRFが μ 番目のデータを生成する確率

BM が観測データセット D を生成する確率

$$\underbrace{P\left(\boldsymbol{v} = \mathbf{d}^{(1)} \mid \boldsymbol{\theta}\right) P\left(\boldsymbol{v} = \mathbf{d}^{(2)} \mid \boldsymbol{\theta}\right) \cdots P\left(\boldsymbol{v} = \mathbf{d}^{(N)} \mid \boldsymbol{\theta}\right)}_{\text{各データは独立に生成されるので積}} = \prod_{\mu=1}^{N} P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \boldsymbol{\theta}\right) = (\text{尤度関数})$$

最尤推定（尤度関数）

尤度関数

$$L_D(\boldsymbol{\theta}) = \prod_{\mu=1}^{N} P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \boldsymbol{\theta}\right)$$

仮定した学習モデルが
実際に観測データを生成する確率

観測データセットに対して一番尤もらしいモデル

↓

尤度関数を最大とするパラメータ値のときのモデル

※　観測データセットを生成する確率が一番高いモデル

最尤推定

最尤推定

$$\boldsymbol{\theta}^{*} = \arg\max_{\boldsymbol{\theta}} L_D(\boldsymbol{\theta})$$

$P(\boldsymbol{v}|\boldsymbol{\theta}^{*})$ が観測データに対して一番尤もらしいモデル

（モデル設計の枠内で）事前確率に一番近いモデル

$$P_g(\boldsymbol{v}) \approx P(\boldsymbol{v}|\boldsymbol{\theta}^{*})$$

対数尤度関数

$$l_D(\boldsymbol{\theta}) = \frac{1}{N} \ln L_D(\boldsymbol{\theta}) = \frac{1}{N} \sum_{\mu=1}^{N} \ln P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \boldsymbol{\theta}\right)$$

尤度関数の対数をとったもの

積が和に変身するので計算的に扱い易いしオーバー(アンダー)フロー等にも強い

対数関数は単調増加関数なので

(尤度関数の最大点)＝(対数尤度関数の最大点)

なので通常最尤推定をするときは

尤度関数よりもむしろ対数尤度関数を用いる

最尤推定のもう一つの解釈

カルバック・ライブラー（Kullback-Leibler; KL）情報量

2つの確率分布 P と Q に対して

$$D(P\|Q)=\sum_{\boldsymbol{v}} P(\boldsymbol{v})\ln\frac{P(\boldsymbol{v})}{Q(\boldsymbol{v})}$$

$\boxed{D(P\|Q)\geq 0}$ ← 非負

$\boxed{D(P\|Q)=0 \quad \text{iff} \quad P(\boldsymbol{v})=Q(\boldsymbol{v})}$ ← 2つの確率が一致するとき 0

2つの確率分布間の距離の指標として使用される

※　距離の定義を満足しないので厳密な意味での距離ではない

KL情報量と最尤推定

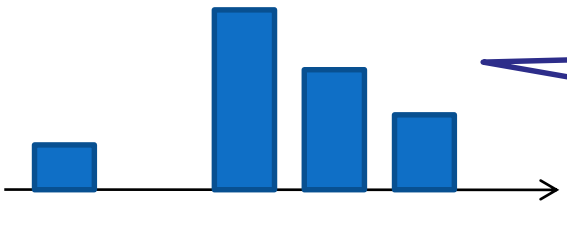
$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ dimensional data point}}$$

$$H_D(\boldsymbol{v}) = \frac{1}{N}\sum_{\mu=1}^{N}\delta\left(\boldsymbol{v}, \mathbf{d}^{(\mu)}\right) \leftarrow \underbrace{\text{観測データの分布（ヒストグラム）}}$$

経験分布 (emprirical distribution)

クロネッカーのデルタ関数 $\delta\left(\boldsymbol{v}, \mathbf{d}^{(\mu)}\right) = \begin{cases} 1 & \left(\boldsymbol{v} = \mathbf{d}^{(\mu)}\right) \\ 0 & \left(\boldsymbol{v} \neq \mathbf{d}^{(\mu)}\right) \end{cases}$

※　確率変数が連続の場合はディラックのデルタ関数になる



観測データが存在するところに

1 個ずつ積み上げていく

KL情報量と最尤推定

経験分布 $H_D(\boldsymbol{v})$ と学習モデル $P(\boldsymbol{v}|\boldsymbol{\theta})$ との間のKL情報量

$$D(H_D \| P) = \sum_{\boldsymbol{v}} H_D(\boldsymbol{v}) \ln \frac{H_D(\boldsymbol{v})}{P(\boldsymbol{v}|\boldsymbol{\theta})}$$

$$D(H_D \| P) = \underbrace{\sum_{\boldsymbol{v}} H_D(\boldsymbol{v}) \ln H_D(\boldsymbol{v})}_{\boldsymbol{\theta} \text{ に関係ない項}} - \quad l_D(\boldsymbol{\theta})$$

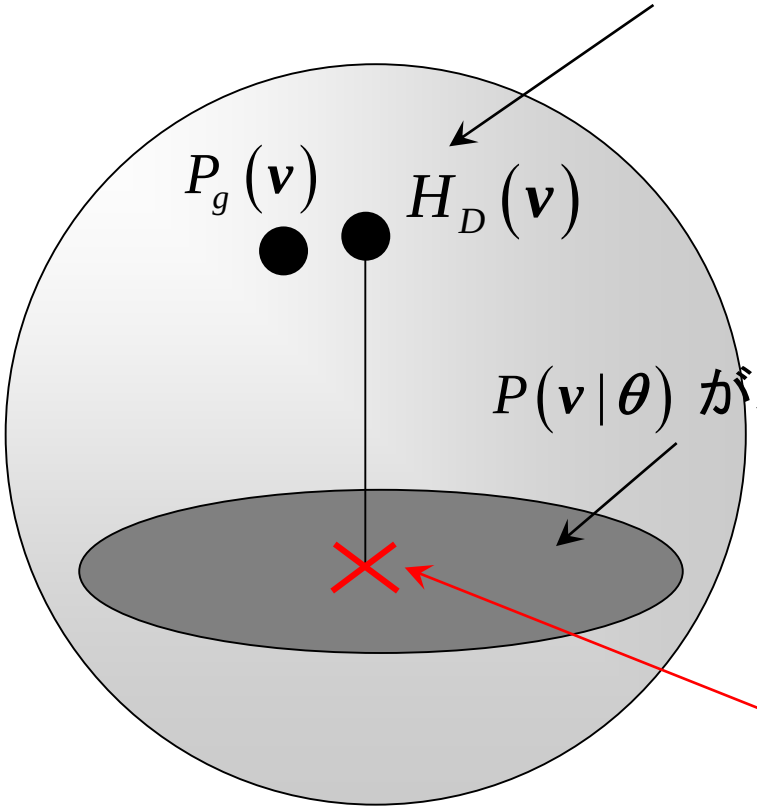
対数尤度関数

（最尤推定）＝（KL情報量の最小化）

（最尤推定）＝（観測データの分布に一番近い分布を見つけること）

最尤推定とKL情報量（イメージ）

事前確率（生成モデル）と経験分布は比較的近いだろう



ボルツマンマシンの最尤推定

$$P(\boldsymbol{v}\,|\,\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{i \in V} b_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_i v_j \right)$$

$$v_i \in \{0,1\}$$

対数尤度関数　$\boldsymbol{\theta} = \{\boldsymbol{b}, \boldsymbol{w}\}$

$$l_D(\boldsymbol{\theta}) = \sum_{i \in V} b_i \mathrm{E}_D[v_i] + \sum_{\{i,j\} \in E} w_{ij} \mathrm{E}_D[v_i v_j] - \ln Z(\boldsymbol{\theta})$$

$\mathrm{E}_D[\cdots]$ ← 観測データの標本平均　$\left(\text{e.g. } \mathrm{E}_D[v_i] = \frac{1}{N} \sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)},\ \mathrm{E}_D[v_i v_j] = \frac{1}{N} \sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} \mathrm{d}_j^{(\mu)} \right)$

勾配

$$\mathrm{E}[\cdots\,|\,\boldsymbol{\theta}] = \sum_{\boldsymbol{v}} (\cdots) P(\boldsymbol{v}\,|\,\boldsymbol{\theta})$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial b_i} = \mathrm{E}_D[v_i] - \mathrm{E}[v_i\,|\,\boldsymbol{\theta}], \quad \frac{\partial l_D(\boldsymbol{\theta})}{\partial w_{ij}} = \mathrm{E}_D[v_i v_j] - \mathrm{E}[v_i v_j\,|\,\boldsymbol{\theta}]$$

モーメントマッチング方程式

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial b_i} = \mathrm{E}_D[v_i] - \mathrm{E}[v_i \mid \boldsymbol{\theta}], \quad \frac{\partial l_D(\boldsymbol{\theta})}{\partial w_{ij}} = \mathrm{E}_D[v_i v_j] - \mathrm{E}[v_i v_j \mid \boldsymbol{\theta}]$$

対数尤度関数を最大化すればよいので勾配上昇法

ところで最大点（勾配が 0 の点）では。。。

指数分布族と呼ばれる分布族を
モデルにすると一般的に表れる方程式

$$\mathrm{E}_D[v_i] = \mathrm{E}[v_i \mid \boldsymbol{\theta}], \quad \mathrm{E}_D[v_i v_j] = \mathrm{E}[v_i v_j \mid \boldsymbol{\theta}]$$

（データの標本平均）＝（モデルの対応する期待値）

→ モーメントマッチング方程式 ！

対数尤度関数の最大化

複雑な関数の最大化問題（最小化問題）における

常套手段は

勾配法

を利用したコンピュータによる

繰り返し計算による数値的最適化法

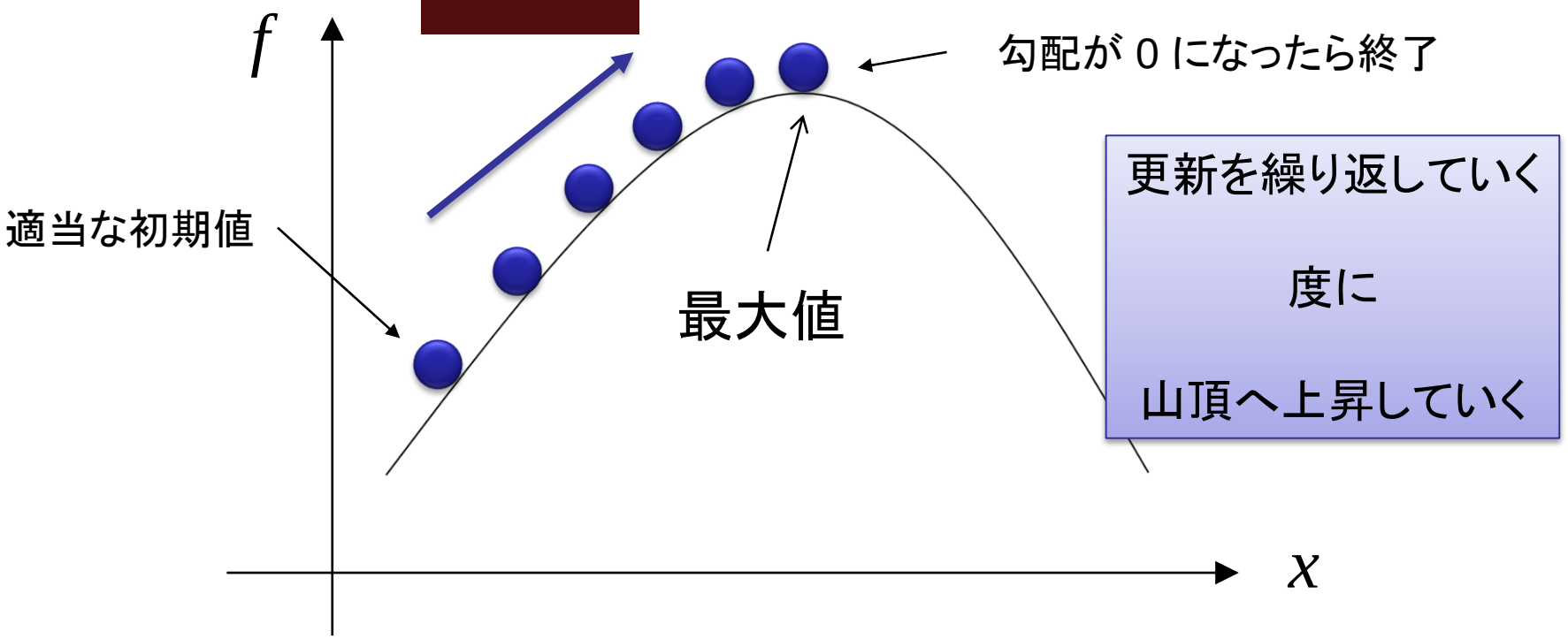
勾配上昇法

山に登らせる更新式

$$x^{\text{new}} \leftarrow x^{\text{old}} + \varepsilon \frac{\partial f\left(x^{\text{old}}\right)}{\partial x}$$

勾配
（導関数）

ε：小さな正の数（学習率）



対数尤度関数の最大化

というわけで

コンピュータを使って勾配上昇法

$$b_i^{\text{new}} \leftarrow b_i^{\text{old}} + \varepsilon\left(\mathrm{E}_D\left[v_i\right] - \mathrm{E}\left[v_i \mid \boldsymbol{\theta}^{\text{old}}\right]\right)$$

$$w_{ij}^{\text{new}} \leftarrow w_{ij}^{\text{old}} + \varepsilon\left(\mathrm{E}_D\left[v_i v_j\right] - \mathrm{E}\left[v_i v_j \mid \boldsymbol{\theta}^{\text{old}}\right]\right)$$

BM の対数尤度関数は
パラメータに関して凹関数なので

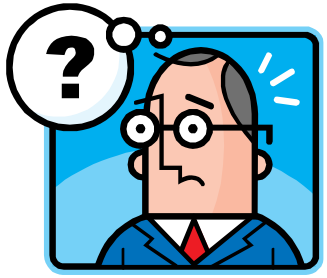
原理的には勾配法で最適な学習解を得ることができる

マルコフ確率場の欠点？

何故今になってマルコフ確率場？

MRF はコンピュータビジョン（CV）等のパターン認識分野での
スタンダードモデルになり始めている

MRF 自体はかなり昔から知られたモデル



何故今になってマルコフ確率場が浮上してきたのか？
何故今まではほとんど利用されてきてこなかったのか？

マルコフ確率場の期待値

確率モデルにおいて期待値計算は基本中の基本！

変数の数を n とする

$$\mathrm{E}\left[v_i \mid \boldsymbol{\theta}\right] = \sum_{\boldsymbol{v}} v_i P\left(\boldsymbol{v} \mid \boldsymbol{\theta}\right)$$

$\sum_{\boldsymbol{v}}(\cdots) = \sum_{v_1}\sum_{v_2}\cdots\sum_{v_n}$ ← 実現可能なすべて値の組み合わせに関する和

変数が 0 or 1 のバイナリだとすると

この和は 2^n 個の項の和になる

組み合わせ爆発の問題

指数関数的な計算量増加はかなり危険！

仮に 1 秒間に 1 億回の足し算が可能であるPCがあるとする

2^n 回の演算に必要な時間は…

同様なロジックで
規格化定数 Z も
そもそも計算できない

n	time
10	約 0.00001 秒
30	約 0.18 分
50	約 130 日
70	約 37万4千 年
100	約 4千億 年

組み合わせ爆発の問題

指数関数的な計算量増加はかなり危険！

組み合わせ爆発の問題により

MRF の実装は

非現実的と考えられていた

BMの様々な近似学習法

BM の近似学習アルゴリズムは多数知られている

確率伝搬法（belief propagation） [Yasuda & Horiguchi, 06]

コントラスティブ・ダイバージェンス（contrastive divergence; CD） [Hinton, 02]

疑似最尤法（pseudo likelihood estimation; PLE） [Basag, 75]

複合最尤法（composite likelihood estimation） [Lindsay, 88]

スコア・マッチング（score matching） [Hyvärinen, 05]

最小確率流（minimum probability flow） [Sohl-Dickstein *et. al.*, 11]

拡張モンテカルロ積分法（spatial Monte Carlo integration） [Yasuda, 15]

etc.

ガウシアングラフィカルモデル

確率変数が連続の実数値の場合の MRF

（連続値のボルツマンマシン）

$$P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w})} \exp\left(-\frac{1}{2} \sum_{i \in V} \lambda_i v_i^2 + \sum_{i \in V} b_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_i v_j \right) \quad (\lambda_i > 0)$$

一般形

$v_i \in (-\infty, +\infty)$ ← 確率変数が実数値

規格化定数が解析的に計算できる！

$$Z(\boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) = \exp\left(\frac{1}{2} \boldsymbol{b}^T \boldsymbol{C}^{-1} \boldsymbol{b} \right) \sqrt{(2\pi)^{|V|} \det \boldsymbol{C}^{-1}}$$

$$C_{ij} = \begin{cases} \lambda_i & (i = j) \\ -w_{ij} & (\{i, j\} \in E) \\ 0 & (\text{otherwise}) \end{cases}$$

ガウシアングラフィカルモデル
（Gaussian graphical model; GGM）

ガウシアングラフィカルモデル

GGM は多次元のガウス分布

$$P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^{|V|} \det \boldsymbol{C}^{-1}}} \exp\left(-\frac{1}{2} (\boldsymbol{v} - \boldsymbol{m})^T \boldsymbol{C} (\boldsymbol{v} - \boldsymbol{m}) \right)$$

$$\boldsymbol{m} = \boldsymbol{C}^{-1} \boldsymbol{b}$$

期待値もとても簡単に計算可能

$$\mathrm{E}\left[v_i \mid \boldsymbol{\theta} \right] = \int_{-\infty}^{\infty} v_i P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) \mathrm{d}\boldsymbol{v} = m_i$$

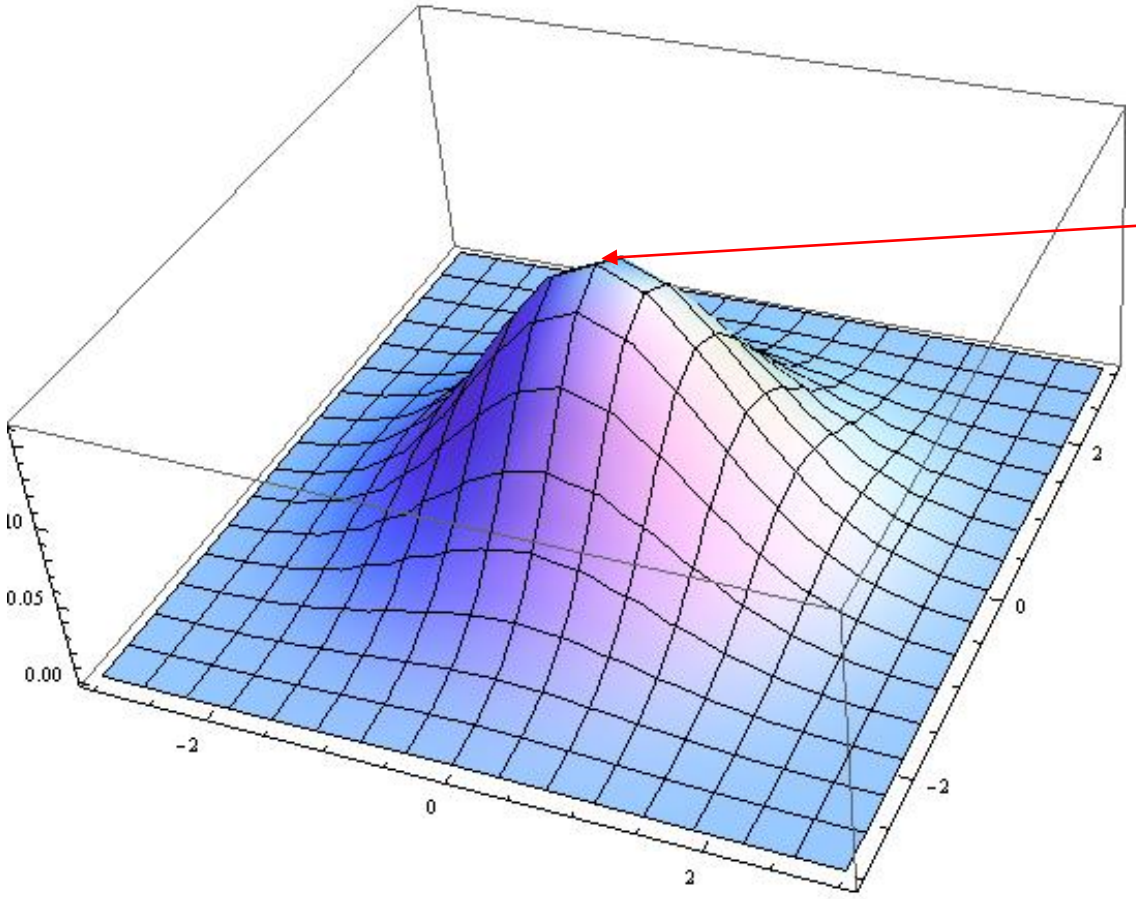
$$\mathrm{E}\left[v_i^2 \mid \boldsymbol{\theta} \right] = \int_{-\infty}^{\infty} v_i^2 P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) \mathrm{d}\boldsymbol{v} = C_{ii}^{-1} + m_i^2$$

$$\mathrm{E}\left[v_i v_j \mid \boldsymbol{\theta} \right] = \int_{-\infty}^{\infty} v_i v_j P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) \mathrm{d}\boldsymbol{v} = C_{ii}^{-1} + m_i m_j$$

$C_{ij}^{-1} \leftarrow$ 行列 C の逆行列 $\boldsymbol{C}^{-1}$ の (i, j) 成分

ガウシアングラフィカルモデル

$$P(\boldsymbol{v} \mid \boldsymbol{\lambda}, \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^{|V|} \det \boldsymbol{C}^{-1}}} \exp\left(-\frac{1}{2} (\boldsymbol{v} - \boldsymbol{m})^T \boldsymbol{C} (\boldsymbol{v} - \boldsymbol{m}) \right)$$



ピーク点は平均ベクトル

$$\boldsymbol{m} = \boldsymbol{C}^{-1}\boldsymbol{b}$$

確率最大の点も
簡単に分かる！

ガウシアングラフィカルモデル

ガウシアングラフィカルモデル
ならば

特別な近似なしに

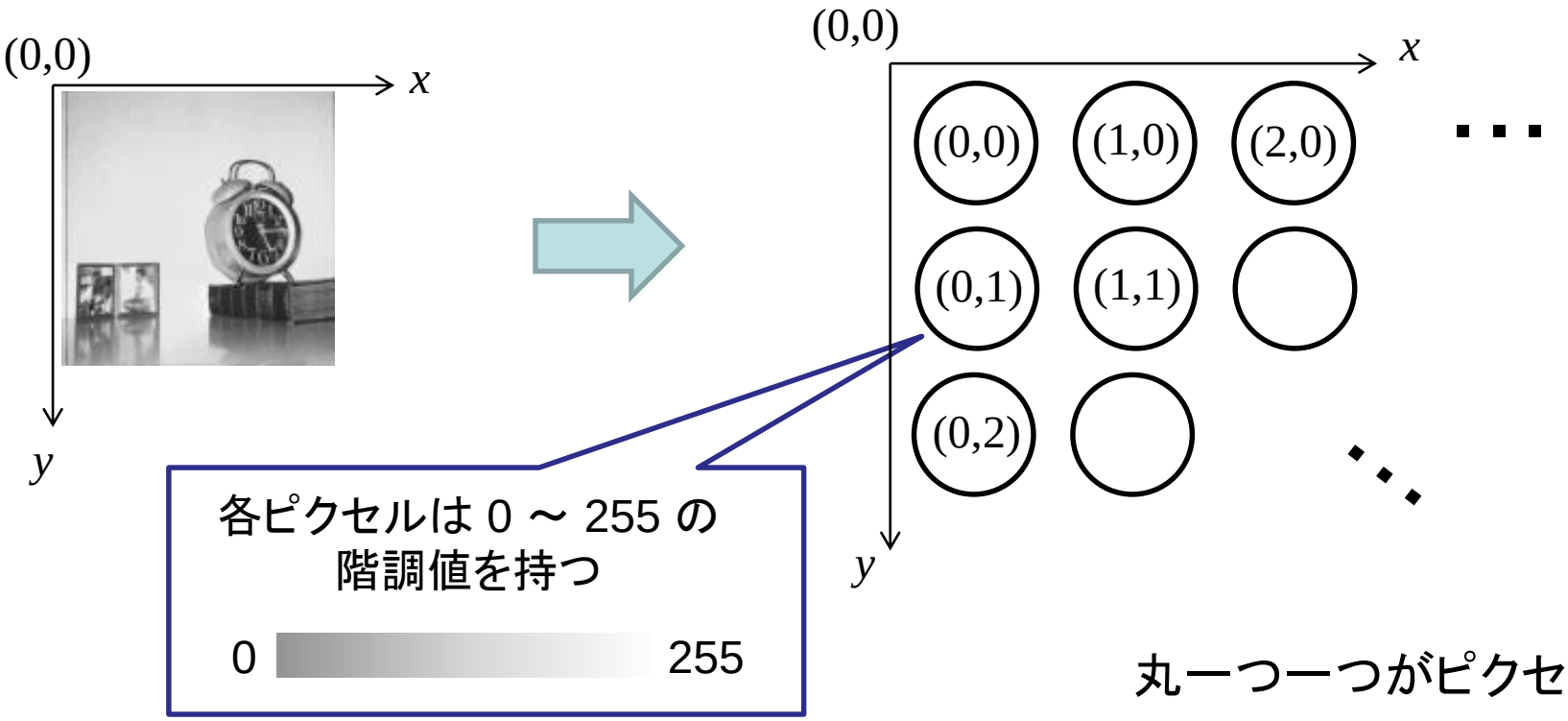
期待値計算と最大値計算
が厳密に可能である

ベイズ推定とMRFの応用例

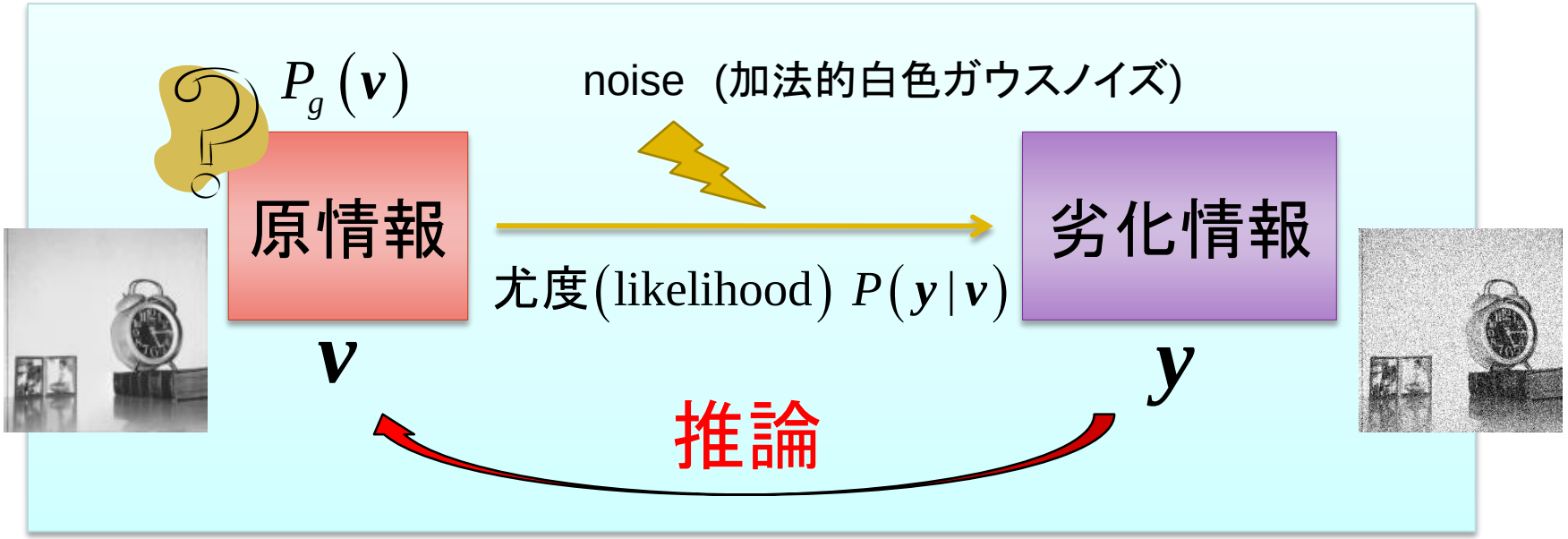
ディジタル画像

- 碁盤の目（正方格子）状に並べられたピクセル（pixel）とよばれる画素の集まりで表現される

- 各ピクセルには階調値（輝度値）とよばれる輝度（色）が割り当てられる



MRFによる画像ノイズ除去の枠組み



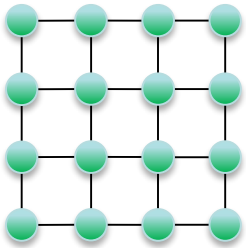
劣化した画像から原画像をベイズの枠組みにより推定する

尤度はガウスノイズで設計することとするとしても

さて事前確率はどうしましょう？

→ MRF で設計することとしましょう

MRFで画像の事前確率を作る



- 各ピクセルを各確率変数に対応させる

  → 確率変数はピクセルの個数分ある（超高次元）

- 確率変数のとる値は階調値にする $v_i \in \{0,1,2,\ldots,255\}$

- ピクセルは正方格子状にならんでいるので同様に正方格子のグラフ上に定義する（各ノードが各ピクセル）

- リンクは最近接のピクセル対にのみあるとする

$$C(\boldsymbol{v}) = \underbrace{-\sum_{i \in V} \Phi_i(v_i)}_{\text{各ノードごとのコスト}} \underbrace{- \sum_{\{i,j\} \in E} \Psi_{\{i,j\}}(v_i, v_j)}_{\text{各リンクごとのコスト}}$$

エネルギー関数をどのように設計すればよいか？？

MRFで画像の事前確率を作る

$$C(\boldsymbol{v}) = \underbrace{-\sum_{i \in V} \Phi_i(v_i)}_{\text{各ノードごとのコスト}} \underbrace{- \sum_{\{i,j\} \in E} \Psi_{\{i,j\}}(v_i, v_j)}_{\text{各リンクごとのコスト}}$$

画像に対する我々の先見知識を基に設計する

空間的に近くにあるピクセル同士は同じ色を取りやすいと仮定

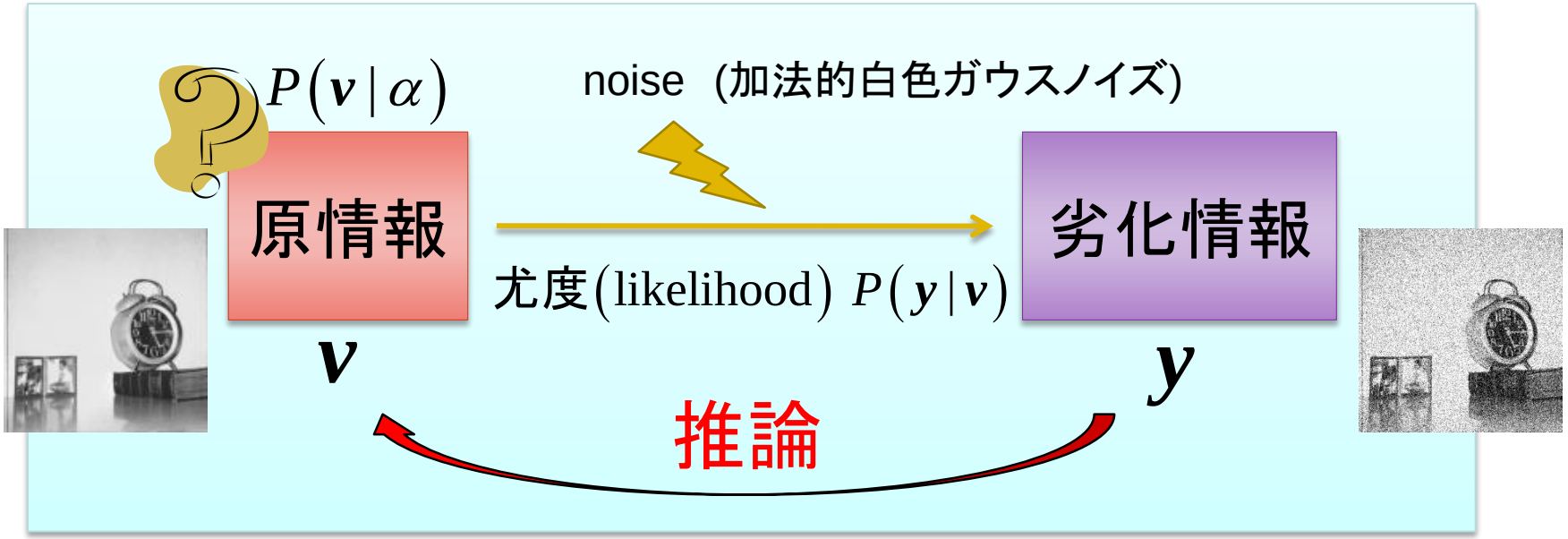
リンクコスト: $\Psi_{\{i,j\}}(v_i, v_j)$ ・・・ v_i と v_j の値が近い程大きくなるようにする

例 $\Psi_{\{i,j\}}(v_i, v_j) = -\alpha |v_i - v_j|$, $\Psi_{\{i,j\}}(v_i, v_j) = -\alpha (v_i - v_j)^2$ $(\alpha > 0)$

座標ごとの一貫した個性は特にないと仮定

$$\Phi_i(v_i) = 0$$

MRFによる事前確率



$$P_g(\boldsymbol{v}) \underset{\text{MRFでモデル化}}{\rightarrow} P(\boldsymbol{v}|\alpha) = \frac{1}{Z(\alpha)} \exp\left(-\alpha \sum_{\{i,j\}\in E} |v_i - v_j| \right)$$

ノイズは標準偏差 σ のガウスノイズでピクセルごとに独立にのるとすると

$$(\text{尤度}) = P(\boldsymbol{y}|\boldsymbol{v},\sigma^2) = \prod_{i\in V} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - v_i)^2}{2\sigma^2} \right)$$

事後確率を計算する

$$(\text{事後確率}) \propto (\text{尤度}) \times (\text{事前確率})$$

$$P\left(\boldsymbol{v} \mid \boldsymbol{y}, \alpha, \sigma^2\right) \propto P\left(\boldsymbol{y} \mid \boldsymbol{v}, \sigma^2\right) P\left(\boldsymbol{v} \mid \alpha\right) \propto \exp\left(-\sum_{i \in V} \frac{\left(y_i - v_i\right)^2}{2\sigma^2} - \alpha \sum_{\{i,j\} \in E} \left| v_i - v_j \right| \right)$$

$\boldsymbol{x}$ ・・・ 原画像（確率変数）

$\boldsymbol{y}$ ・・・ 劣化画像（データ）

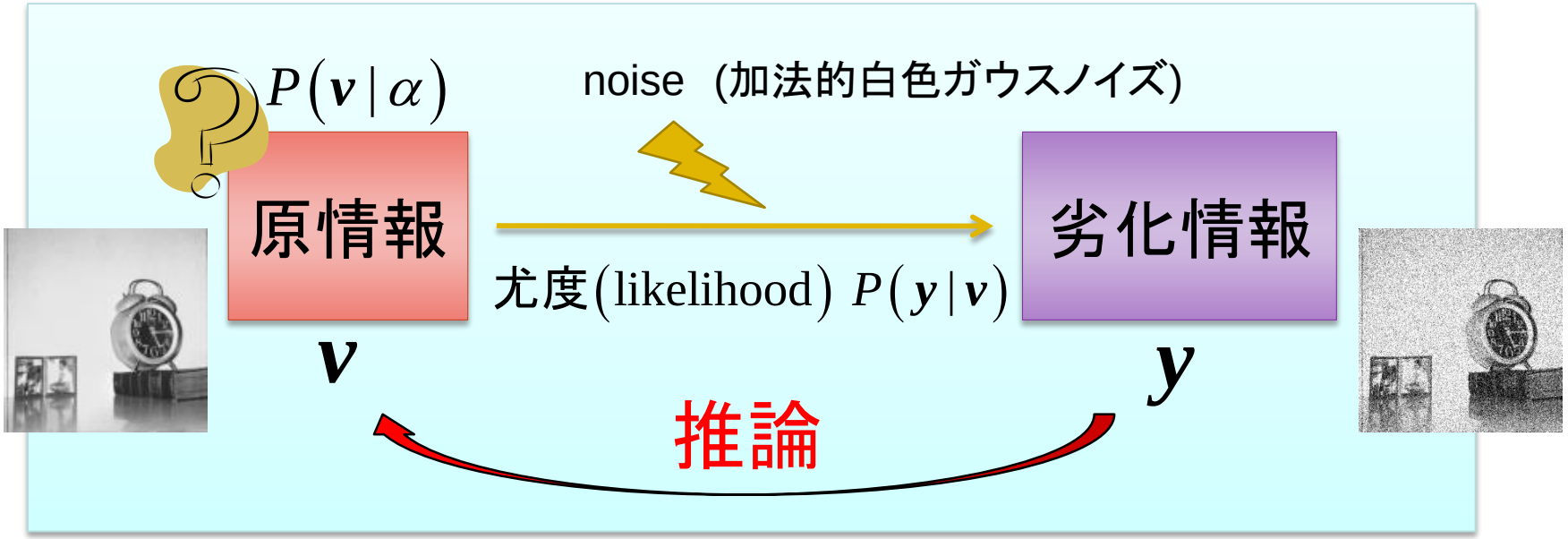
$$P\left(\boldsymbol{v} \mid \boldsymbol{y}, \alpha, \sigma^2\right) = \frac{1}{Z\left(\alpha, \sigma^2\right)} \exp\left(-\sum_{i \in V} \frac{\left(y_i - v_i\right)^2}{2\sigma^2} - \alpha \sum_{\{i,j\} \in E} \left| v_i - v_j \right| \right)$$

ノードのコスト（データ項）

リンクのコスト（平滑化項）

事後確率も MRF になる

MRFによるノイズ除去フィルタ

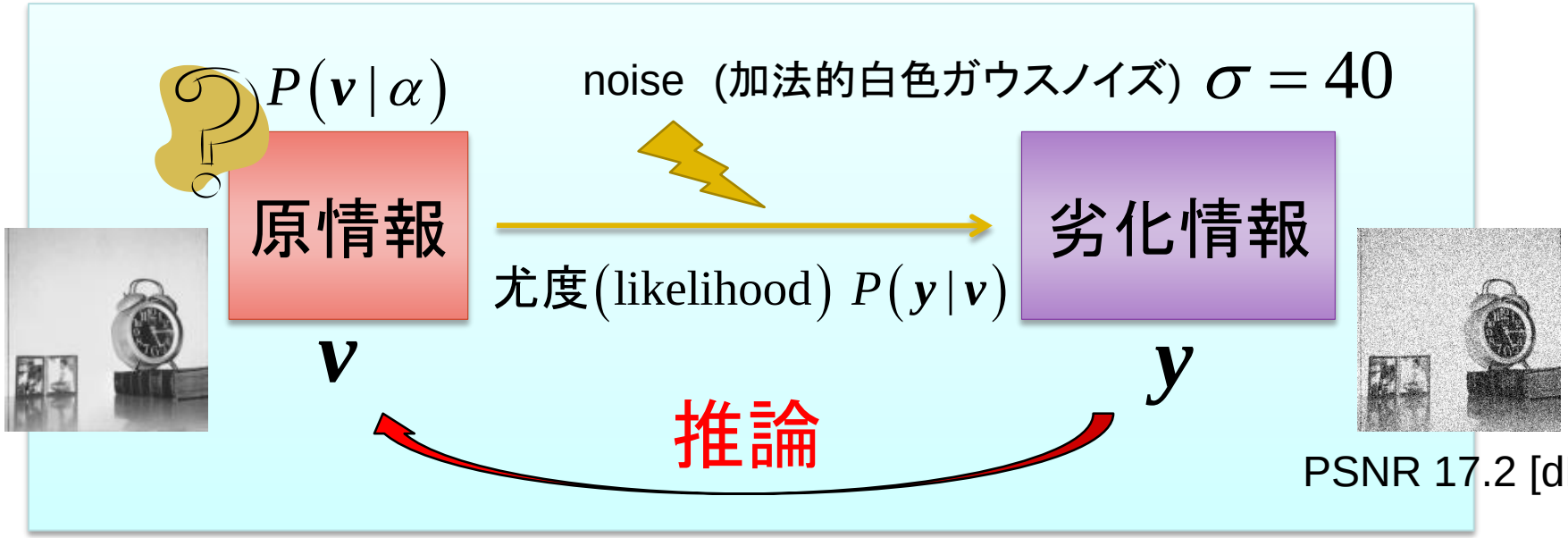


最大事後確率(MAP)推定

$$\left(\text{推定画像 } \boldsymbol{v}^*\right) = \arg\max_{\boldsymbol{v}} \underbrace{P\left(\boldsymbol{v}\,|\,\boldsymbol{y},\alpha,\sigma^2\right)}_{\text{事後確率}}$$

max-product アルゴリズムにより最大化を実行

MRFによるノイズ除去フィルタ



MRF PSNR 25.6 [dB]

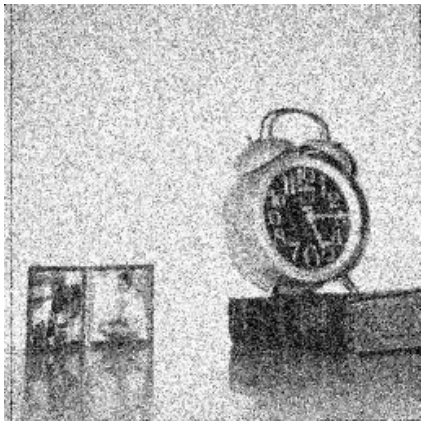
平滑化フィルタ PSNR 24.2 [dB]

MRFによるノイズ除去フィルタ

$$P\left(\boldsymbol{v} \mid \boldsymbol{y}, \alpha, \sigma^2\right) = \frac{1}{Z\left(\alpha, \sigma^2\right)} \exp\left(-\sum_{i \in V} \frac{\left(y_i - v_i\right)^2}{2\sigma^2} - \alpha \sum_{\{i,j\} \in E} \left| v_i - v_j \right| \right)$$

$\alpha \leftarrow$ データ項と平滑化項の重要度を調整するパラメータ

$\alpha = 0.002$

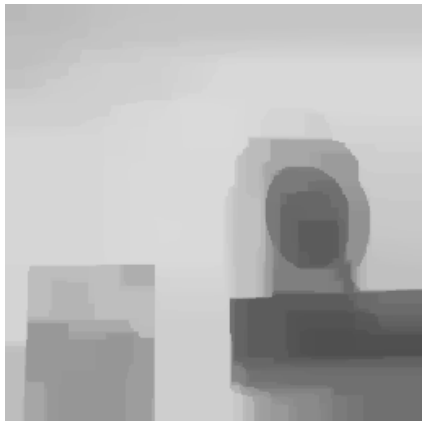


PSNR 19.0 [dB]

$\alpha = 0.02$

PSNR 25.6 [dB]

$\alpha = 0.2$



PSNR 18.8 [dB]

パラメータの値によって結果はかなり異なる

良いパラメータ設定がかなり重要

良い結果を得るためには
パラメータの値を良いものにしないといけない

しかし我々はその値を事前には知り得ない

どうすればよいだろうか？？

先程の例はノイズの分散も既知としたものである

しかし実際にはそれすら知り得ないこともある

α と σ^2 の値をなんらかの方法で推定したい

結合確率から出発する

原画像 $\boldsymbol{v}$ と劣化画像 $\boldsymbol{y}$ の結合確率は

$$P\left(\boldsymbol{v},\boldsymbol{y}\,|\,\alpha,\sigma^2\right)=P\left(\boldsymbol{y}\,|\,\boldsymbol{v},\sigma^2\right)P\left(\boldsymbol{v}\,|\,\alpha\right)$$

$$=\left\{\prod_{i\in V}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(y_i-v_i\right)^2}{2\sigma^2}\right)\right\}\times\left\{\frac{1}{Z(\alpha)}\exp\left(-\alpha\sum_{\{i,j\}\in E}\left|v_i-v_j\right|\right)\right\}$$

$$P\left(\boldsymbol{v},\boldsymbol{y}\,|\,\alpha,\sigma^2\right)$$

データとして
観測されない変数

隠れ変数
（潜在変数）

データ（劣化画像）として
観測されている変数

可視変数
（観測変数）

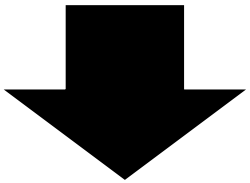
決めたい
パラメータ

EMアルゴリズム

データとして観測されない変数（隠れ変数）を含む

確率モデルのパラメータの値を

観測したデータ（劣化画像）を用いて決定したい！



expectation-maximization (EM) アルゴリズム

Q-関数

ある時刻 t において次の Q-関数 を定義する

$$Q\left(\boldsymbol{\theta}_{t+1} \mid \boldsymbol{\theta}_t, \boldsymbol{y}\right) = \sum_{\boldsymbol{x}} P\left(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}_t\right) \ln P\left(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}_{t+1}\right)$$

事後確率　　結合確率

$\boldsymbol{\theta}_t = \left\{\alpha_t, \sigma_t^2\right\}$ ← パラメータをまとめて記述する

$\boldsymbol{\theta}_t$ を既知とするとこの関数は $\boldsymbol{\theta}_{t+1}$ の関数となる

EMアルゴリズムの流れ

(1) $t=0$ としてパラメータ $\boldsymbol{\theta}_0$ の値を適当に初期化

(2) Q-関数 $Q(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t, \boldsymbol{y})$ を計算する← Eステップ

(3) Q-関数 $Q(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t, \boldsymbol{y})$ を $\boldsymbol{\theta}_{t+1}$ に関して最大化する← Mステップ

$$\boldsymbol{\theta}_{t+1}^{\max} = \arg\max_{\boldsymbol{\theta}_{t+1}} Q(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t, \boldsymbol{y})$$

(4) $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1}^{\max}$ となっていれば $\boldsymbol{\theta}_{t+1}^{\max}$ を結果として出力して終了

そうでなければ

$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t+1}^{\max}$ と値を更新して時刻を一つ進め$(t \leftarrow t+1)$て(2)に戻る

ノイズ除去の場合のEステップ

$$Q\left(\boldsymbol{\theta}_{t+1}\,|\,\boldsymbol{\theta}_t,\boldsymbol{y}\right)=-\frac{1}{2\sigma_{t+1}^2}\sum_{i\in V}\mathrm{E}_{\mathrm{pos}}\left[\left(y_i-v_i\right)^2\right]_t$$
$$-\alpha_{t+1}\sum_{\{i,j\}\in E}\mathrm{E}_{\mathrm{pos}}\left[\left|v_i-v_j\right|\right]_t-\frac{|V|}{2}\ln\left(2\pi\sigma_{t+1}^2\right)-\ln Z\left(\alpha_{t+1}\right)$$

$$\mathrm{E}_{\mathrm{pos}}\left[\cdots\right]_t=\sum_{\boldsymbol{v}}\left(\cdots\right)P\left(\boldsymbol{v}\,|\,\boldsymbol{y},\boldsymbol{\theta}_t\right)\leftarrow\text{事後確率による期待値}$$

$|V|=\left(V\text{の要素の数}\right)$
$=\left(\text{ピクセル数}\right)$

最大化のステップ（Mステップ）のためパラメータに関する勾配を計算する

$$\begin{cases}\dfrac{\partial Q\left(\boldsymbol{\theta}_{t+1}\,|\,\boldsymbol{\theta}_t,\boldsymbol{y}\right)}{\partial\sigma_{t+1}^2}=\dfrac{1}{2\sigma_{t+1}^4}\displaystyle\sum_{i\in V}\mathrm{E}_{\mathrm{pos}}\left[\left(y_i-v_i\right)^2\right]_t-\dfrac{|V|}{2\sigma_{t+1}^2}\\ \dfrac{\partial Q\left(\boldsymbol{\theta}_{t+1}\,|\,\boldsymbol{\theta}_t,\boldsymbol{y}\right)}{\partial\alpha_{t+1}}=-\displaystyle\sum_{\{i,j\}\in E}\mathrm{E}_{\mathrm{pos}}\left[\left|v_i-v_j\right|\right]_t+\sum_{\{i,j\}\in E}\mathrm{E}_{\mathrm{pri}}\left[\left|v_i-v_j\right|\right]_{t+1}\end{cases}$$

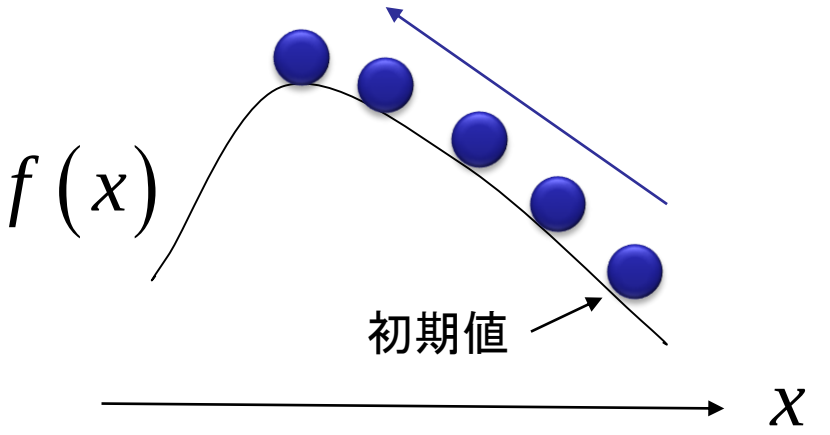
$$\mathrm{E}_{\mathrm{pri}}\left[\cdots\right]_{t+1}=\sum_{\boldsymbol{v}}\left(\cdots\right)P\left(\boldsymbol{v}\,|\,\alpha_{t+1}\right)\leftarrow\text{事前確率による期待値}$$

ノイズ除去の場合のMステップ

$$\begin{cases} \dfrac{\partial Q(\boldsymbol{\theta}_{t+1} \mid \boldsymbol{\theta}_t, \boldsymbol{y})}{\partial \sigma_{t+1}^2} = \dfrac{1}{2\sigma_{t+1}^4} \displaystyle\sum_{i \in V} \mathrm{E}_{\mathrm{pos}} \left[(y_i - v_i)^2 \right]_t - \dfrac{|V|}{2\sigma_{t+1}^2} \\ \dfrac{\partial Q(\boldsymbol{\theta}_{t+1} \mid \boldsymbol{\theta}_t, \boldsymbol{y})}{\partial \alpha_{t+1}} = - \displaystyle\sum_{\{i,j\} \in E} \mathrm{E}_{\mathrm{pos}} \left[|v_i - v_j| \right]_t + \sum_{\{i,j\} \in E} \mathrm{E}_{\mathrm{pri}} \left[|v_i - v_j| \right]_{t+1} \end{cases}$$

各勾配は確率伝搬法（sum-productアルゴリズム）で計算できる

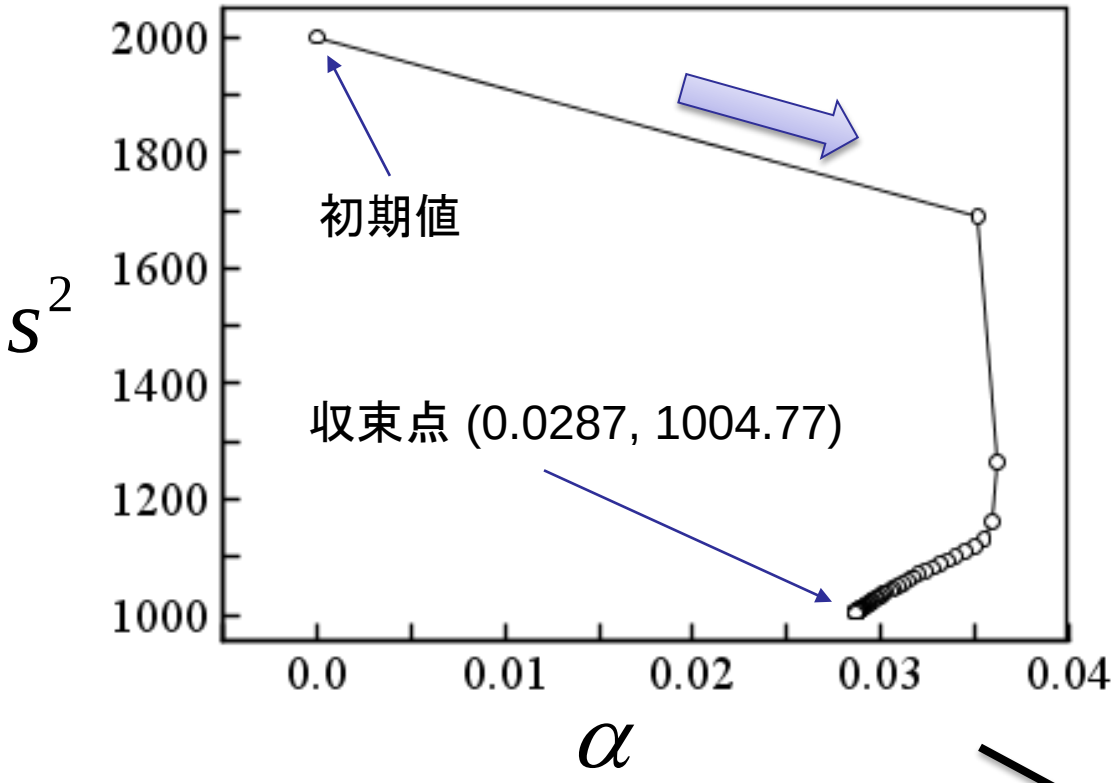
最大化はこれらの勾配を利用した勾配上昇法で実装可能！



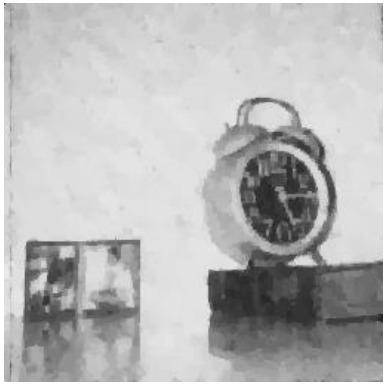
$$x^{\mathrm{new}} \leftarrow x^{\mathrm{old}} + \varepsilon \frac{\partial f\left(x^{\mathrm{old}}\right)}{\partial x}$$

ε：小さな正の数

EMアルゴリズムの実行結果



それなりのパラメータが
EMアルゴリズムにより
推定された



PSNR 26.2 [dB]

画像補修フィルタ

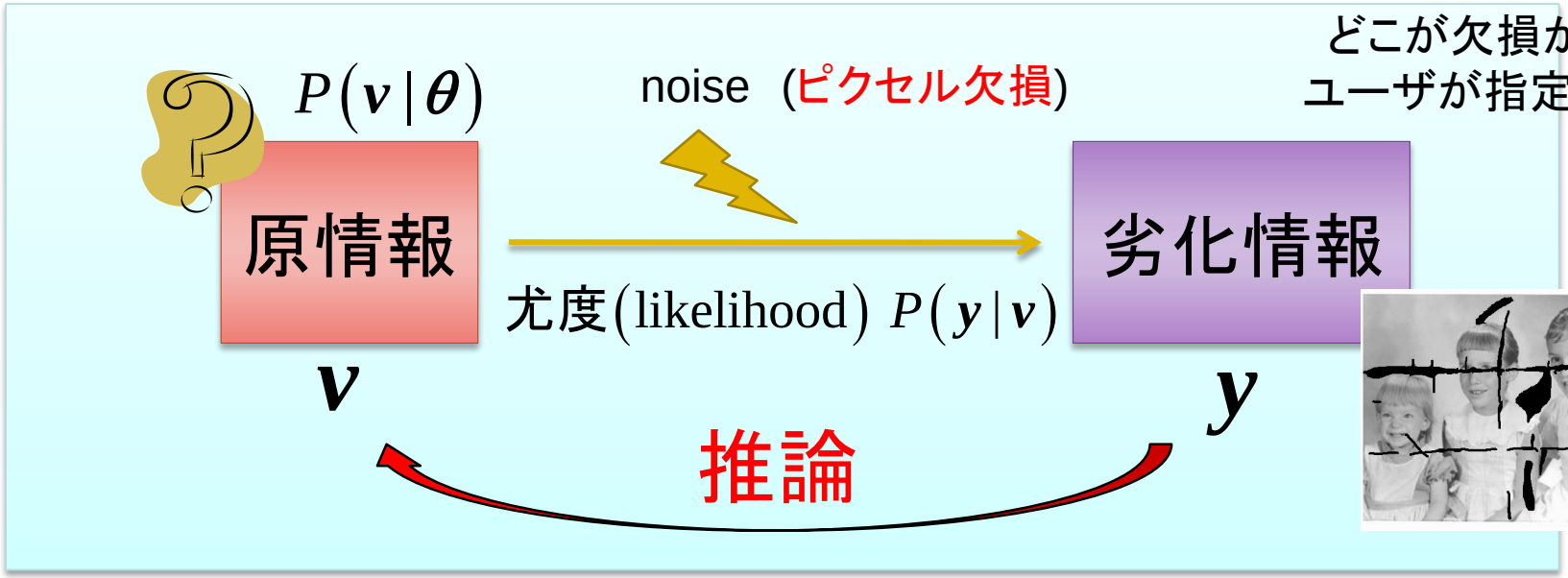
画像補修（Image Inpainting）フィルタ

- 画像中の落書き傷を自動補修するための画像処理フィルタ
- 2000年初頭の提案から現在ではいくつかの商用ソフトに実装されてきている

大切な写真が傷付いてしまった

黒く塗りつぶした傷の部分を治したい

画像補修フィルタの枠組み



欠損した箇所の情報はまったくない
欠損していない箇所は原画像と同じ

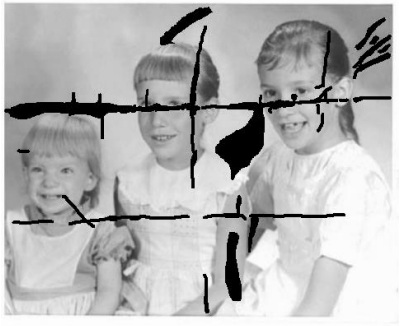
デルタ関数

$$(\text{尤度}) = P(y|v) = \prod_{i \in (\text{非欠損部})} \delta(y_i, v_i)$$

MRFを用いた画像補修フィルタ

大切な写真が傷付いてしまった

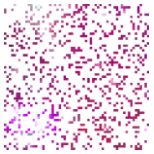
黒く塗りつぶした傷の部分を治したい



治った！！

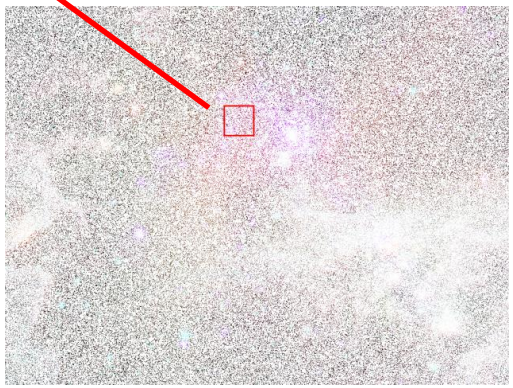
[Yasuda *et. al.*, 05, 13]

MRFを用いた画像補修フィルタ



80%をランダムに欠損させた画像

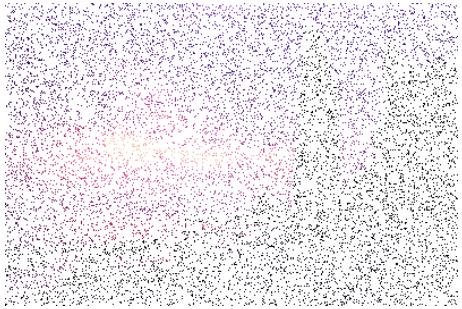
原画像



復元画像

多くの自然画像を MRF で学習して
それを事前分布の近似としたベイズ推定の方法で補修している

MRFを用いた画像補修フィルタ



熊も城も断崖絶壁もなんでもござれ！！

交通流予測

仙台市

交通予測

車両感知器やプローブカーデータ等から得られる各道路の情報から
現在（ナウ・キャスト）または未来（フォア・キャスト）の交通状態を予測すること

問題点

車両感知器・・・主要道路のみにしか設置されていない

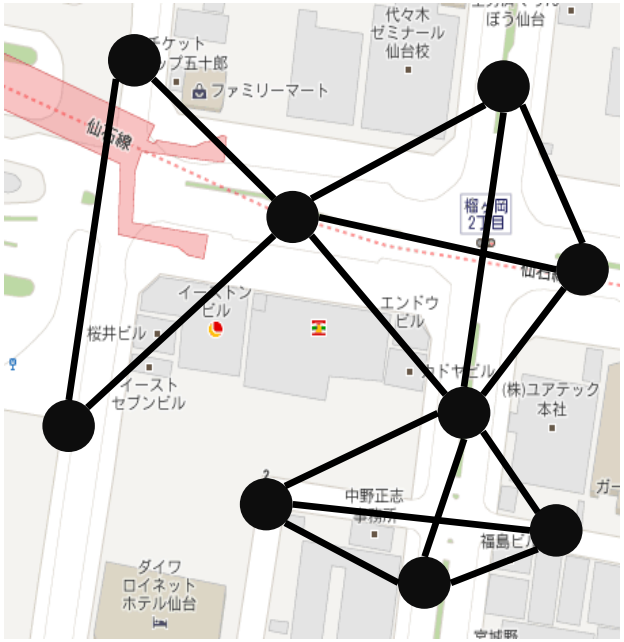
プローブ情報・・・すべての車、道路から情報が得られるわけではない

⇒ 部分的な道路の情報から全体の交通状態を予測したい

道路交通に対するMRF

ノード ・・・ 各道路上に配置 ← 予測対象が道路のため，道路がノードに対応

リンク ・・・ 直接移動可能な道路同士を結ぶ ← 隣接道路間の平滑化性
（一本前の道路が混雑していたら次の道路も混雑しやすいという仮定）



グラフの構造

各道路上に確率変数 v を配置する

確率変数は配置された道路の交通密度を表わす

（交通密度）＝（車の台数）÷（道路面積）

確率変数は任意の実数値をとるとする

$$v_i \in (-\infty, +\infty)$$

道路交通に対するMRF

[Kataoka, Yasuda, Furtlehner, Tanaka, 14]

道路交通に対する事前確率

$$P(\boldsymbol{v} \mid \boldsymbol{b}, \lambda, \alpha) \propto \exp\left(-\frac{\lambda}{2} \sum_{i \in V} v_i^2 + \sum_{i \in V} b_i v_i - \alpha \sum_{\{i,j\} \in E} (v_i - v_j)^2 \right)$$

バイアス項

平滑化項

- 道路毎の個性を表わす
- 潜在的に混みやすい道路とそうでない道路があるであろう

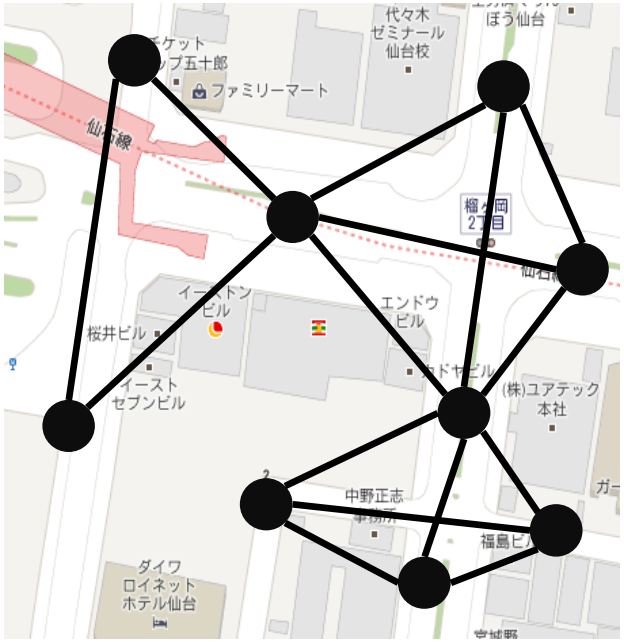
つながった道路同士は似たような混雑状況であろう

以上のような仮定の下で MRF を定義する

MRFを学習する

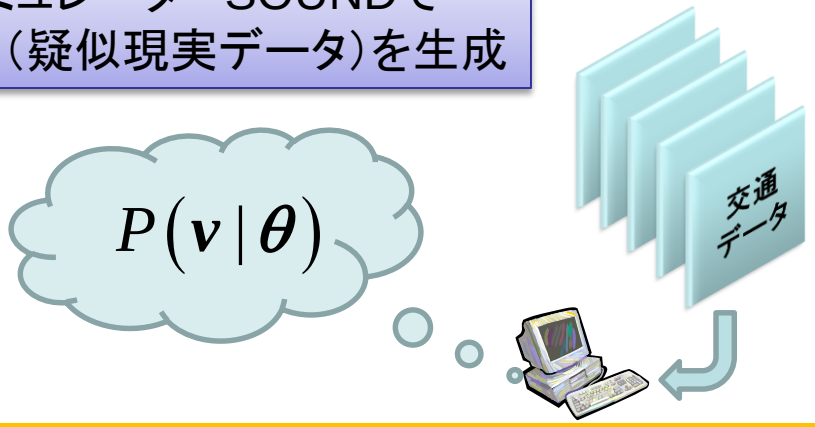
ノード ・・・ 各道路上に配置 ← 予測対象が道路のため，道路がノードに対応

リンク ・・・ 直接移動可能な道路同士を結ぶ ← 隣接道路間の平滑化性
（一本前の道路が混雑していたら次の道路も混雑しやすいという仮定）



グラフの構造

交通シミュレーターSOUNDで
交通データ（疑似現実データ）を生成

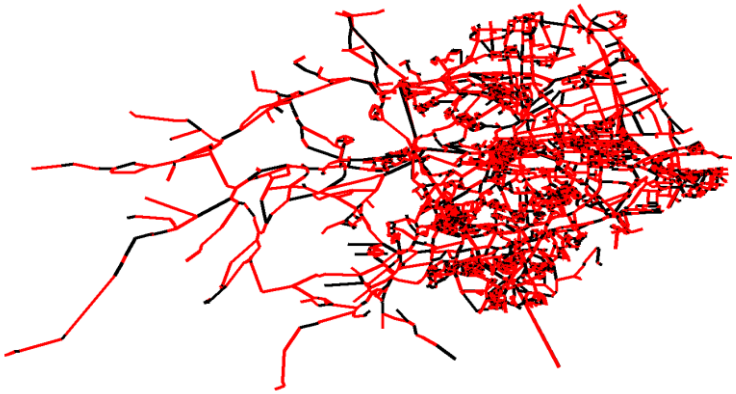


統計的機械学習を利用して
8時～9時の1時間分の交通データからモデルを設計

テストは同時間帯の新しいデータを用いておこなう

SOUND: 株式会社アイ・トランスポートラボ

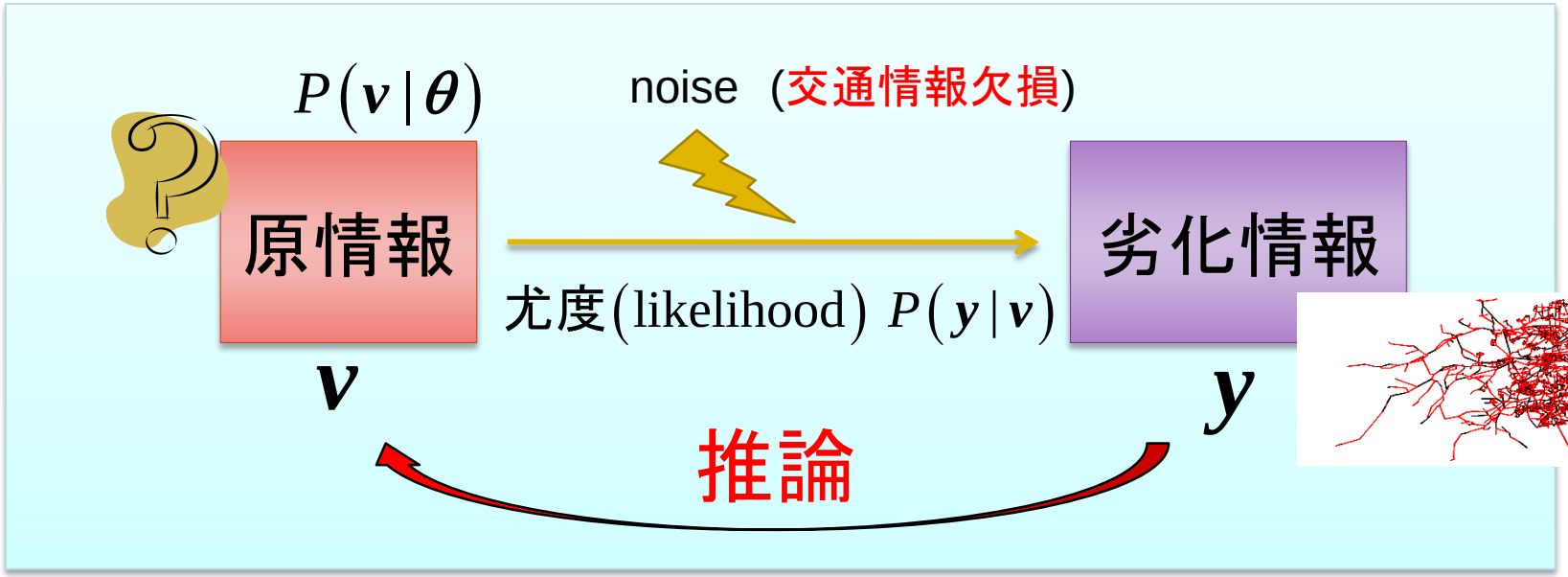
MRFによる交通量予測の枠組み



未観測道路（欠損道路）をランダムに70%選択

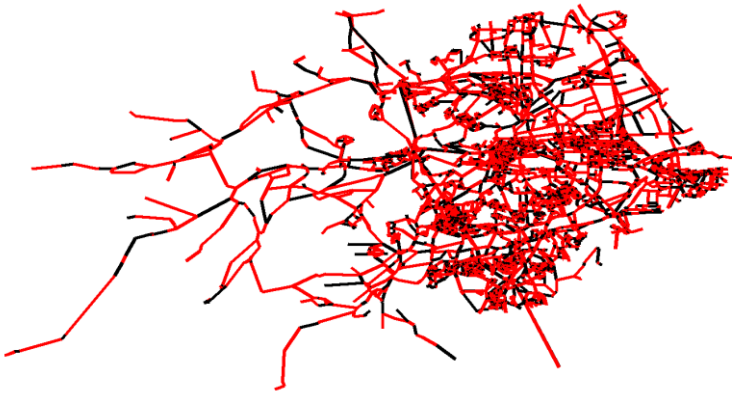
赤 ・・・ 未観測道路

黒 ・・・ 観測道路



枠組みは画像補修とまったく同じ！

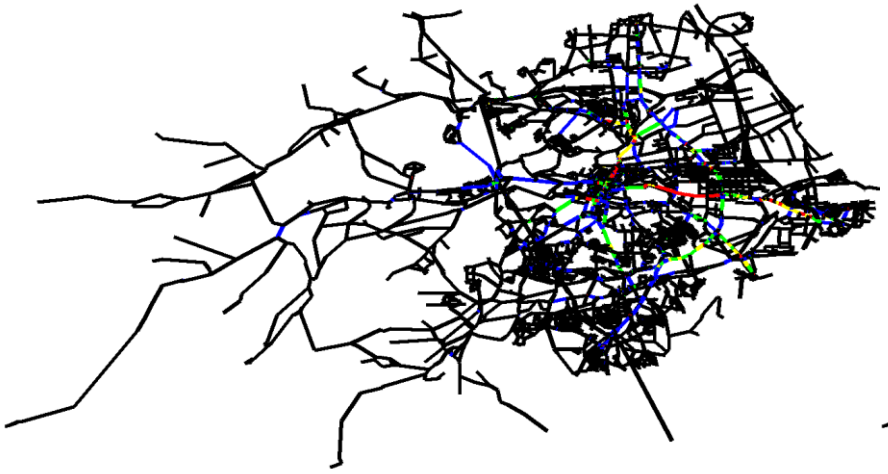
MRFによる交通流予測



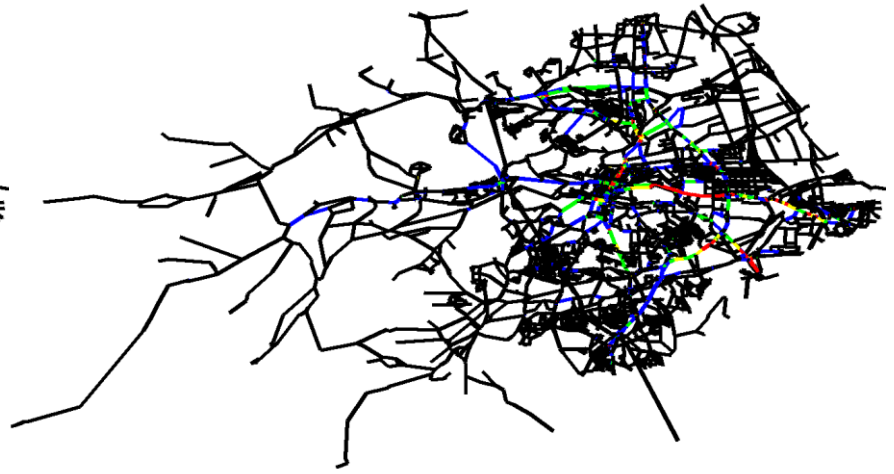
未観測道路（欠損道路）をランダムに70%選択

赤　・・・　未観測道路

黒　・・・　観測道路



正解のデータ



推定結果

黒 ＜ 青 ＜ 緑 ＜ 黄 ＜ 赤

少 ← 交通密度 → 多

これも先の画像の例と同様に
ベイズ推定の枠組みを用いて予測している

MRFによる交通流予測

仙台中心部

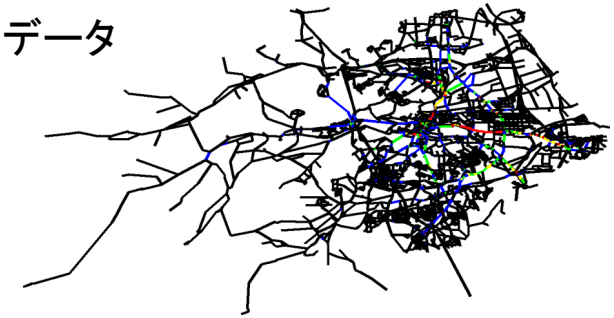
未観測道路　(赤：未観測)

正解のデータ

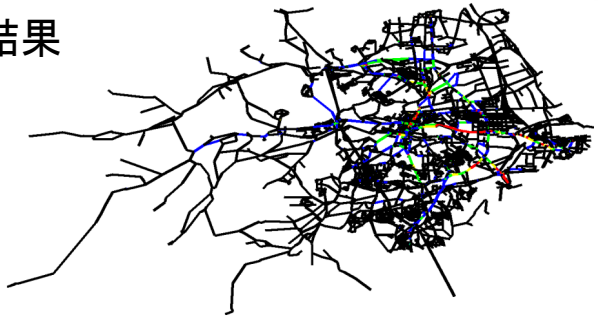
推定結果

MRFによる交通流予測

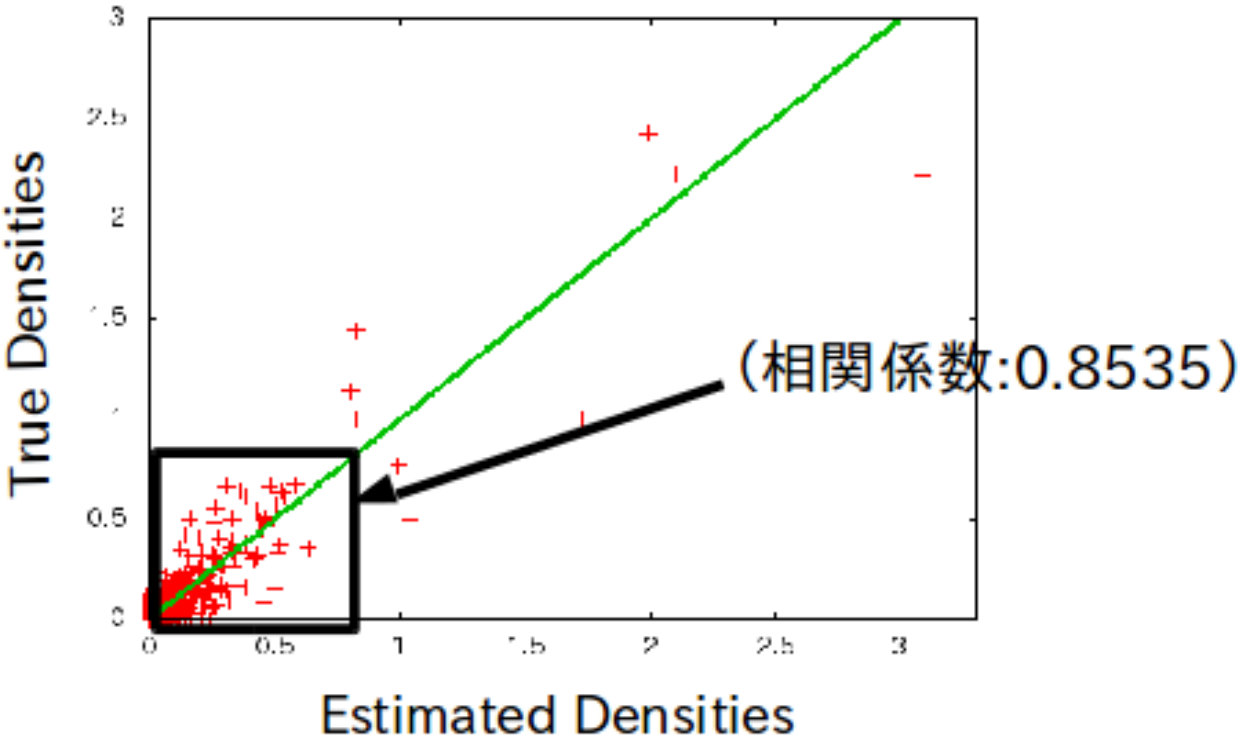
正解のデータ



推定結果

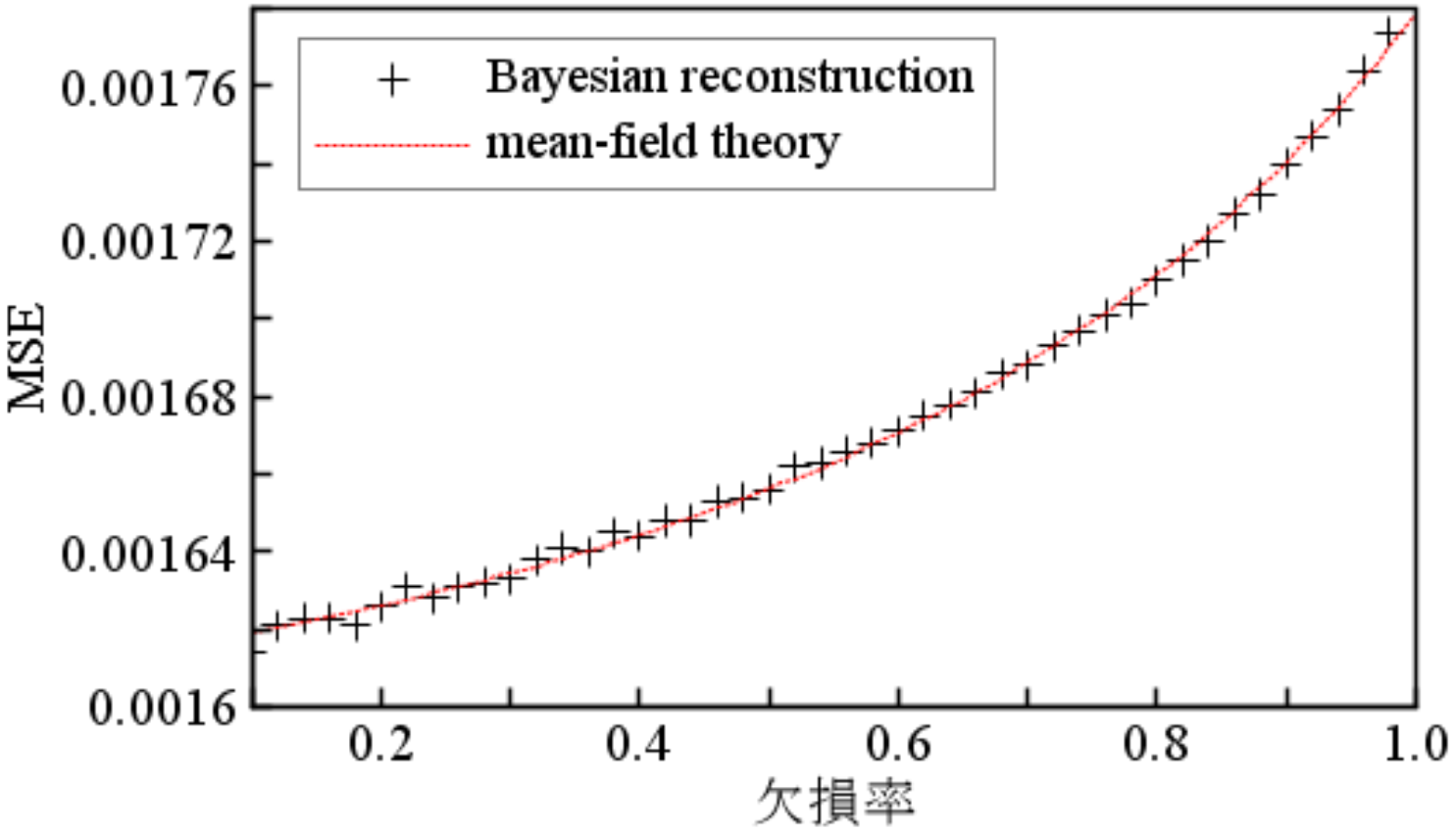


真値と推定値の比較（相関係数:0.9168）



MRFによる交通流予測

欠損率と推定値のMSEの比較



赤の破線・・・平均場解析による理論予測曲線

[Kataoka, Yasuda and Tanaka, 15]

ネットワークマイニングへの応用とその他

ネットワークマイニングへの応用

$$P(\boldsymbol{v} \mid \boldsymbol{b}, \boldsymbol{w}) = \frac{1}{Z(\boldsymbol{b}, \boldsymbol{w})} \exp\left(\sum_{i \in V} b_i v_i + \sum_{\{i,j\} \in E} w_{ij} v_i v_j \right)$$

$$v_i \in \{-1, +1\}$$

確率変数が $\{+1, -1\}$ の 2 値をとるボルツマンマシンを考えてみる

パラメータが作る局所的な構造

- もし $b_i > 0$ ならば局所的には $v_i = +1$ で逆に $b_i < 0$ なら $v_i = -1$ が確率的に高い
- もし $w_{ij} > 0$ ならば v_i と v_j が同じ値の方が局所的には確率が高い
- もし $w_{ij} < 0$ならば v_i と v_j が同じ値でない方が局所的には確率が高い

ネットワークマイニングへの応用

生成モデルのより深い構造理解
（データマイニング）

リンク構造に注目する

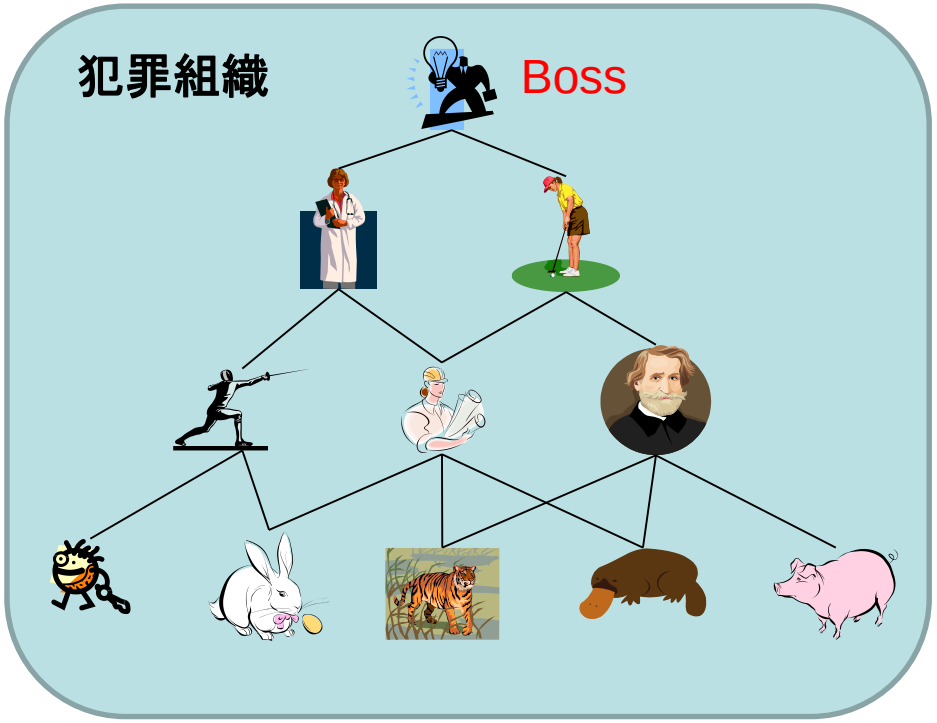
リンクの重みはノードの局所的な関連性を表現している

$$w_{ij}\begin{cases} > 0 : \text{ノード } i \text{ と } j \text{ は局所的に正の相関} \\ = 0 : \text{ノード } i \text{ と } j \text{ は局所的には関連しない} \\ < 0 : \text{ノード } i \text{ と } j \text{ は局所的に負の相関} \end{cases}$$

学習結果の重みを調べることで**データの詳細な構造**がわかる

犯罪組織を調査する

とある犯罪組織を調査する刑事さんがいる



ボスが誰かは判明しているが

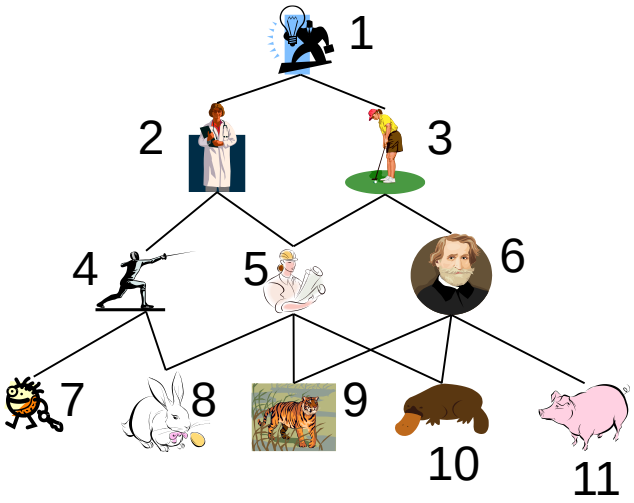
幹部と末端のつながりが分からないので

それを調査している

つまり刑事さんは左の組織図のリンク構造を知らない

つながりは命令系統をあらわしており
リンクでつながった者同士は同時期に活動することが（確率的に）多いとする

犯罪組織の活動モデル



人（？）をノードとし，つながりをリンクとする

それぞれノードは活動中（+1）・休止中（-1) の2値をとるものとする

それぞれのノードは隣接するノードと同じ状態を取りやすいとする

モデリング

$$\begin{cases} C_g(\boldsymbol{v}) = -\sum_{\{i,j\}\in E} v_i v_j \\ P_g(\boldsymbol{v}) = \frac{1}{Z_g}\exp\left(-C_g(\boldsymbol{v})\right) \end{cases}$$

この犯罪組織の活動状況をあらわす真のモデルとする

刑事さんはもちろんこの分布を知らない
唯一知っているのは懸命に調べた組織の活動記録のみ

活動記録からつながりを推定

組織の活動モデル

$$P_{\mathrm{g}}(\boldsymbol{v}) = \frac{1}{Z_{\mathrm{g}}} \exp\left(\sum_{\{i,j\} \in E} v_i v_j \right)$$

10000 データ点

MCMCで生成

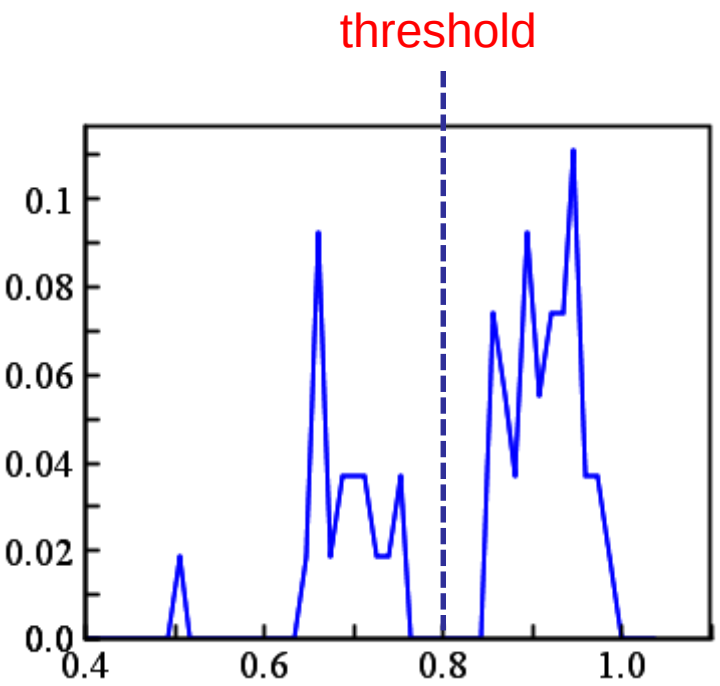
-1,1,1,1,-1,-1,1, ･･･,-1,1 $\mathbf{d}^{(1)}$

1,-1,-1,1,1,-1,1, ･･･,-1,-1 $\mathbf{d}^{(2)}$

⋮

-1,1,1,1,1,-1,1, ･･･,-1,-1 $\mathbf{d}^{(10000)}$

組織の活動記録



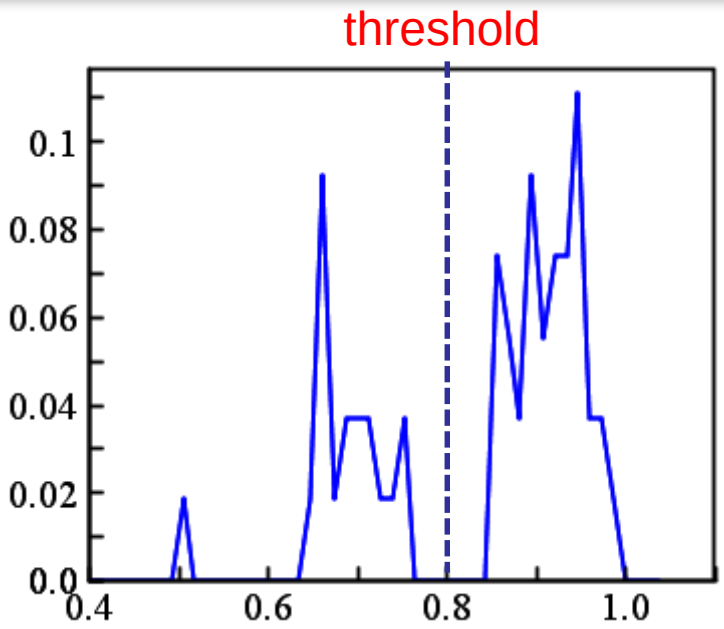
すべてのノードペアの**相関係数**のヒストグラム

単純な例として

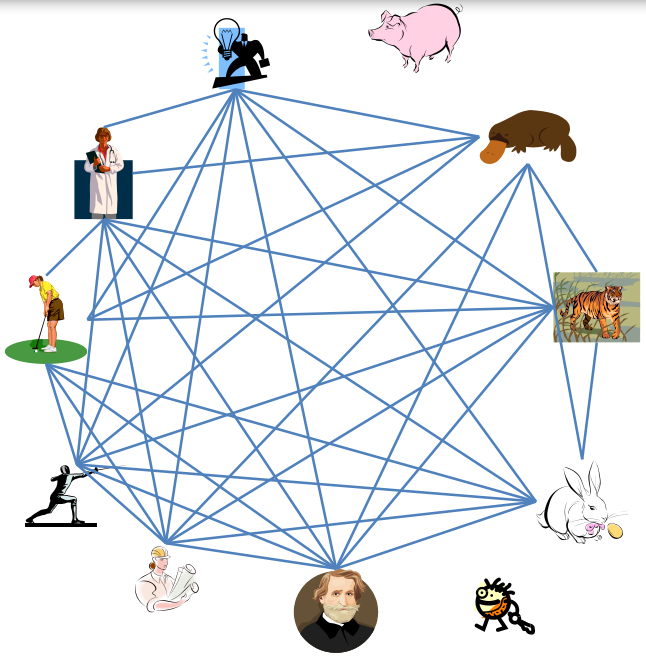
データから**相関係数**を計算し

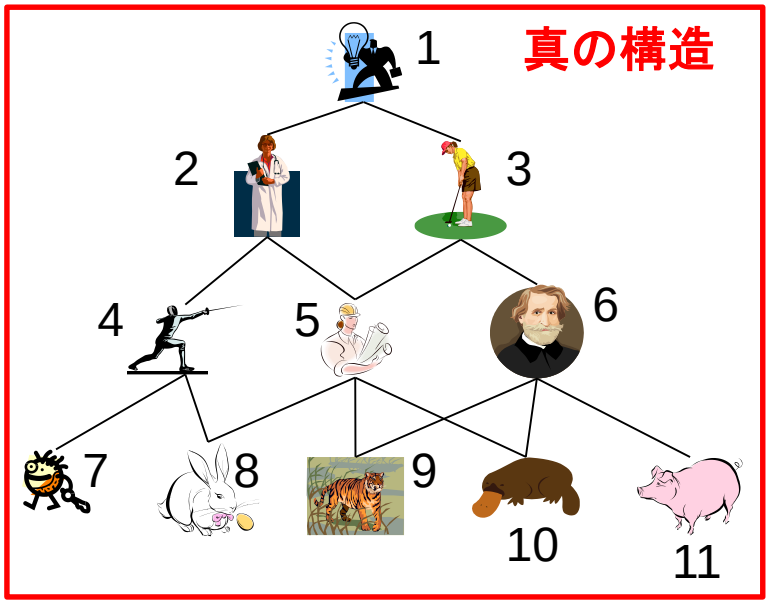
0.8以上の値をもつペアのみに注目する

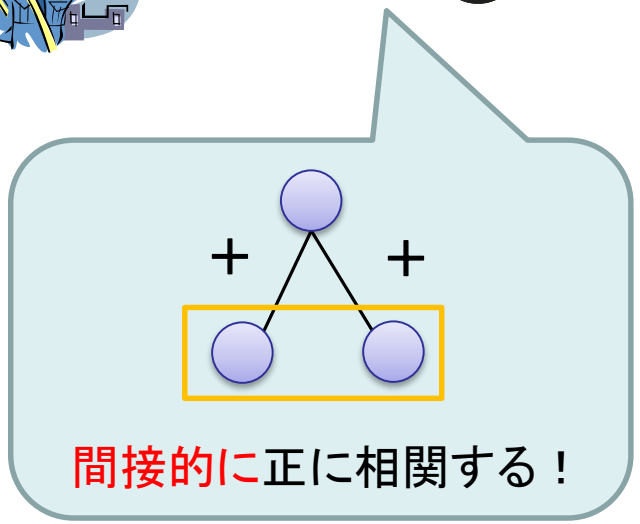
単純な方法では大失敗



推定







単純な相関係数ではこのような間接的な相関も検知してしまう

統計的機械学習で推定してみる

ボルツマンマシンによる推定

-1,1,1,1,-1,-1,1, ･･･,-1,1 $\mathbf{d}^{(1)}$

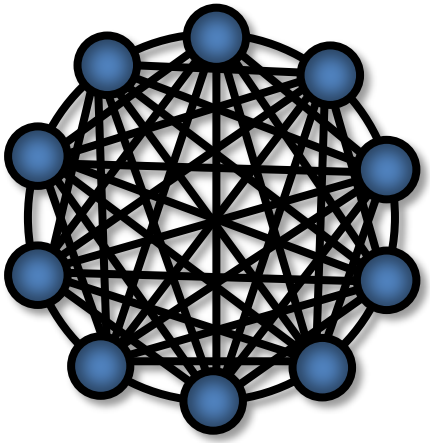
1,-1,-1,1,1,-1,1, ･･･,-1,-1 $\mathbf{d}^{(2)}$

⋮

-1,1,1,1,1,-1,1, ･･･,-1,-1 $\mathbf{d}^{(10000)}$

学習

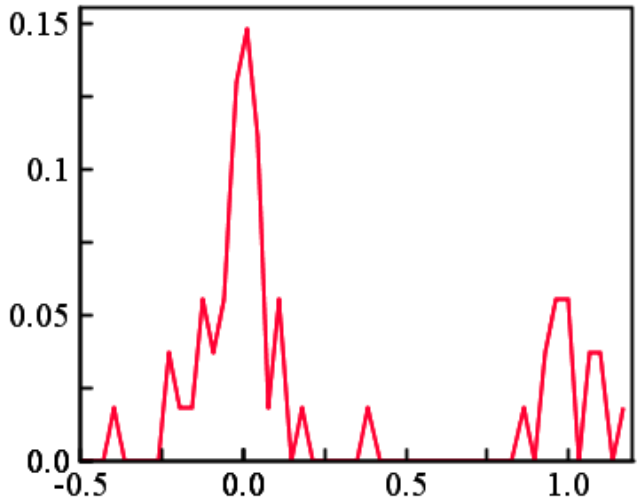
完全結合のボルツマンマシン



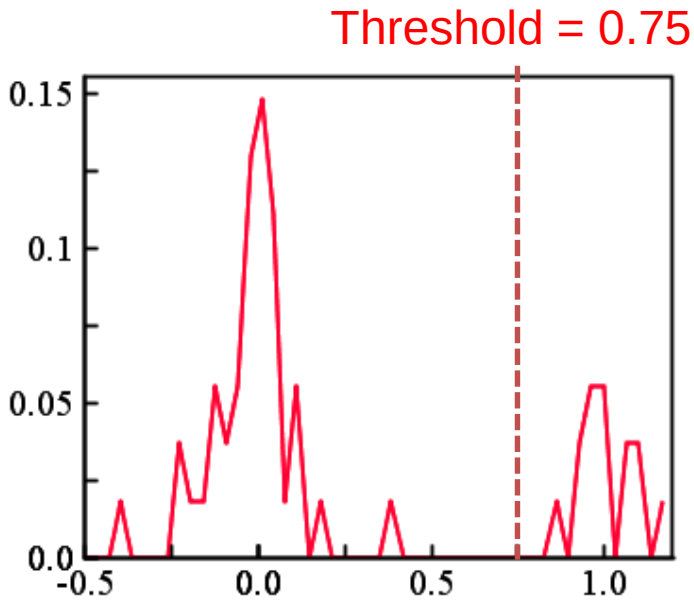
学習により得られた

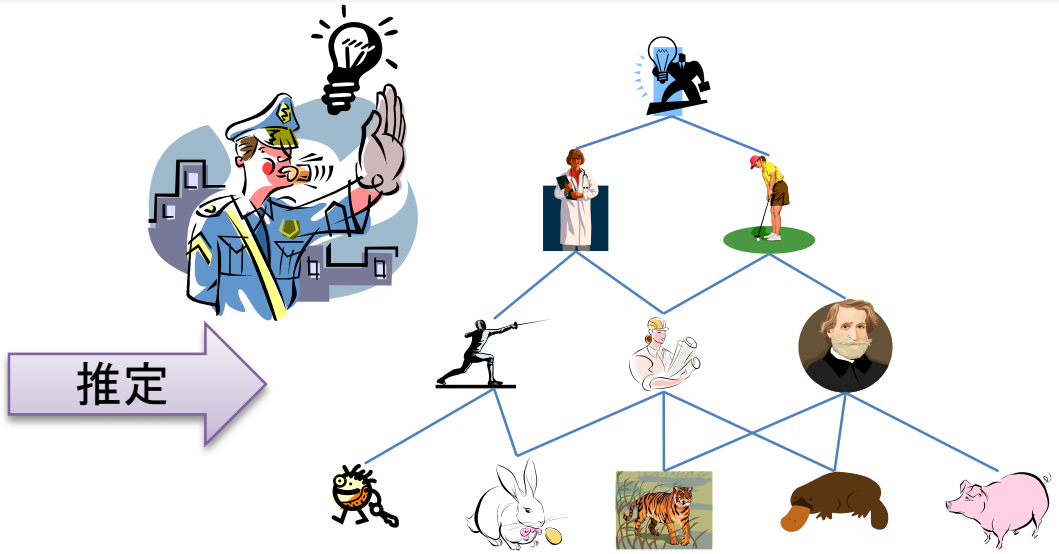
すべてのリンク重み w の

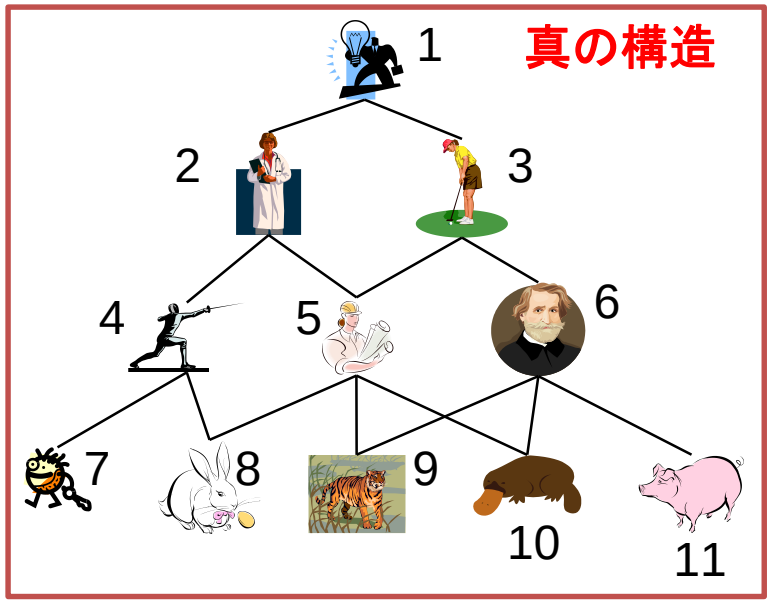
ヒストグラム



ネットワーク構造の抽出に成功







- 組織の詳細な構造の抽出に成功
- モデル自体が間接的な関係性も考慮にいれているため相関係数を用いたときのような問題は起こりづらい
- 多少恣意的な閾値処理が入っているがここをある程度自動化する理論（正則化理論）が存在する

ネットワーク構造の抽出に成功

もちろんこの例は生成モデルと
学習モデルが等しいので
ある意味チートである

しかし重要なことは
MRF的にデータが生成されている
場合は
MRFでないと正確に構造抽出が難しい
ということである

MRFでも入出力関係を作ることができる

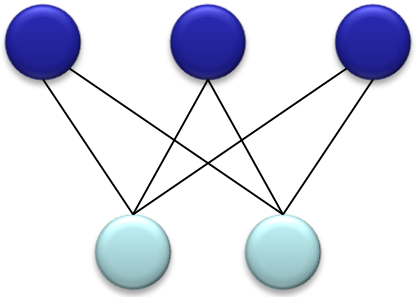
例えばニューラルネットワークは

入力データと出力データのセット（訓練セット）

を学習して入力から出力をだすシステムを構成できる

MRFでも入出力関係を作ることができる

例えば2部グラフ上の MRF を考えてみる



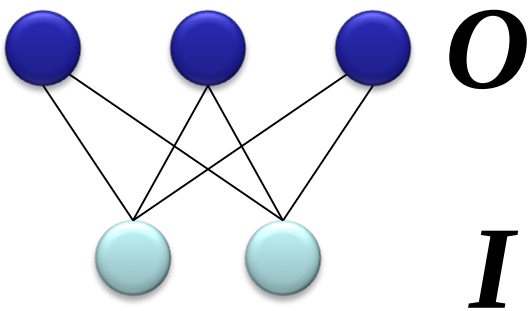
下層のノードを入力ノードとし上層を出力ノードと決めてしまう

入出力関係のデータがそろっていれば
それを使ってこの MRF を学習することができる
（MRF 自身はどれが入力でどれが出力なのかは区別していない
人間が勝手に入出力を分けているだけ）

MRFが学習するのは入出力関係の確率分布となる

MRFでも入出力関係を作ることができる

例えば2部グラフ上の MRF を考えてみる



便宜的に下層（入力）の変数を I で上層（出力）の変数を O で表す

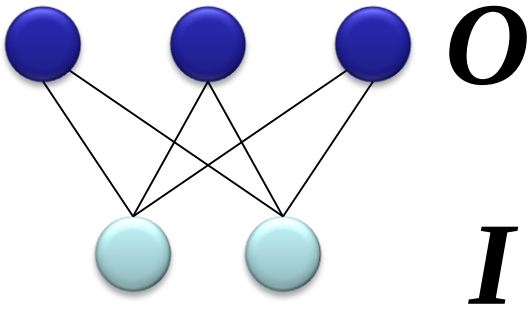
学習される MRF はもちろん I と O の結合確率

$$P(I,O) = \frac{1}{Z}\exp\left(-C(I,O)\right)$$

MRFでも入出力関係を作ることができる

例えば2部グラフ上の MRF を考えてみる

$$P(\boldsymbol{I},\boldsymbol{O}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{I},\boldsymbol{O})\right)$$



2部グラフだと
結構簡単に計算できる！

条件付き確率を計算する

$$P(\text{出力}\,|\,\text{入力}) = P(\boldsymbol{O}\,|\,\boldsymbol{I})$$

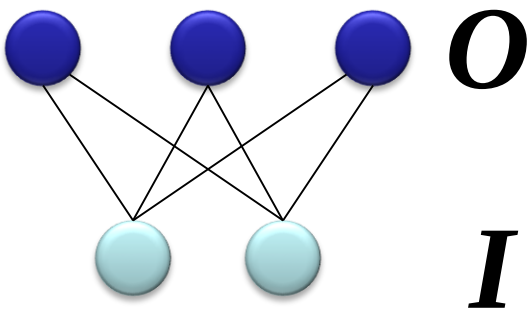
これはある入力 $\boldsymbol{I}$ がやって来たときの

出力 $\boldsymbol{O}$ の確率として解釈できる

MRFでも入出力関係を作ることができる

例えば2部グラフ上の MRF を考えてみる

$$P(\boldsymbol{I},\boldsymbol{O}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{I},\boldsymbol{O})\right)$$



条件付き確率を計算する

$$P(\boldsymbol{O}\,|\,\boldsymbol{I})$$

2部グラフだと
結構簡単に計算できる！

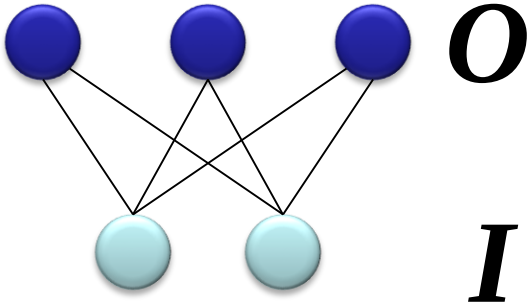
この条件付き確率を最大にする $\boldsymbol{O}$ が
出力となるべきであろう

もちろん確率なので尤もあり得そうな出力だけでなくそこからの分散等も評価できる

MRFでも入出力関係を作ることができる

例えば2部グラフ上の MRF を考えてみる

$$P(\boldsymbol{I},\boldsymbol{O}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{I},\boldsymbol{O})\right)$$



2部グラフだと
結構簡単に計算できる！

逆の条件付き確率を計算する

$$P(\text{入力}|\text{出力}) = P(\boldsymbol{I}|\boldsymbol{O})$$

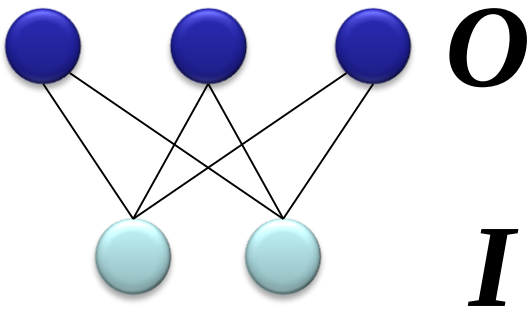
MRF にはそもそも方向性がないので
逆に出力から入力を推定することも可能である

通常のニューラルネットワークではこれは単一では出来ない

生活習慣病推定システム

[Asari *et. al.*, Master Thesis,14]

$$P(\boldsymbol{I},\boldsymbol{O}) = \frac{1}{Z}\exp\left(-C(\boldsymbol{I},\boldsymbol{O})\right)$$



入力に問診項目（体重・運動・飲酒・喫煙・歯磨きの頻度　等）

出力に検査結果（BMI・血圧・コレステロール・メタボリック　等）

問診項目（生活習慣）から体内異常を推定するシステムができる

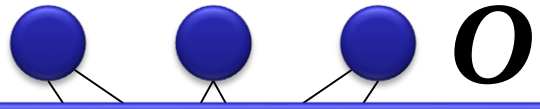
宮城県（2008年）の13209人分の人間ドックのデータで学習

ロジスティック回帰システムを上回る推定性能を実現

特に外れ値に対する推定に優れる

生活習慣病推定システム

[Asari *et. al.*, Master Thesis,14]



体内異常から逆に普段の生活習慣を
推定することも可能である

これは単一のニューラルネットワーク
では容易に実現できない

ロジスティック回帰システムを上回る推定性能を実現

特に外れ値に対する推定に優れる

制限ボルツマンマシン
(restricted Boltzmann machine; RBM)

隠れ変数があるボルツマンマシン

これまで**すべての変数に対応した観測データ**がそろっていることが
統計的機械学習において前提となっていた

データ数よりも多い数の変数をもった
ボルツマンマシンを考える

全ての変数のうち
対応する観測データが得られた変数を $\boldsymbol{v}$
得られない変数を $\boldsymbol{h}$ と記述する

$$P(\boldsymbol{v}|\boldsymbol{\theta}) \rightarrow P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})$$

隠れ変数があるボルツマンマシン

データに対応した変数 v
→ 可視変数（観測変数）

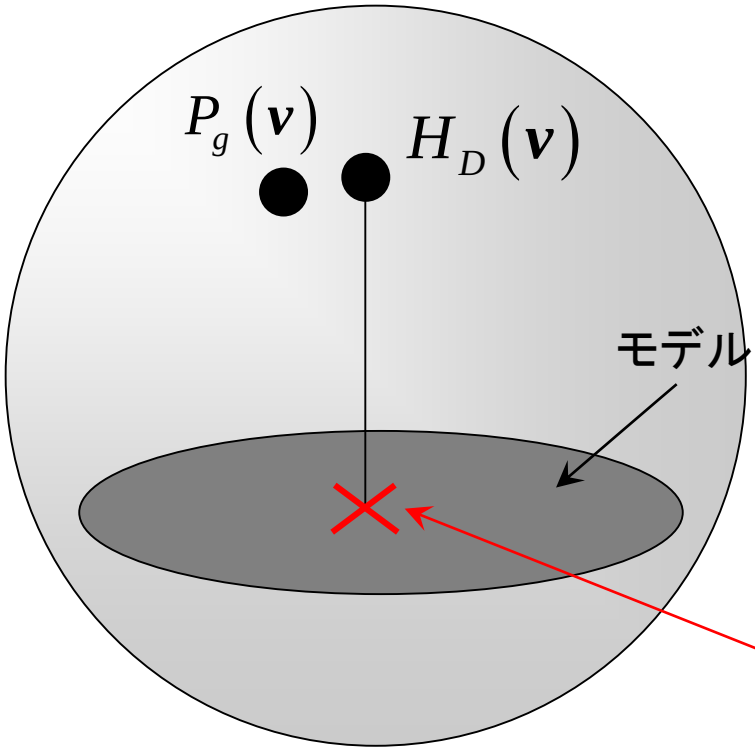
対応するデータがない変数 h
→ 隠れ変数（潜在変数）

隠れ変数はシステムの内部変数となる

$$P(v|\theta) \rightarrow P(v,h|\theta)$$

隠れ変数を導入する意味

隠れ変数を追加することによりモデルの表現能力が増加する



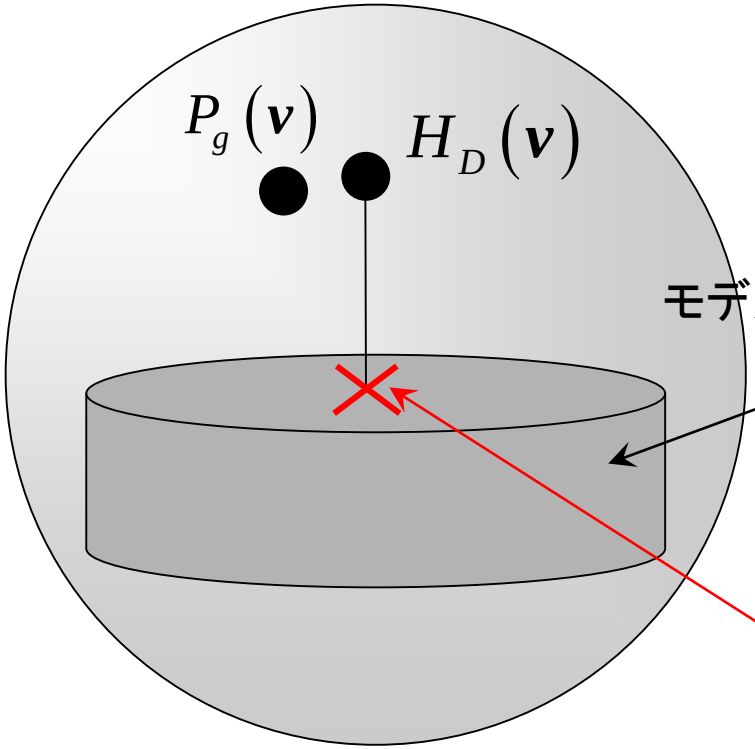
モデルがパラメータの変化によって動ける空間
（隠れ変数なし）

学習で求まるモデル

隠れ変数を導入する意味

隠れ変数を追加することによりモデルの表現能力が増加する

※ 隠れ変数の数がより多くなると一般により表現能力が増す
現に後述の RBM は十分数の隠れ変数により任意の確率分布が表現できることが知られている



モデルがパラメータの変化によって動ける空間
（隠れ変数追加）

学習で求まるモデル

動ける空間が広がるので真の確率により近いモデルが得られる

隠れ変数を導入する意味

隠れ変数を追加することによりモデルの表現能力が増加する

隠れ変数を導入すると確かに分布の表現力が上昇するが

実は良いことばかりではない

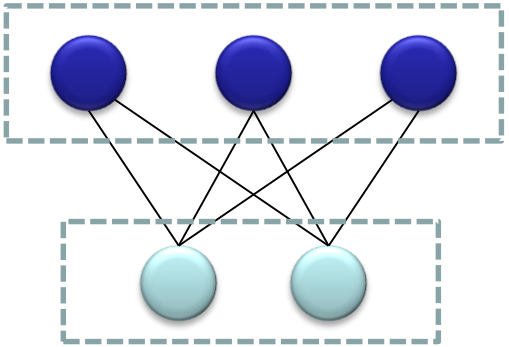
後述のように隠れ変数がある場合も
基本的に最尤法で学習することになる

隠れ変数があると学習の目的関数（対数尤度関数）が
一般に凹関数でなくなるため
最適解を見つけることが隠れ変数がない場合に比べ
格段に困難になる

動ける空間が広がるので真の確率により近いモデルが得られる

制限ボルツマンマシン [Smolensky, 86; Hinton, 02]

- RBM は完全2部グラフ上に定義された2層構造のボルツマンマシン
- 片方の層は可視層であり，もう片方の層は隠れ層である
- RBM は深層学習の基本単位



Hidden layer H $\quad \boldsymbol{h} = \{h_j \in \{0,1\} \mid j \in H\}$

Visible layer V $\quad \boldsymbol{v} = \{v_i \in \{0,1\} \mid i \in V\}$

可視層は可視変数のみで構成され

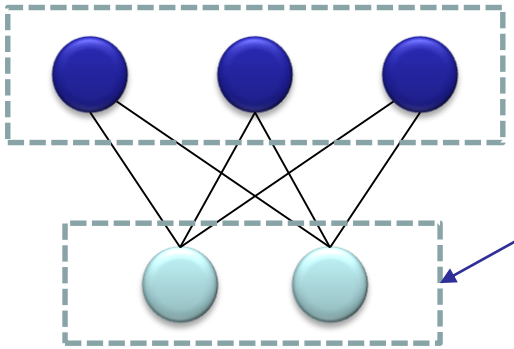
隠れ層は隠れ変数のみで構成される

層内結合は存在しない

これまで V はすべてのノードの集合として使用してきたが

今後は可視変数に対応したノードの集合として用いる

制限ボルツマンマシン



データに対応した変数は可視変数なので
データはこの層の変数にのみ対応する
（可視変数の個数はデータの要素数と同数）

$$v_i, h_j \in \{0,1\}$$

RBMのエネルギー関数

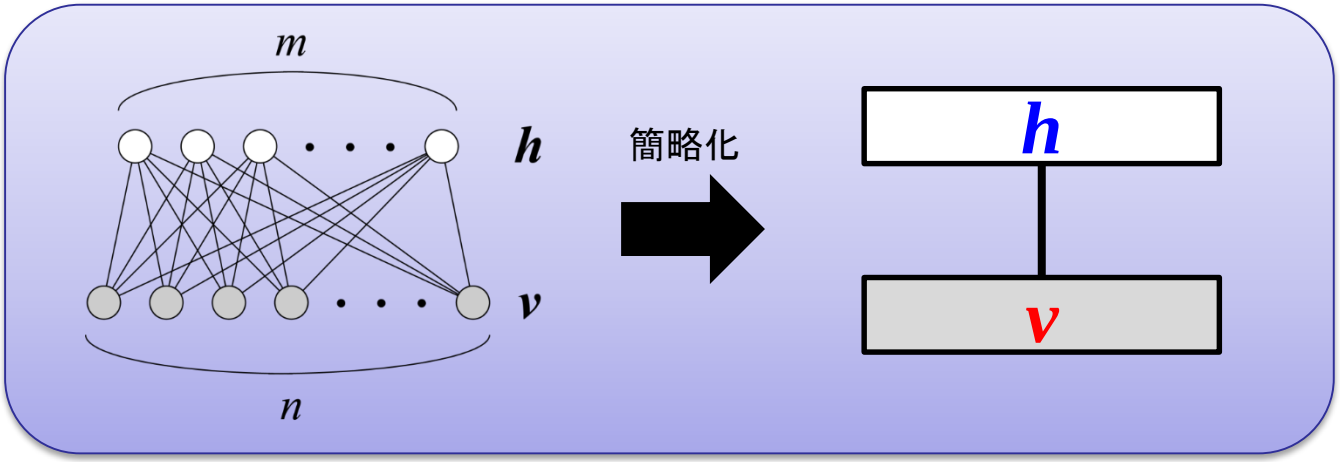
$$C(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) = -\sum_{i\in V} b_i v_i - \sum_{j\in H} c_j h_j - \sum_{i\in V}\sum_{j\in H} w_{ij} v_i h_j$$

バイアスパラメータ

結合パラメータ

$$P(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\left(-C(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})\right)$$

RBMのギブスサンプリング



片方の層の変数を条件としたもう片方の層の条件付き確率

$$P(\boldsymbol{v}|\boldsymbol{h}) = \frac{P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})}{\sum_{\boldsymbol{v}} P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} = \prod_{i\in V} \frac{\exp(\lambda_i v_i)}{1+\exp(\lambda_i)}$$

$$P(\boldsymbol{h}|\boldsymbol{v}) = \frac{P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})}{\sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} = \prod_{j\in H} \frac{\exp(\lambda_j h_j)}{1+\exp(\lambda_j)}$$

$$\lambda_i = b_i + \sum_{j\in H} w_{ij} h_j, \quad \lambda_j = c_j + \sum_{i\in V} w_{ij} v_i$$

RBMのギブスサンプリング

条件付き確率が各変数ごとに因数分解されえた形になっている
→ 各変数が統計的に独立

条件付き独立の性質

片方の層の変数を条件としたもう片方の層の条件付き確率

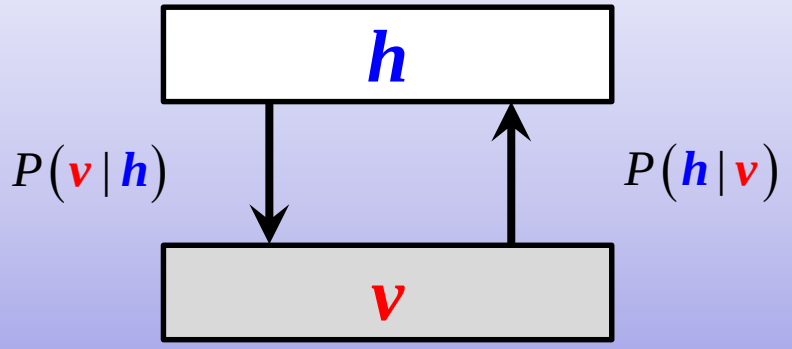
$$P(\boldsymbol{v}|\boldsymbol{h}) = \frac{P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})}{\sum_{\boldsymbol{v}} P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} = \prod_{i\in V} \frac{\exp(\lambda_i v_i)}{1+\exp(\lambda_i)}$$

$$P(\boldsymbol{h}|\boldsymbol{v}) = \frac{P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})}{\sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} = \prod_{j\in H} \frac{\exp(\lambda_j h_j)}{1+\exp(\lambda_j)}$$

$$\lambda_i = b_i + \sum_{j\in H} w_{ij} h_j, \quad \lambda_j = c_j + \sum_{i\in V} w_{ij} v_i$$

RBMのギブスサンプリング

各変数が統計的に独立なので
（他の変数の値を気にせず）各変数ごとに独立にサンプリングが可能



条件付き独立の性質より

片方の層からもう片方の層のサンプリング
がとても簡単

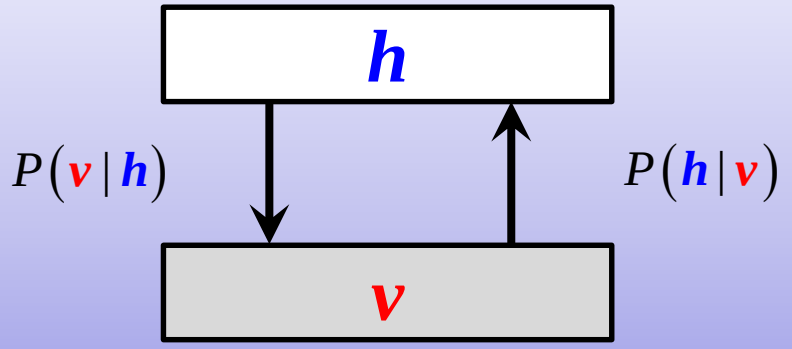
※統計的に独立でないと他の変数の値が確率に影響を与えるためサンプリングが面倒

RBMのギブスサンプリングは
通常の BM のギブスサンプリングと違って
層の変数の値をいっぺんに更新できる

ブロック化ギブスサンプリング

RBMのギブスサンプリング

各変数が統計的に独立なので
（他の変数の値を気にせず）各変数ごとに独立にサンプリングが可能



条件付き独立の性質より

片方の層からもう片方の層のサンプリング
がとても簡単

※統計的に独立でないと他の変数の値が確率に影響を与えるためサンプリングが面倒

層間のギブスサンプリングが容易であるというこの性質は
RBM の学習法であるコントラスティブ・ダイバージェンス
と深層学習に大きく影響している

RBMの学習
と
コントラスティブ・
ダイバージェンス

制限ボルツマンマシンの学習

最尤推定

尤度関数（＝学習モデルがデータセットを実際に生成する確率）

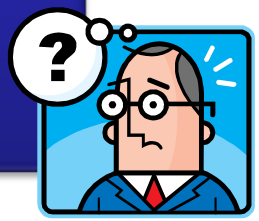
$$L_D(\boldsymbol{\theta}) = \prod_{\mu=1}^{N} P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \boldsymbol{\theta}\right)$$

を最大化

データを可視変数に代入した確率

対応するデータをもたない隠れ変数が

存在する場合はどうしたらいい？？



制限ボルツマンマシンの学習

周辺化して可視変数のみの確率分布に変換する

$$P(\boldsymbol{v}|\boldsymbol{\theta}) = \sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})$$

周辺化により得られた可視変数のみの確率分布を使って尤度関数を構成する

$$L_D(\boldsymbol{\theta}) = \prod_{\mu=1}^{N} P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} | \boldsymbol{\theta}\right)$$

隠れ変数が存在しても周辺化を経由することで最尤推定が使える！

制限ボルツマンマシンの学習

対数尤度関数を定義してそれぞれのパラメータに関する勾配を計算する

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial b_i} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} - \mathrm{E}_{\mathrm{RBM}}\left[v_i \mid \boldsymbol{\theta}\right]$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial c_j} = \frac{1}{N}\sum_{\mu=1}^{N} \sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \mathrm{E}_{\mathrm{RBM}}\left[h_j \mid \boldsymbol{\theta}\right]$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial w_{ij}} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)}\sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \mathrm{E}_{\mathrm{RBM}}\left[v_i h_j \mid \boldsymbol{\theta}\right]$$

$$\sigma_1(x) = \frac{1}{1+e^{-x}}$$

$$\mathrm{E}_{\mathrm{RBM}}\left[(\cdots)\mid\boldsymbol{\theta}\right] = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}(\cdots)P(\boldsymbol{v},\boldsymbol{h}\mid\boldsymbol{\theta})$$

RBM の期待値

制限ボルツマンマシンの学習

RBM の学習は下記の勾配を用いた勾配上昇法によって実行されることとなる

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial b_i} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} - \mathrm{E}_{\mathrm{RBM}}\left[v_i \mid \boldsymbol{\theta}\right]$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial c_j} = \frac{1}{N}\sum_{\mu=1}^{N} \sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \mathrm{E}_{\mathrm{RBM}}\left[h_j \mid \boldsymbol{\theta}\right]$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial w_{ij}} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)}\sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \mathrm{E}_{\mathrm{RBM}}\left[v_i h_j \mid \boldsymbol{\theta}\right]$$

各勾配の第１項はデータとパラメータから直ちに計算できるが

第２項は RBM の期待値であるため組み合わせ爆発の問題が起こる

※ 第１項が計算できるのはRBMの２部グラフ構造のおかげである
一般の構造の隠れ変数あり BM の場合は第１項の計算においても組み合わせ爆発の問題をもつ

RBM の学習もやはり近似学習法を必要とする

コントラスティブ・ダイバージェンス

ギブスサンプリングを基礎とした確率的近似学習法

コントラスティブダイバージェンス(contrastive divergence; CD)

は実装が非常に容易で性能もそれなりに良いので広く普及している

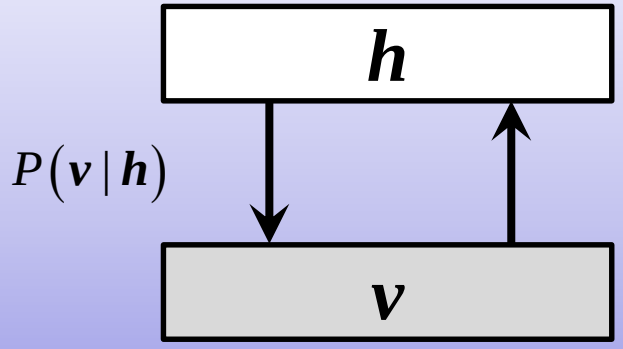
RBM から深層学習へつながるための**立役者的アルゴリズム**

より性能の良いアルゴリズムも多数提案されてきているが

計算コスト等の兼ね合いも含めて総合的に見ると

やはりまだ CD がバランス良いと思われる(私見)

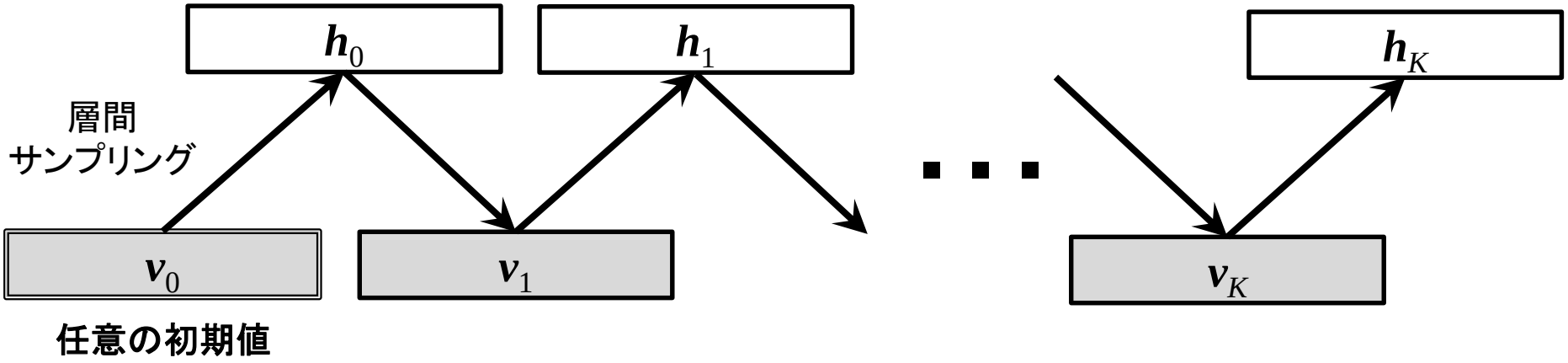
ギブスサンプリング再考



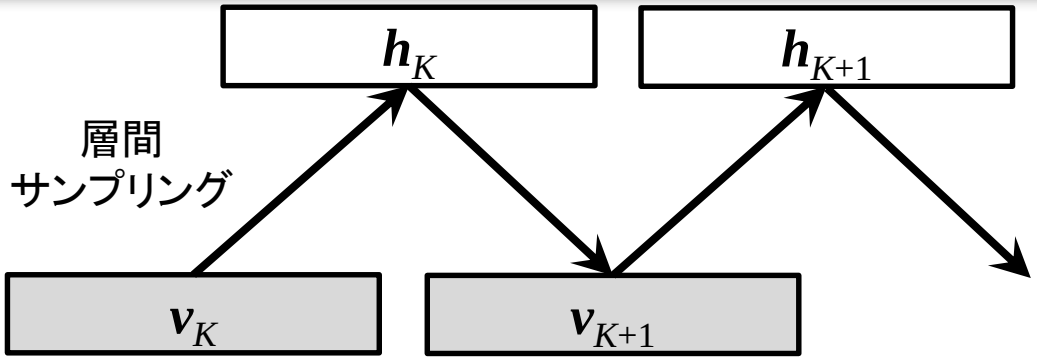
条件付き独立の性質より

片方の層からもう片方の層のサンプリング
がとても簡単

可視変数の任意の初期値から出発して
層間サンプリングを十分回繰り返す（緩和期間）



ギブスサンプリング再考



十分回層間サンプリング
を行った後に
一定間隔でサンプル点
を取得していく

可視変数と隠れ変数に対応するサンプル点をそれぞれ S 点取得したとする

$$\left\{\mathbf{v}^{(1)},\mathbf{h}^{(1)}\right\},\left\{\mathbf{v}^{(2)},\mathbf{h}^{(2)}\right\},\ldots,\left\{\mathbf{v}^{(S)},\mathbf{h}^{(S)}\right\}$$

得られたサンプル点の標本平均で RBM の期待値は近似される
（モンテカルロ積分）

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i \mid \boldsymbol{\theta}\right] \approx \frac{1}{S}\sum_{r=1}^{S} \mathrm{v}_i^{(r)}, \quad \mathrm{E}_{\mathrm{RBM}}\left[h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{S}\sum_{r=1}^{S} \mathrm{h}_j^{(r)},$$

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{S}\sum_{r=1}^{S} \mathrm{v}_i^{(r)} \mathrm{h}_j^{(r)}$$

$S \to \infty$ のときは大数の法則で厳密

ギブスサンプリング再考

h_K h_{K+1}

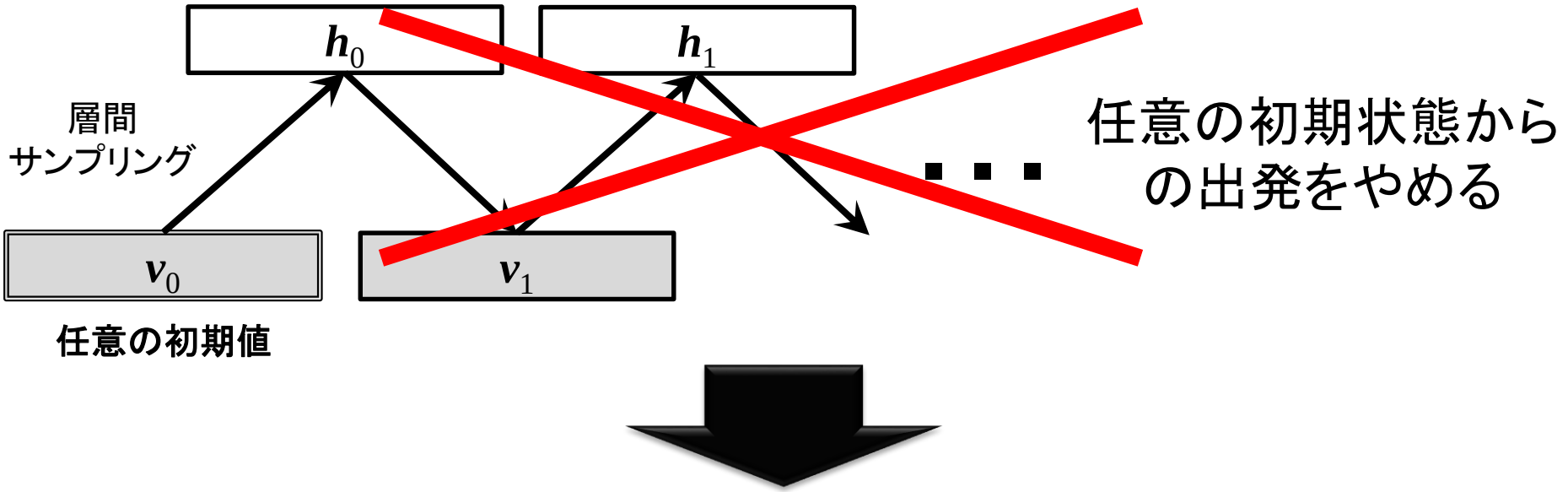
原理的にはモンテカルロ積分でRBMの期待値を
近似して勾配を計算すればよいのだが

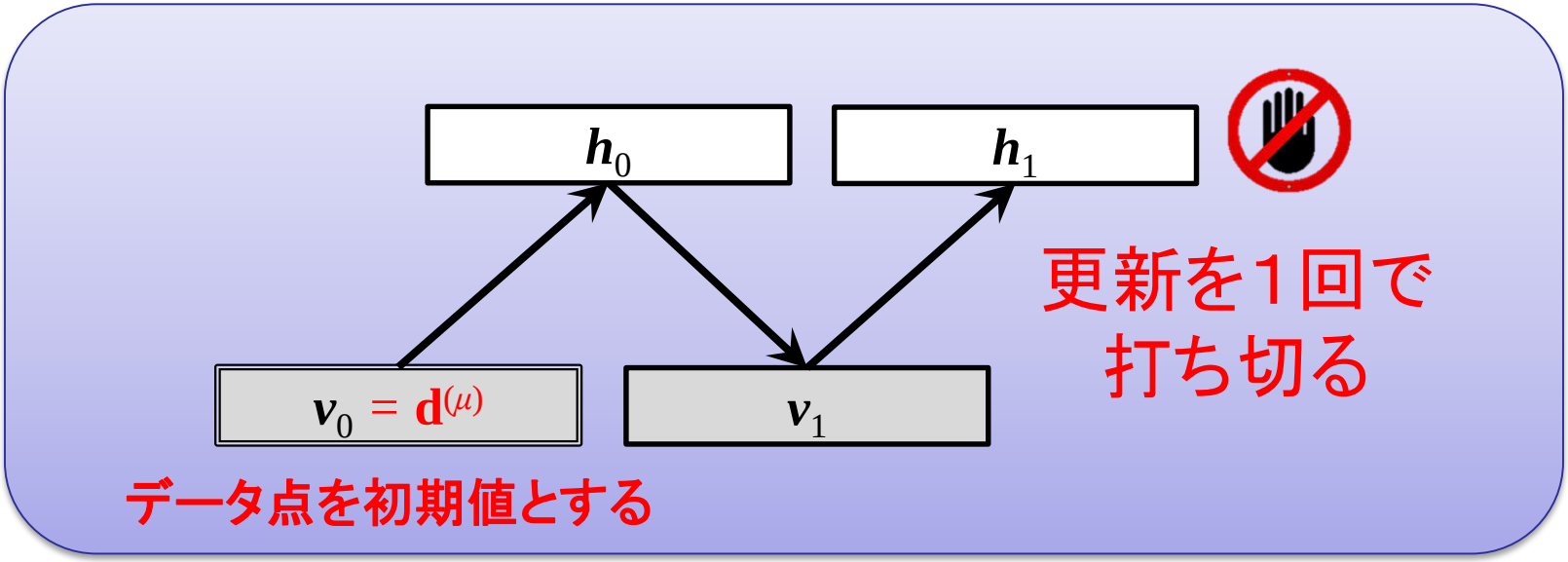
勾配法の毎ステップにおいて**緩和期間**を
必要としてしまい
また**サンプル間隔**もある程度空ける必要があるため

計算コストが非常に高くなってしまう

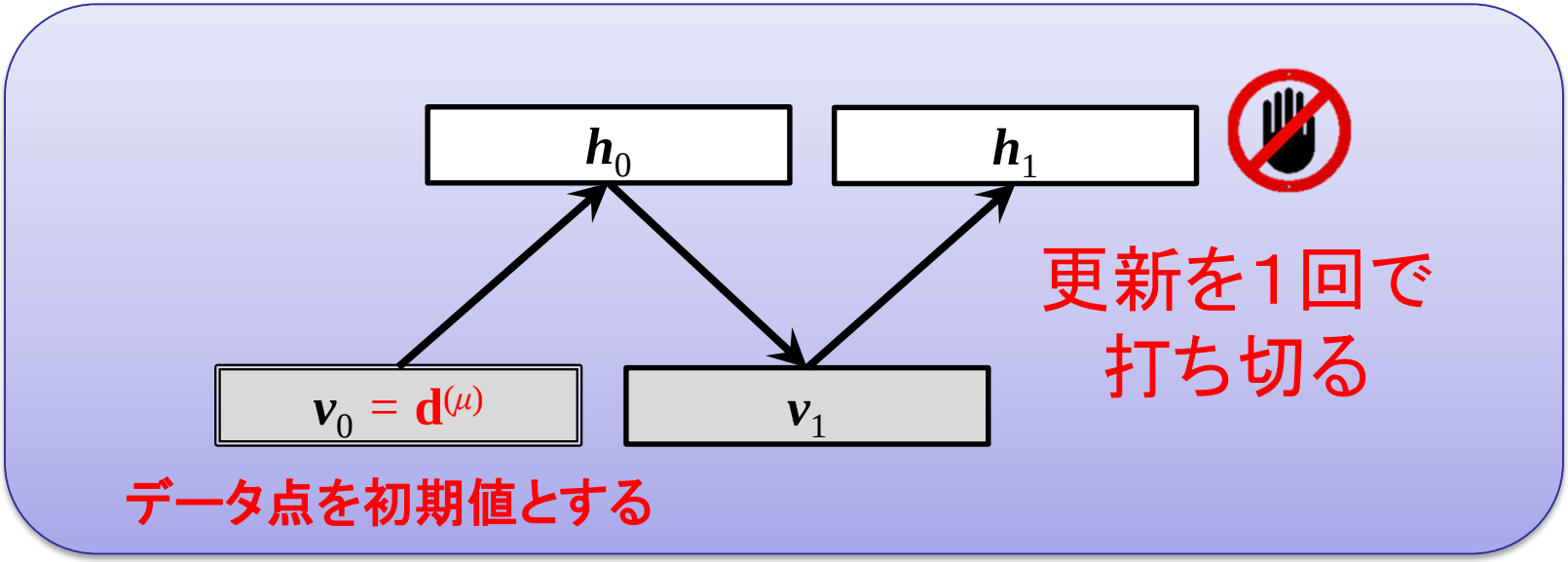
$S \to \infty$ のときは大数の法則で厳密

コントラスティブ・ダイバージェンス





コントラスティブ・ダイバージェンス



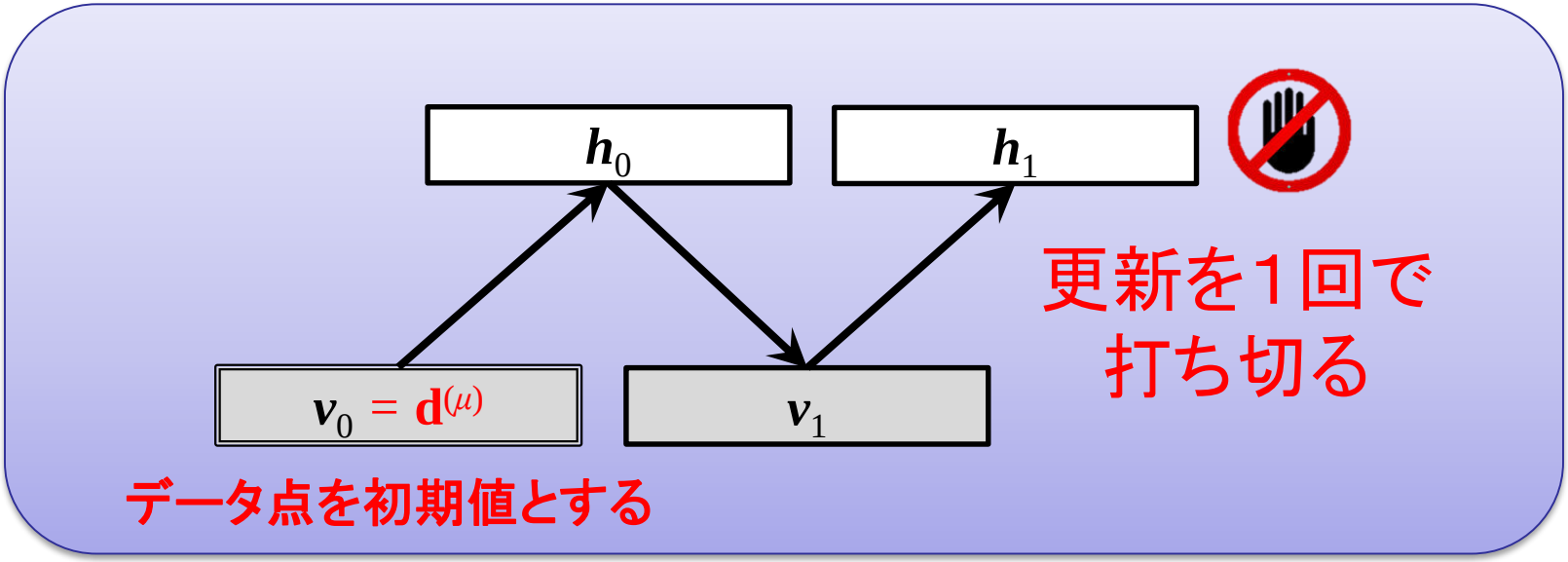
データ点を初期値とした1回の更新でサンプル点を取得する

$\left\{\mathbf{v}^{(\mu)}, \mathbf{h}^{(\mu)}\right\}$ μ番目のデータ点に対するサンプル点

同様な方法によりデータの個数分手に入る

$$\left\{\mathbf{v}^{(1)}, \mathbf{h}^{(1)}\right\}, \left\{\mathbf{v}^{(2)}, \mathbf{h}^{(2)}\right\}, \ldots, \left\{\mathbf{v}^{(N)}, \mathbf{h}^{(N)}\right\}$$

コントラスティブ・ダイバージェンス



各データ点に対するサンプル点の標本平均で
RBM の期待値を近似する

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)}, \quad \mathrm{E}_{\mathrm{RBM}}\left[h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{h}_j^{(\mu)},$$

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)} \mathrm{h}_j^{(\mu)}$$

コントラスティブ・ダイバージェンス

データ点を初期値とすることで
コストの高い緩和期間を省略する

また

サンプル点は各データ点に対して
1つだけ取得するため
一定間隔空けて複数サンプルする
必要がない

コントラスティブ・ダイバージェンス

CD は以下の近似勾配を用いた勾配上昇法によって実行されることとなる

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial b_i} \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} - \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)}$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial c_j} \approx \frac{1}{N}\sum_{\mu=1}^{N} \sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{h}_j^{(\mu)}$$

$$\frac{\partial l_D(\boldsymbol{\theta})}{\partial w_{ij}} \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)}\sigma_1\left(c_j + \sum\nolimits_{k\in V} w_{kj}\mathrm{d}_k^{(\mu)}\right) - \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)}\mathrm{h}_j^{(\mu)}$$

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)}, \quad \mathrm{E}_{\mathrm{RBM}}\left[h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{h}_j^{(\mu)},$$

$$\mathrm{E}_{\mathrm{RBM}}\left[v_i h_j \mid \boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{v}_i^{(\mu)}\mathrm{h}_j^{(\mu)}$$

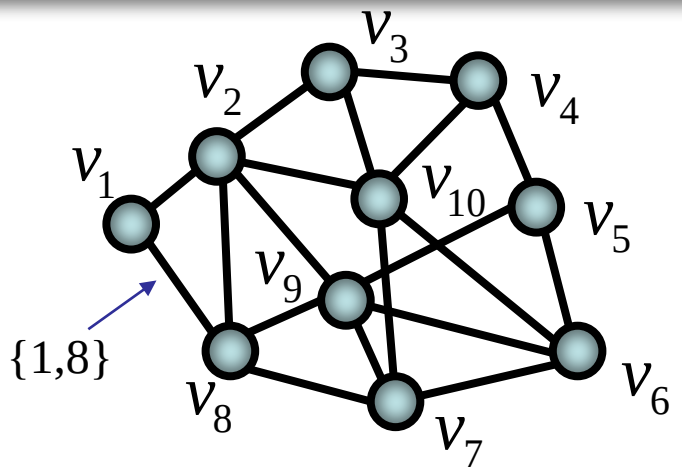
最新の研究結果

An Effective Algorithm for Boltzmann Machine Learning

Boltzmann Machine

$G(V,E)$ ← undirected graph

$V = \{1,2,\ldots,10\}$ ← set of nodes



$E = \{\{1,2\},\{1,8\},\{2,3\},\{2,8\},\{2,9\},\ldots\}$ ← set of links

energy function of BM

$$C(\boldsymbol{v};\boldsymbol{b},\boldsymbol{w}) := -\sum_{i\in V} b_i v_i - \sum_{\{i,j\}\in E} w_{ij} v_i v_j$$

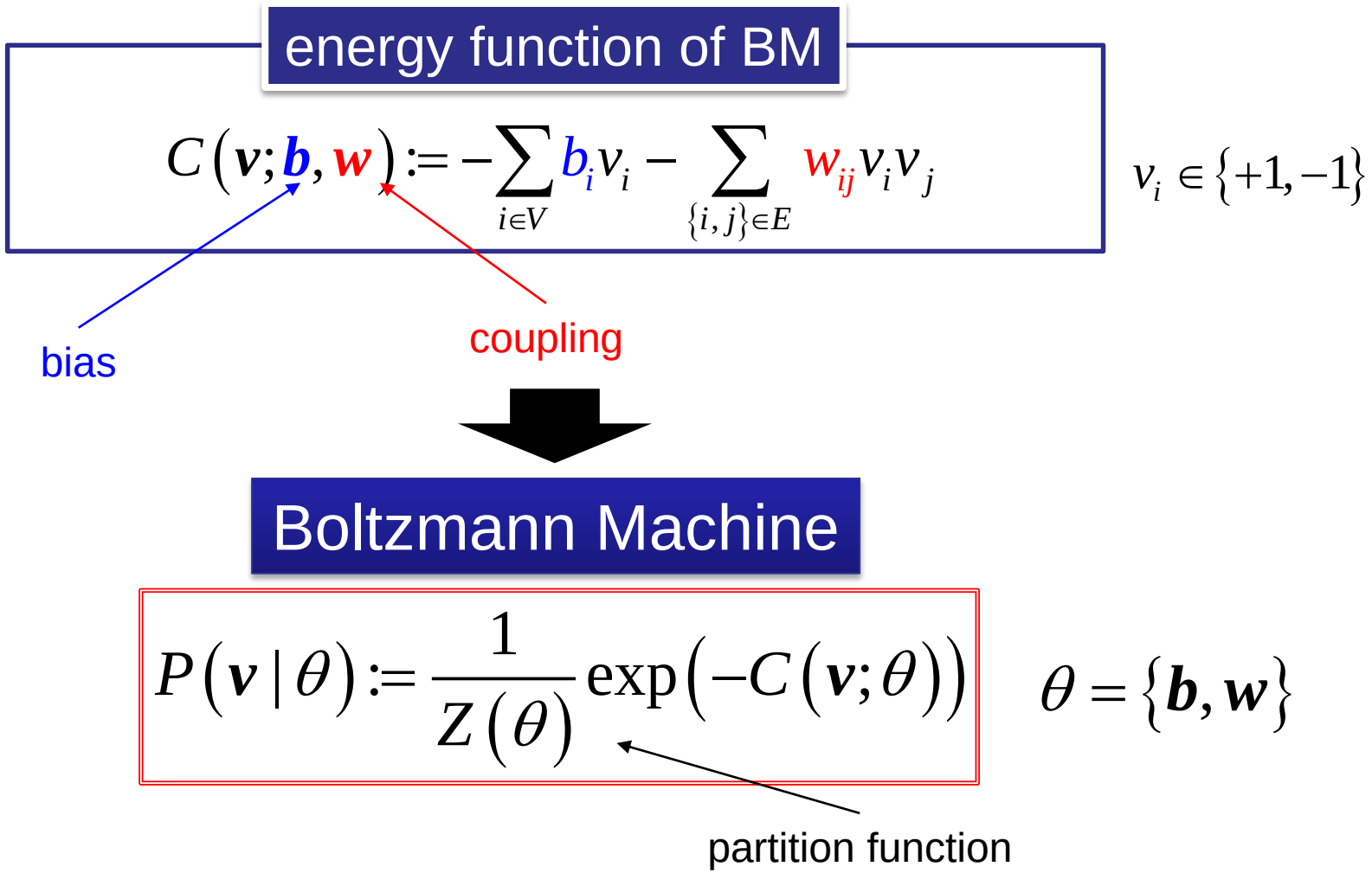
$v_i \in \{+1,-1\}$

bias

coupling

$w_{ij} = w_{ji}$ (symmetric couplings)

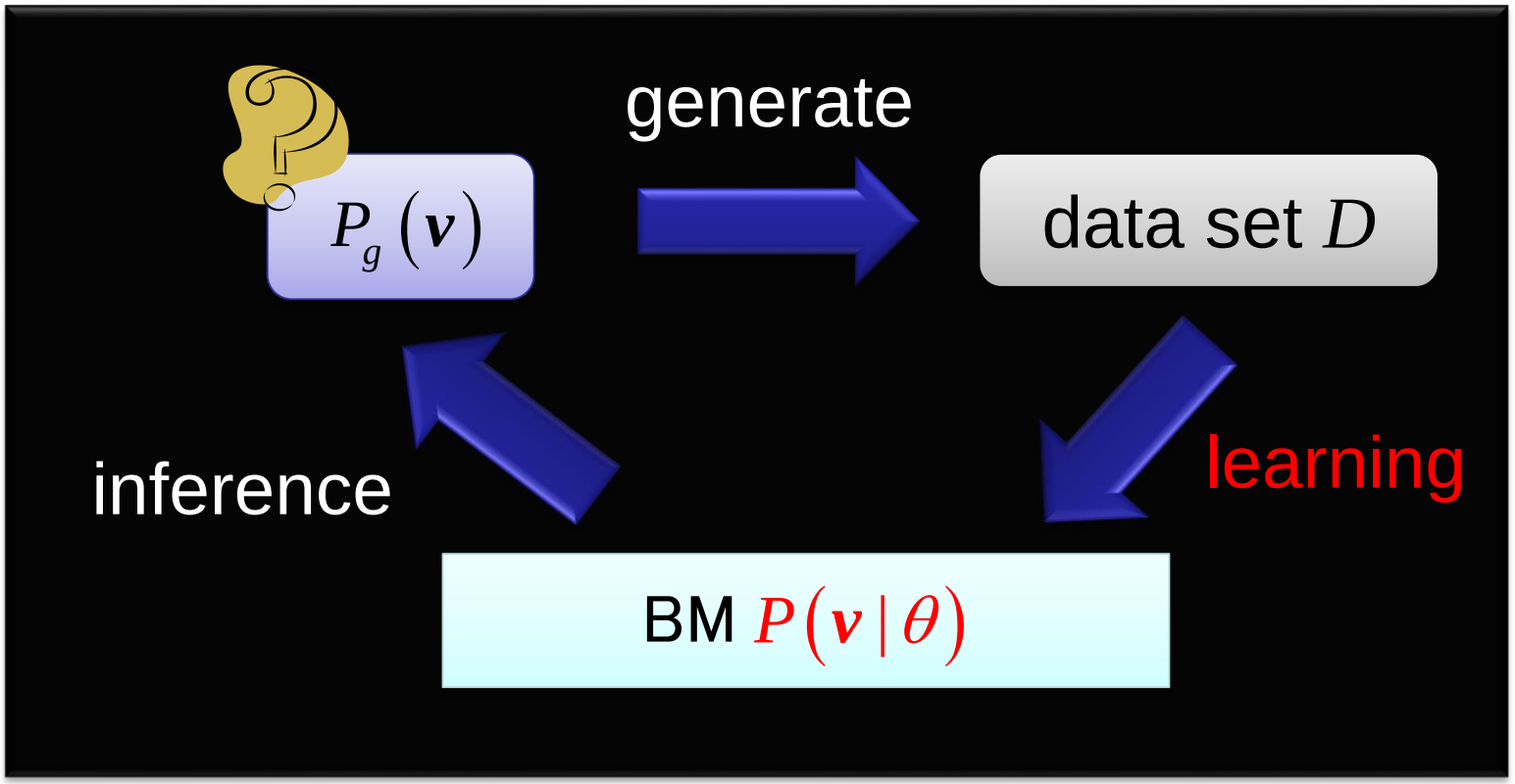
Boltzmann Machine



BM is the simplest probabilistic model in Markov random fields

Statistical Machine Learning

The aim of the statistical machine learning is to find the unknown generative model.



Using data set D to tune the values of the parameters,

we approximate the generative model by our model (BM).

Maximum Likelihood Estimation

set of N data points (each data point contains n elements):

$$D = \underbrace{\left\{\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}\right\}}_{N \text{ data points}}, \quad \mathbf{d}^{(\mu)} = \underbrace{\left\{\mathrm{d}_1^{(\mu)}, \mathrm{d}_2^{(\mu)}, \ldots, \mathrm{d}_n^{(\mu)}\right\}}_{n \text{ elements}}$$

$$\mathrm{d}_i^{(\mu)} \in \{+1, -1\}$$

Each data point is generated from a certain generative model

$$P_g(\boldsymbol{v}) \xrightarrow[\text{generate (i.i.d.)}]{} \mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(N)}$$

log-likelihood function $l_D(\theta) := \frac{1}{N} \sum_{\mu=1}^{N} \ln P\left(\boldsymbol{v} = \mathbf{d}^{(\mu)} \mid \theta\right)$

Maximum likelihood estimation (MLE)

$$\theta^* = \arg\max_{\theta} l_D(\theta)$$

Maximum Likelihood Estimation

Maximum likelihood estimation (MLE)

$$\theta^* = \arg\max_{\theta} l_D(\theta)$$

gradients

$$\frac{\partial l_D(\theta)}{\partial b_i} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} - \mathrm{E}\left[v_i \mid \theta\right]$$

$$\frac{\partial l_D(\theta)}{\partial w_{ij}} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)}\mathrm{d}_j^{(\mu)} - \mathrm{E}\left[v_i v_j \mid \theta\right]$$

$$\mathrm{E}\left[(\cdots) \mid \theta\right] := \sum_{\boldsymbol{v}} (\cdots) P(\boldsymbol{v} \mid \theta) = \sum_{v_1}\sum_{v_2}\cdots\sum_{v_n} (\cdots) P(\boldsymbol{v} \mid \theta)$$

The second terms are the expectations of BM and they include the intractable multiple-summations.

we need to evaluate the expectations with an approximate method

Monte Carlo Integration

Monte Carlo Integration (MCI) is one of the most fundamental method to approximate the expectations.

sampling phase

MC sampling

$P(\boldsymbol{v} \mid \theta)$ ➡ $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(K)}$

model

sampled points

averaging phase

$$\mathrm{E}\left[f(\boldsymbol{v}) \mid \theta \right] \approx \frac{1}{K} \sum_{k=1}^{K} f\left(\mathbf{s}^{(k)}\right)$$

sample average

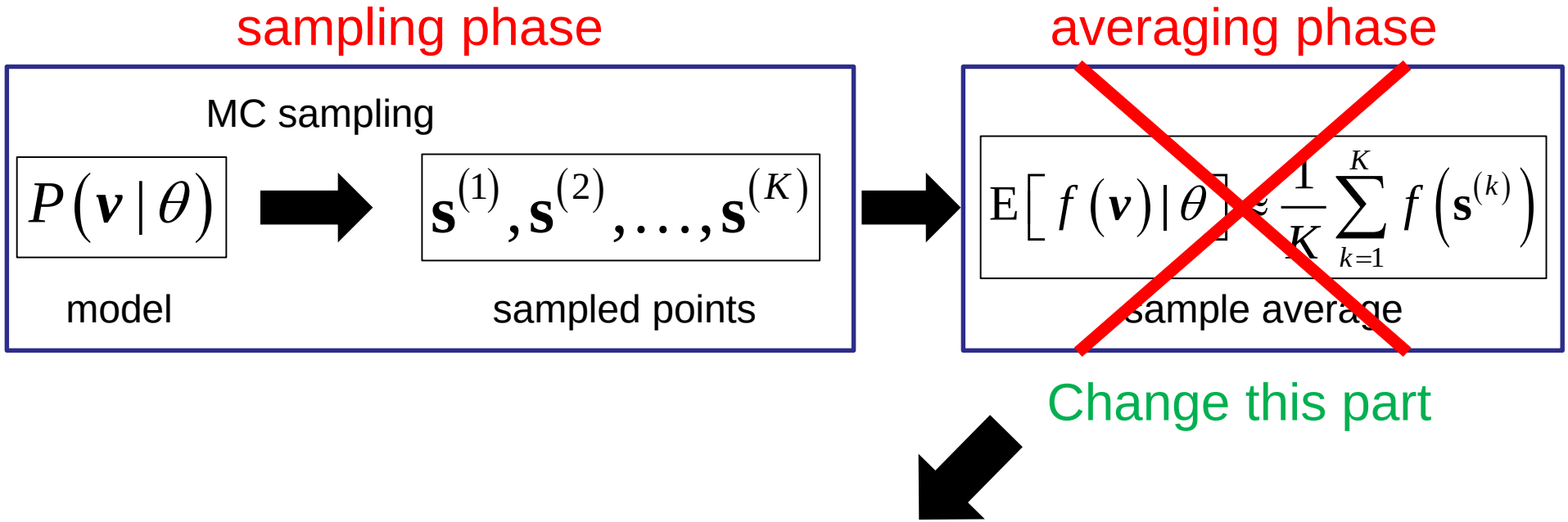
MCI is the approximate method on the basis of the random sampling.

In the sampling phase,
we use information of the model (parameters, structures, and so on)

In the averaging phase,
we do not use any information of the model

Spatial Monte Carlo Integration

Monte Carlo Integration (MCI) is one of the most fundamental method to approximate the expectations.

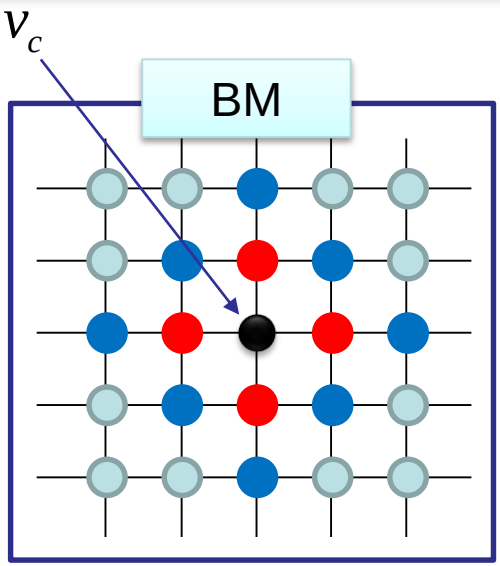


In the new method, spatial Monte Carlo Integration (SMCI),

we use available information of the model in the averaging phase.

[Yasuda, 15]

Spatial Monte Carlo Integration



$P(\boldsymbol{v} \mid \theta)$

We focus on
the expectation of a function of the target variable(s) v_c

$$E\left[f(v_c) \mid \theta \right] = \sum_{\boldsymbol{v}} v_c P(\boldsymbol{v} \mid \theta).$$

v_c : target variable(s)

$\boldsymbol{v}_c^{(1)}$: set of first-nearest variables of the target (shown by red)

$\boldsymbol{v}_c^{(2)}$: set of second-nearest variables of the target (shown by blue)

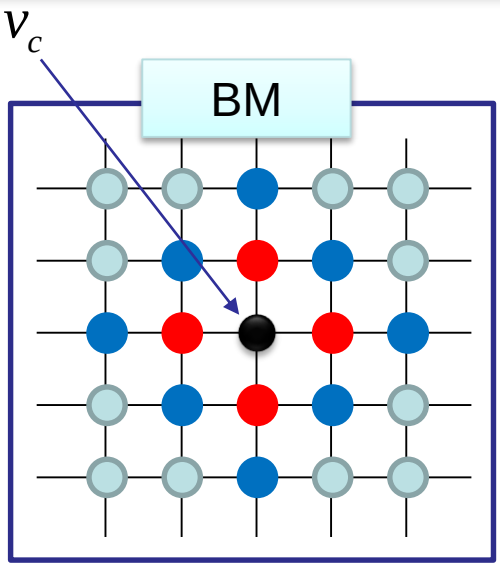
We can easily obtain the distribution conditioned by the kth-nearest variables:

$$P\left(v_c, \boldsymbol{v}_c^{(1)}, \boldsymbol{v}_c^{(2)}, \ldots, \boldsymbol{v}_c^{(k-1)} \mid \boldsymbol{v}_c^{(k)}, \theta\right)$$

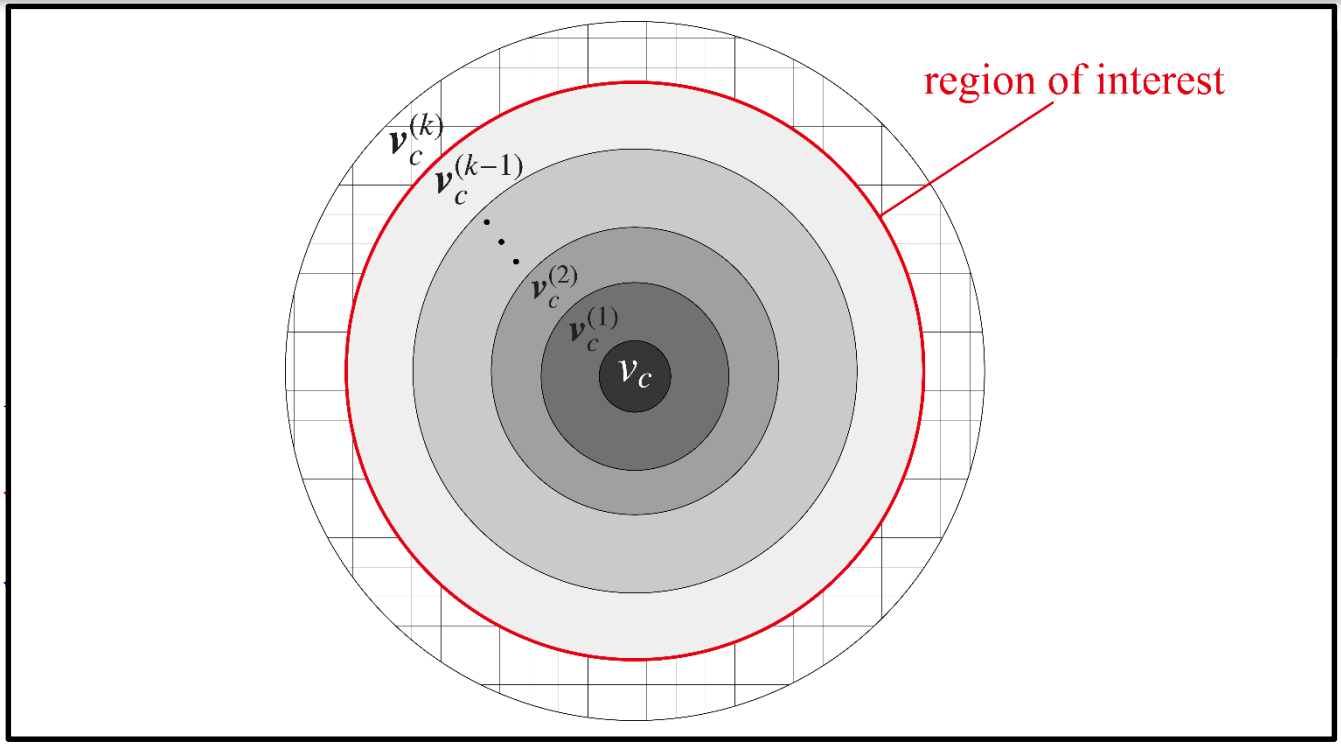
This distribution is also BM on the region of interest.

$$\left\{ v_c, \boldsymbol{v}_c^{(1)}, \boldsymbol{v}_c^{(2)}, \ldots, \boldsymbol{v}_c^{(k-1)} \right\}$$

Spatial Monte Carlo Integration



$$P(\boldsymbol{v}\,|\,\theta)$$



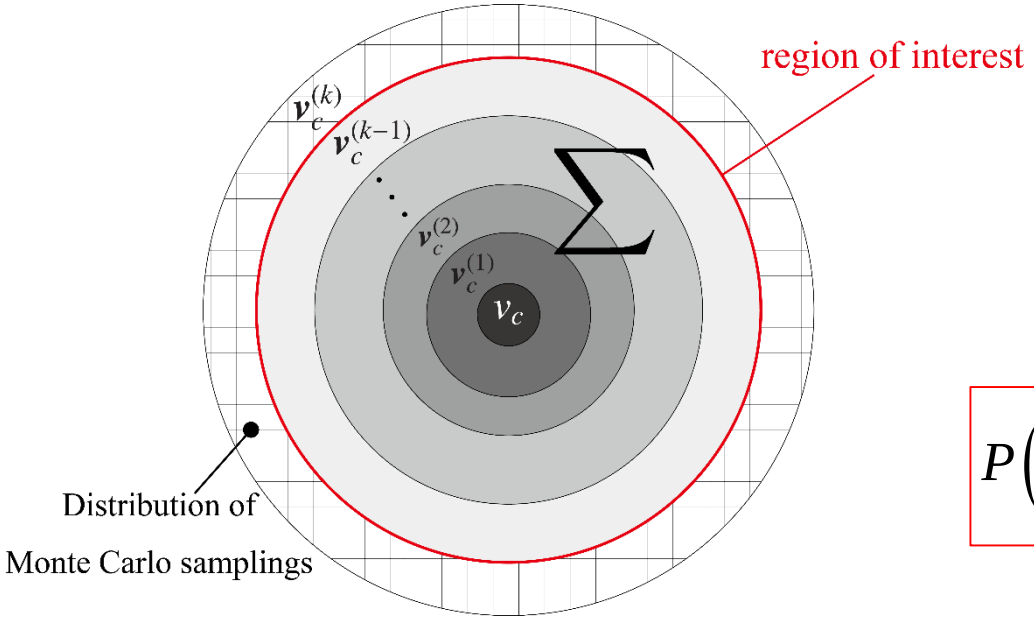
We can easily obtain the distribution conditioned with the kth-nearest variables:

$$P\left(v_c, \boldsymbol{v}_c^{(1)}, \boldsymbol{v}_c^{(2)}, \ldots, \boldsymbol{v}_c^{(k-1)} \,|\, \boldsymbol{v}_c^{(k)}, \theta\right)$$

This distribution is also BM on the region of interest.

$$\left\{v_c, \boldsymbol{v}_c^{(1)}, \boldsymbol{v}_c^{(2)}, \ldots, \boldsymbol{v}_c^{(k-1)}\right\}$$

Spatial Monte Carlo Integration



The marginal distribution
on the outside of the region

is approximated by
the distribution of the sampled points

$$P\left(\mathbf{v}_c^{(k)} \mid \theta\right) \approx Q\left(\mathbf{v}_c^{(k)}\right) := \frac{1}{K}\sum_{k=1}^{K} \delta\left(\mathbf{v}_c^{(k)}, \mathbf{s}_c^{(k)}\right)$$

$$E\left[f\left(v_c\right) \mid \theta \right] = \sum_{v_c} \sum_{\mathbf{v}_c^{(1)}} \sum_{\mathbf{v}_c^{(2)}} \cdots \sum_{\mathbf{v}_c^{(k-1)}} \sum_{\mathbf{v}_c^{(k)}} f\left(v_c\right) P\left(v_c, \mathbf{v}_c^{(1)}, \mathbf{v}_c^{(2)}, \ldots, \mathbf{v}_c^{(k-1)} \mid \mathbf{v}_c^{(k)}, \theta\right) P\left(\mathbf{v}_c^{(k)} \mid \theta\right)$$

$$\approx \sum_{v_c} \sum_{\mathbf{v}_c^{(1)}} \sum_{\mathbf{v}_c^{(2)}} \cdots \sum_{\mathbf{v}_c^{(k-1)}} \sum_{\mathbf{v}_c^{(k)}} f\left(v_c\right) P\left(v_c, \mathbf{v}_c^{(1)}, \mathbf{v}_c^{(2)}, \ldots, \mathbf{v}_c^{(k-1)} \mid \mathbf{v}_c^{(k)}, \theta\right) Q\left(\mathbf{v}_c^{(k)}\right)$$

$$= \frac{1}{K}\sum_{k=1}^{K} \sum_{v_c} \sum_{\mathbf{v}_c^{(1)}} \sum_{\mathbf{v}_c^{(2)}} \cdots \sum_{\mathbf{v}_c^{(k-1)}} f\left(v_c\right) P\left(v_c, \mathbf{v}_c^{(1)}, \mathbf{v}_c^{(2)}, \ldots, \mathbf{v}_c^{(k-1)} \mid \mathbf{s}_c^{(k)}, \theta\right)$$

*k*th-order SMCI (*k*-SMCI)

Example: 1-SMCI

kth-order SMCI (k-SMCI)

$$E\left[f\left(v_c\right) | \theta \right] \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{v_c} \sum_{\boldsymbol{v}_c^{(1)}} \sum_{\boldsymbol{v}_c^{(2)}} \cdots \sum_{\boldsymbol{v}_c^{(k-1)}} f\left(v_c\right) P\left(v_c, \boldsymbol{v}_c^{(1)}, \boldsymbol{v}_c^{(2)}, \ldots, \boldsymbol{v}_c^{(k-1)} | \mathbf{s}_c^{(k)}, \theta\right)$$

The first- and second-order moments given by 1-SMCI

$$E\left[v_i | \theta\right] \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{v_i} v_i P\left(v_i | \mathbf{s}_i^{(1)}, \theta\right) = \frac{1}{K} \sum_{k=1}^{K} \tanh U_i^{(k)}$$

$$E\left[v_i \, \mathrm{v}_j | \theta\right] \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{v_i} \sum_{v_i} v_i v_j P\left(v_i \, \mathrm{v}_j | \mathbf{s}_{\{i,j\}}^{(1)}, \theta\right) = \frac{1}{K} \sum_{k=1}^{K} \tanh\left[\tanh^{-1}\left[\tanh U_{i,j}^{(k)} \tanh U_{j,i}^{(k)} \right] + w_{ij} \right]$$

$$U_i^{(k)} := b_i + \sum_{j \in \partial i} w_{ij} \mathrm{s}_j^{(k)}, \quad U_{i,j}^{(k)} := U_i^{(k)} - w_{ij} \mathrm{s}_j^{(k)}$$

$\partial i := \left\{ j \,|\, \{i, j\} \in E \right\}$: the set of nearest-neighbor nodes of node i.

Spatial Monte Carlo Integration

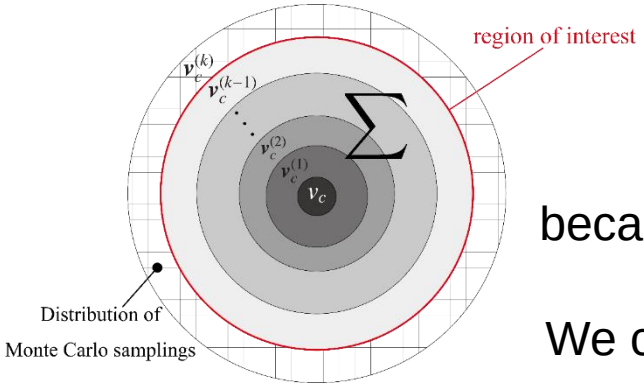
The important two facts can be proven:

- SMCI is statistically **better** than the normal MCI
- a higher-order SMCI is statistically **better** than a lower-order one

Merit of SMCI

SMCI gives **high-accuracy** approximations

Demerit of SMCI



Higher-order SMCI is slower,
because we have to take the sum over the region of interest.

We cannot use SMCI in a dense system, except for 1-SMCI.

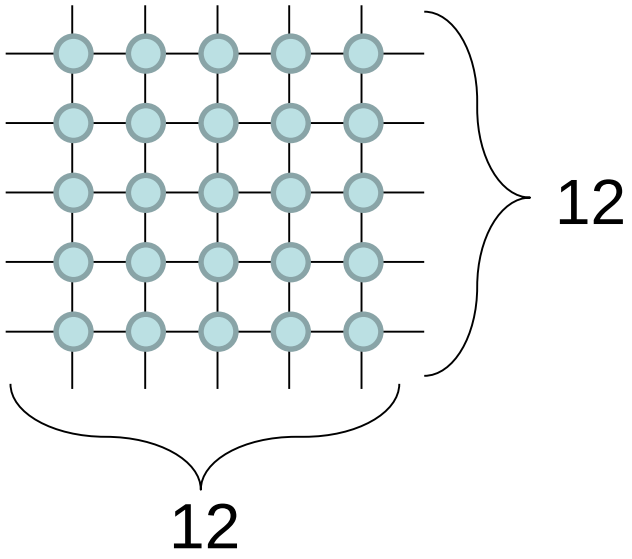
Numerical Experiment

Generative Model

Boltzmann machine

$$P_g(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_{\{i,j\}\in E} w_{ij} v_i v_j\right), \qquad v_i \in \{+1,-1\}$$

$$w_{ij} \underset{\text{i.i.d.}}{\sim} U[-0.5, 0.5]$$

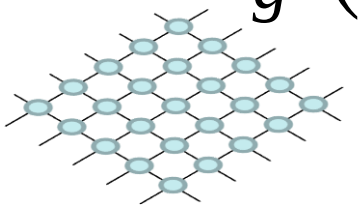


Gibbs sampling method

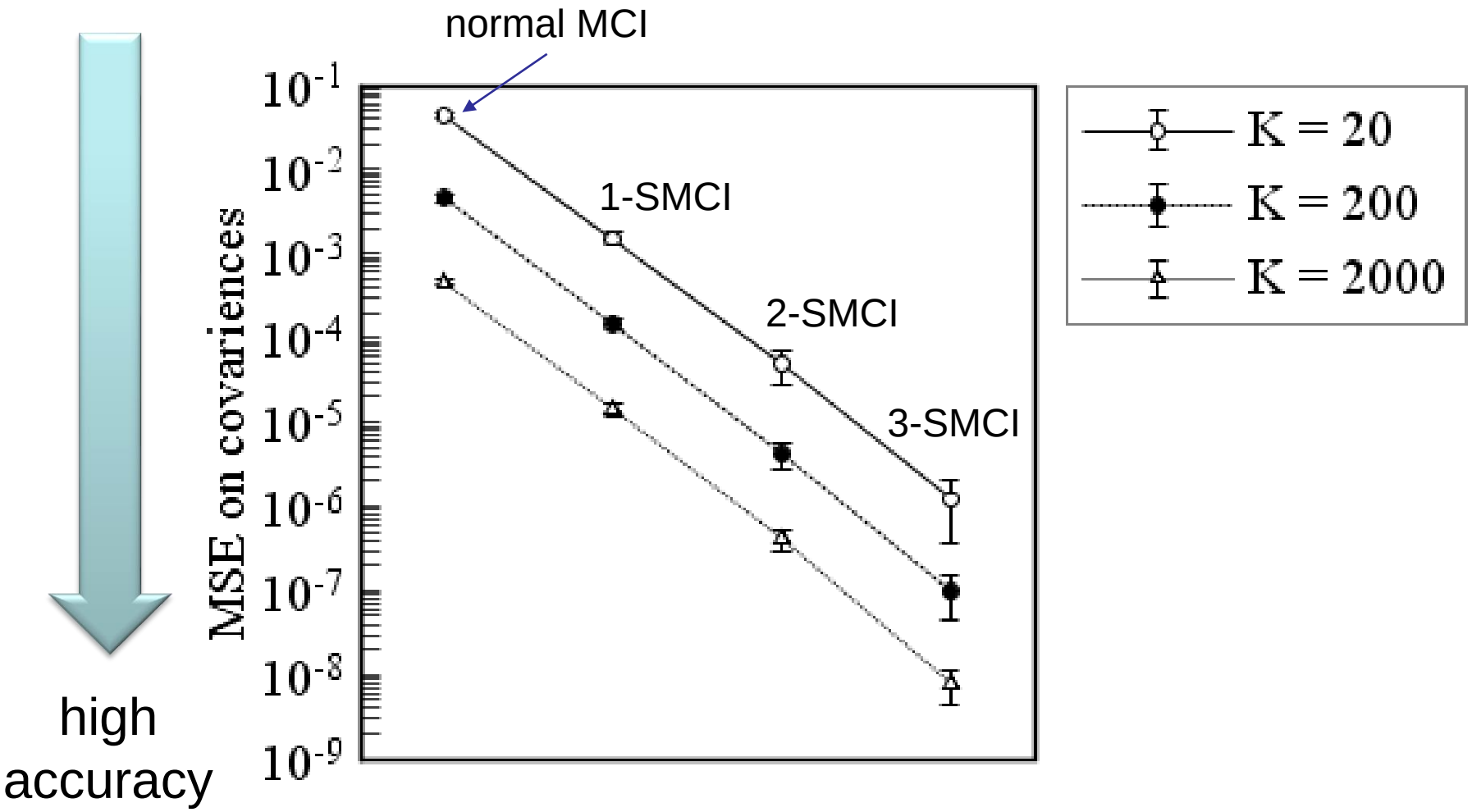
$$P_g(\boldsymbol{v})$$

SAMPLING

$$\left\{\mathbf{s}^{(k)} \mid k = 1, 2, \cdots, K\right\}$$



Numerical Experiment



A higher-order SMCI is at least 10 times better than a lower-order one.

MSE : mean square error between the exact values and the approximated values

Application to BM Learning

Basic Idea

$$\frac{\partial l_D(\theta)}{\partial b_i} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)} - \mathrm{E}\left[v_i \mid \theta\right], \quad \frac{\partial l_D(\theta)}{\partial w_{ij}} = \frac{1}{N}\sum_{\mu=1}^{N} \mathrm{d}_i^{(\mu)}\mathrm{d}_j^{(\mu)} - \mathrm{E}\left[v_i v_j \mid \theta\right]$$

Approximate these intractable expectations by SMCI

By regarded the observed data set D as the MC samplings S generated from the model and by using D instead of S,

we can skip the sampling phase in the MCI.

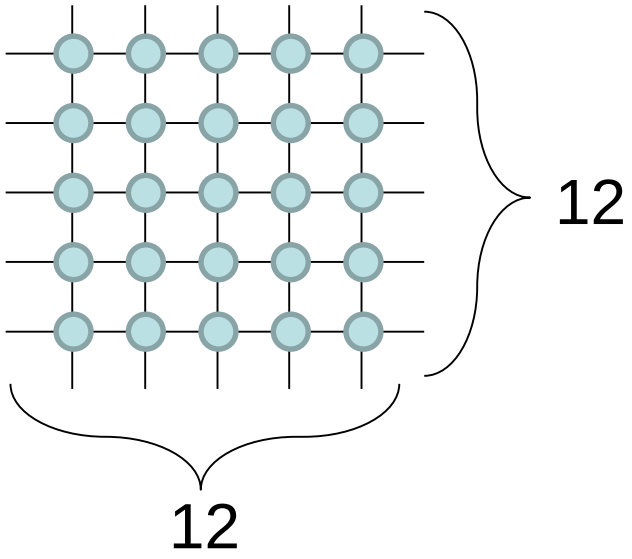
Numerical Experiment (Learning)

Generative Model

Boltzmann machine

$$P_g(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_{\{i,j\}\in E} w_{ij} v_i v_j\right), \qquad v_i \in \{+1,-1\}$$

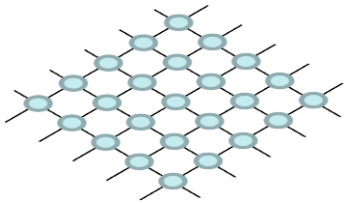
$$w_{ij} \underset{\text{i.i.d.}}{\sim} U[-0.5, 0.5]$$

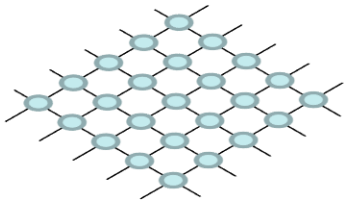


Gibbs sampling method

$P_g(\boldsymbol{v})$ → SAMPLING → $D = \{\mathbf{d}^{(\mu)} \mid \mu = 1, 2, \cdots, N\}$ → LEARNING → $P(\boldsymbol{v} \mid \theta)$

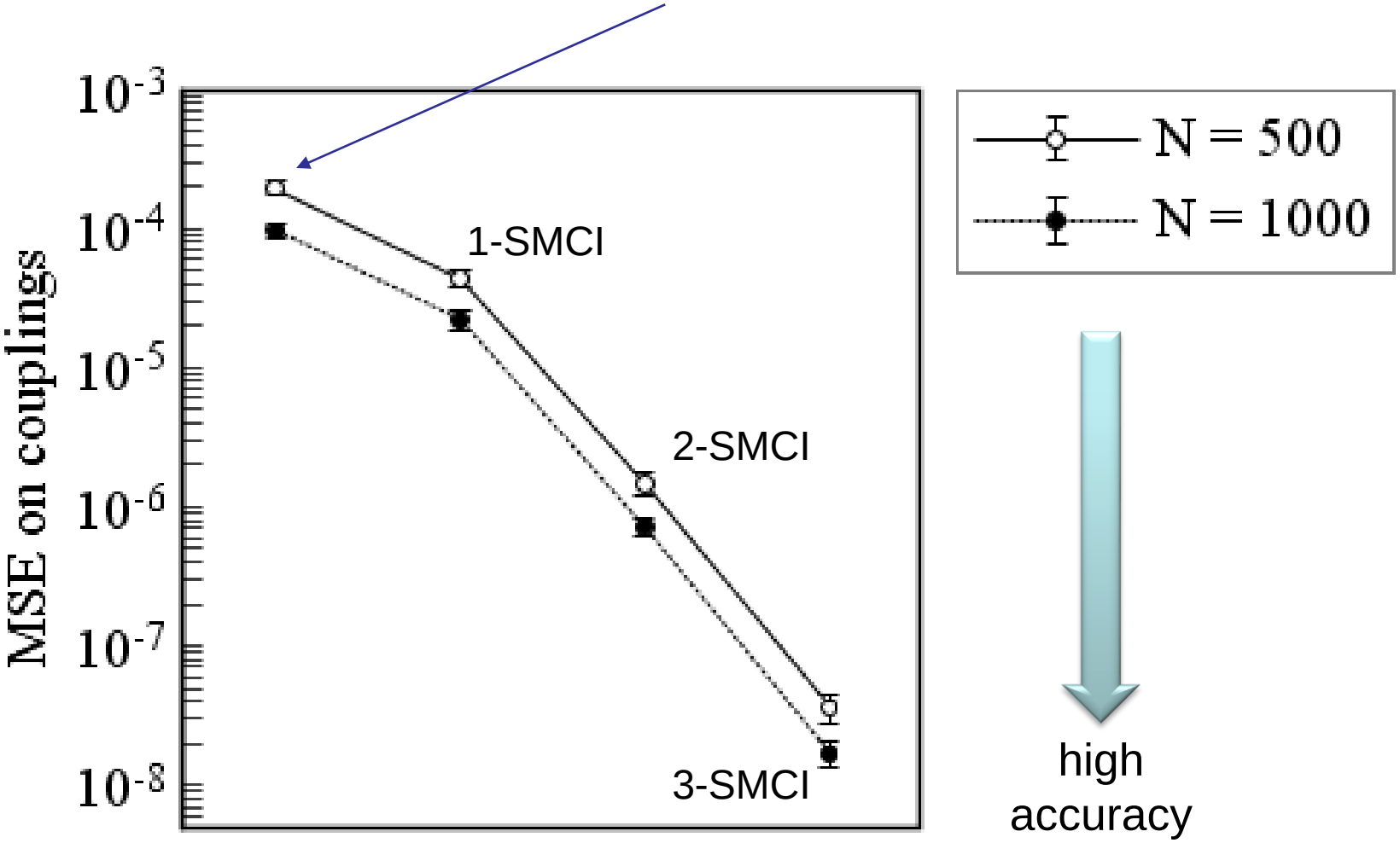
synthetic data set





Numerical Experiment (Learning)

maximum pseudo-likelihood estimation (MPLE) [Basag: *The Statistician*, 75]



MSE : mean square error between the ML estimators and the approximated values

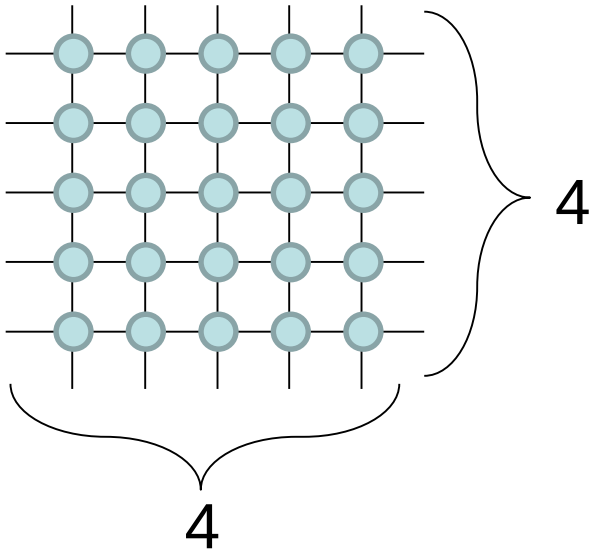
Numerical Experiment (Learning)

Generative Model

Boltzmann machine

$$P_g(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_{\{i,j\}\in E} w_{ij} v_i v_j\right), \qquad v_i \in \{+1,-1\}$$

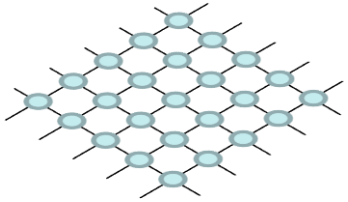
$$w_{ij} \underset{\text{i.i.d.}}{\sim} U[-0.5, 0.5]$$

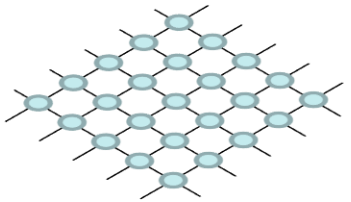


Gibbs sampling method

$P_g(\boldsymbol{v})$ → SAMPLING → $D = \{\mathbf{d}^{(\mu)} \mid \mu = 1, 2, \cdots, 200\}$ → LEARNING → $P(\boldsymbol{v} \mid \theta)$

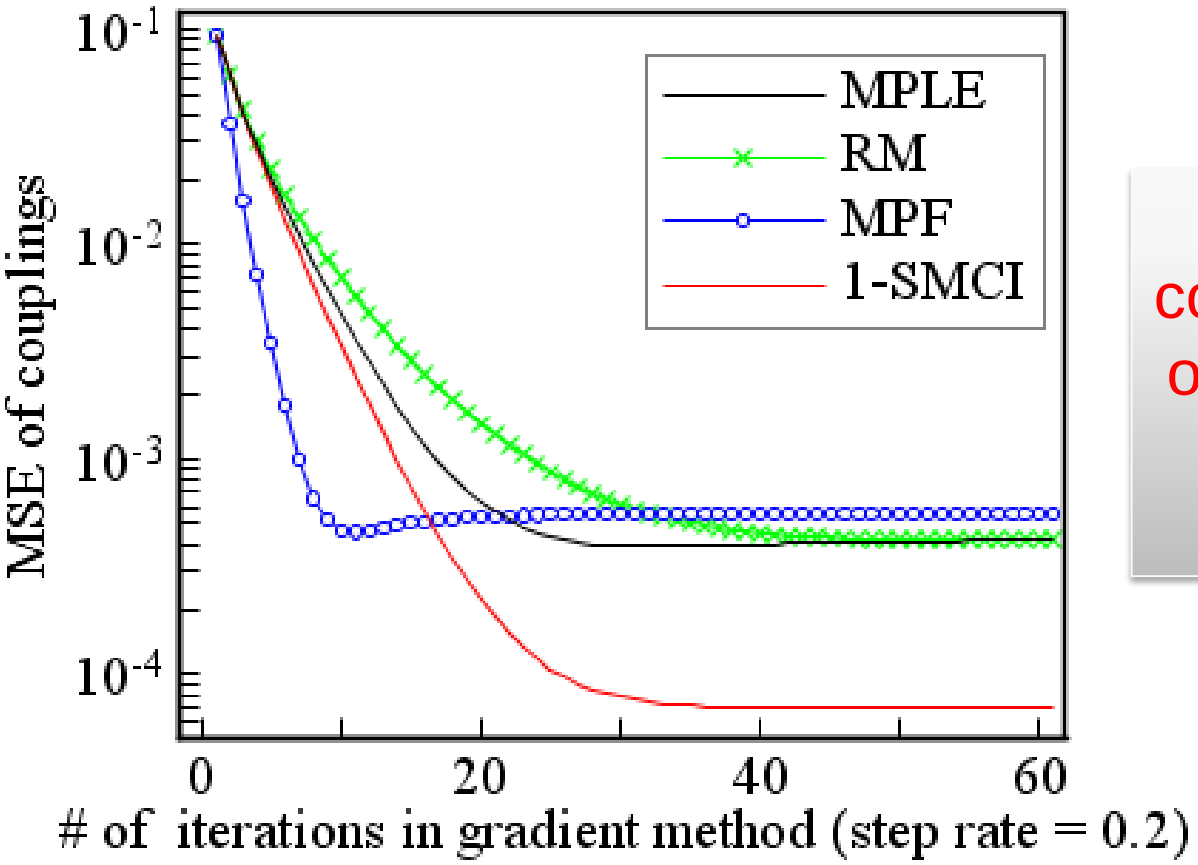
synthetic data set





Numerical Experiment (Learning)

comparison with the other recent learning algorithms



The orders of computational costs of these algorithms are the same

MPLE: maximum pseudo-likelihood estimation [Basag: *The Statistician*, 75]
RM: ratio matching [Hyvärinen: *CSDA*, 07]
MPF: minimum probability flow [Sohl-Dickstein: *PRL*, 11]

平均場アルゴリズムを用いた

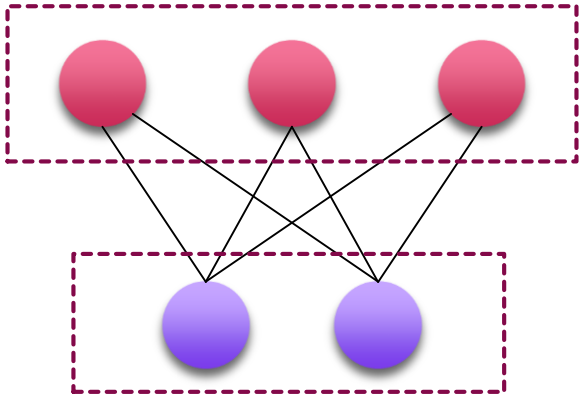
ガウス型制限ボルツマンマシン

背景

制限ボルツマンマシンは２部グラフ上に定義される確率モデル

隠れ層 H
隠れ変数 $\boldsymbol{h}$: データと関連しない

可視層 V
可視変数 $\boldsymbol{v}$: データと関連する



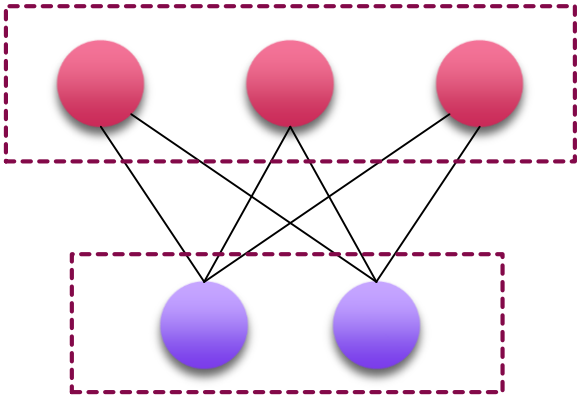
$$\boldsymbol{h} = \{h_j \mid j \in H\}$$

$$\boldsymbol{v} = \{v_i \mid i \in V\}$$

制限ボルツマンマシン (RBM)　→　可視変数：離散　隠れ変数：離散

ガウス型制限ボルツマンマシン (GRBM)　→　可視変数：**連続**　隠れ変数：離散

背景



学習フェーズ

得られたデータを用いて最適なパラメータの値を決定

e.g. コントラスティブ・ダイバージェンス

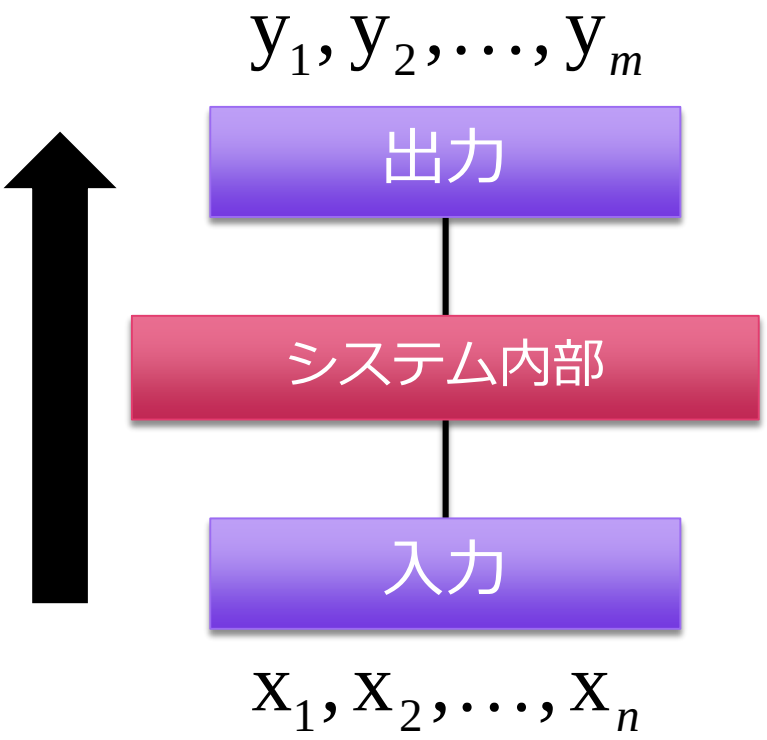
推定フェーズ

学習により得られたモデルを用いた入出力値推定

出力変数の期待値が出力値となる（MMSE推定法）

ギブスサンプリング
や
平均場近似法

学習？推定？ ～ 研究領域のイメージのために～



ある入力 $\mathbf{x}$ から

適切な出力 $\mathbf{y}$ を得る

（確率モデルベースの）
システムを考える

画像修復システム設計を例として

$y_1, y_2, \ldots, y_m$

出力

システム内部

入力

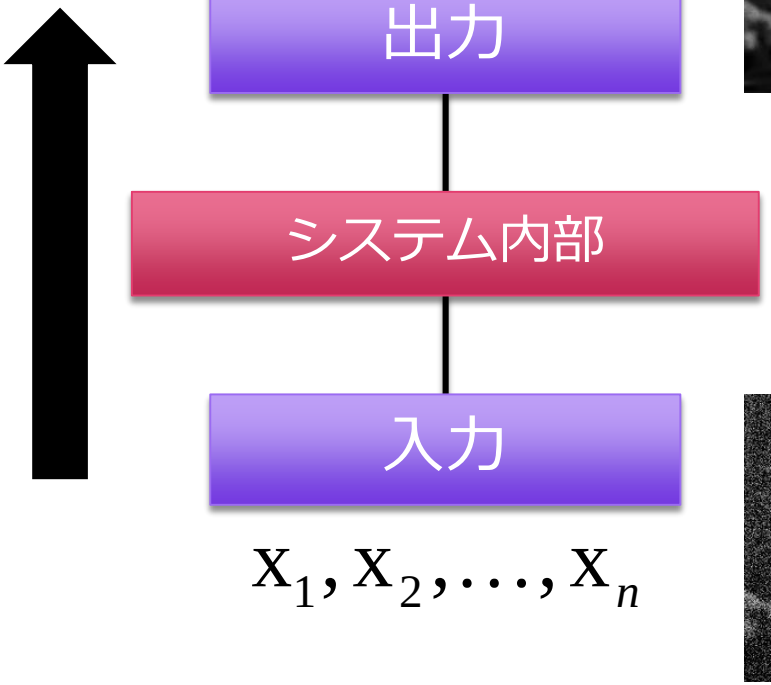
$x_1, x_2, \ldots, x_n$

例えば
画像修復システムなら

入力 $\mathbf{x}$ が劣化画像
で
出力 $\mathbf{y}$ が修復画像

という具合

機械学習



入力から適切な出力を出すように

システムの内部パラメータを
最適化する必要がある

教師あり機械学習では

入力とそれに対応する出力（教師）
の集合（訓練データ集合）
をシステムに与えて

その与えた入出力データの関係が
正しく成立するように
内部パラメータを最適化する

機械学習

$y_1, y_2, \ldots, y_m$

出力

システム内部

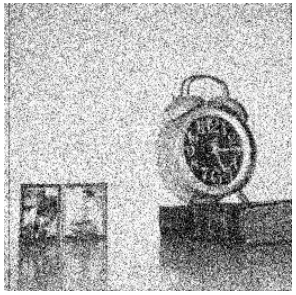
入力

$x_1, x_2, \ldots, x_n$



・・・

訓練データ集合の入出力関係が
（高確率で）実現するようなパラメータ
を探すことが機械学習の目的


・・・

出力の推定

$y_1, y_2, \ldots, y_m$

出力

システム内部

入力

$x_1, x_2, \ldots, x_n$

機械学習により
システムの内部パラメータ
の最適化が完了したら

得られたシステムを用いて
実際に画像修復を行うことが出来る

出力の推定

$y_1, y_2, \ldots, y_m$

出力

システム内部

入力

$x_1, x_2, \ldots, x_n$

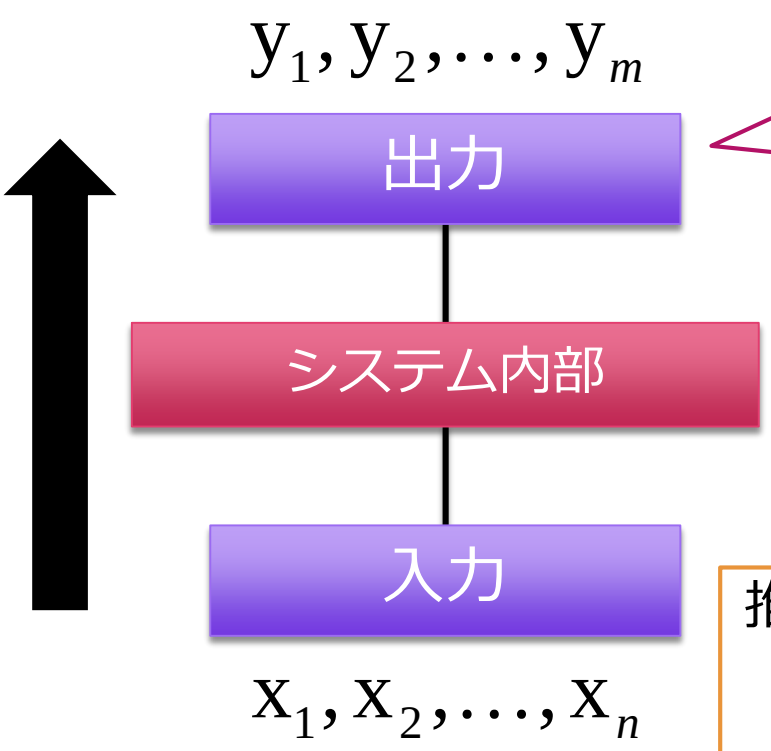
確率モデルベースのシステムでは
出力 $\mathbf{y}$ は確率的にサンプリングされる
ので
サンプリング毎に違った値となる

我々が欲しいのは唯一の修復画像なので
出力を何か一つに決めなければならない

機械学習により
システムの内部パラメータ
の最適化が完了したら

得られたシステムを用いて
実際に画像修復を行うことが出来る

出力の推定



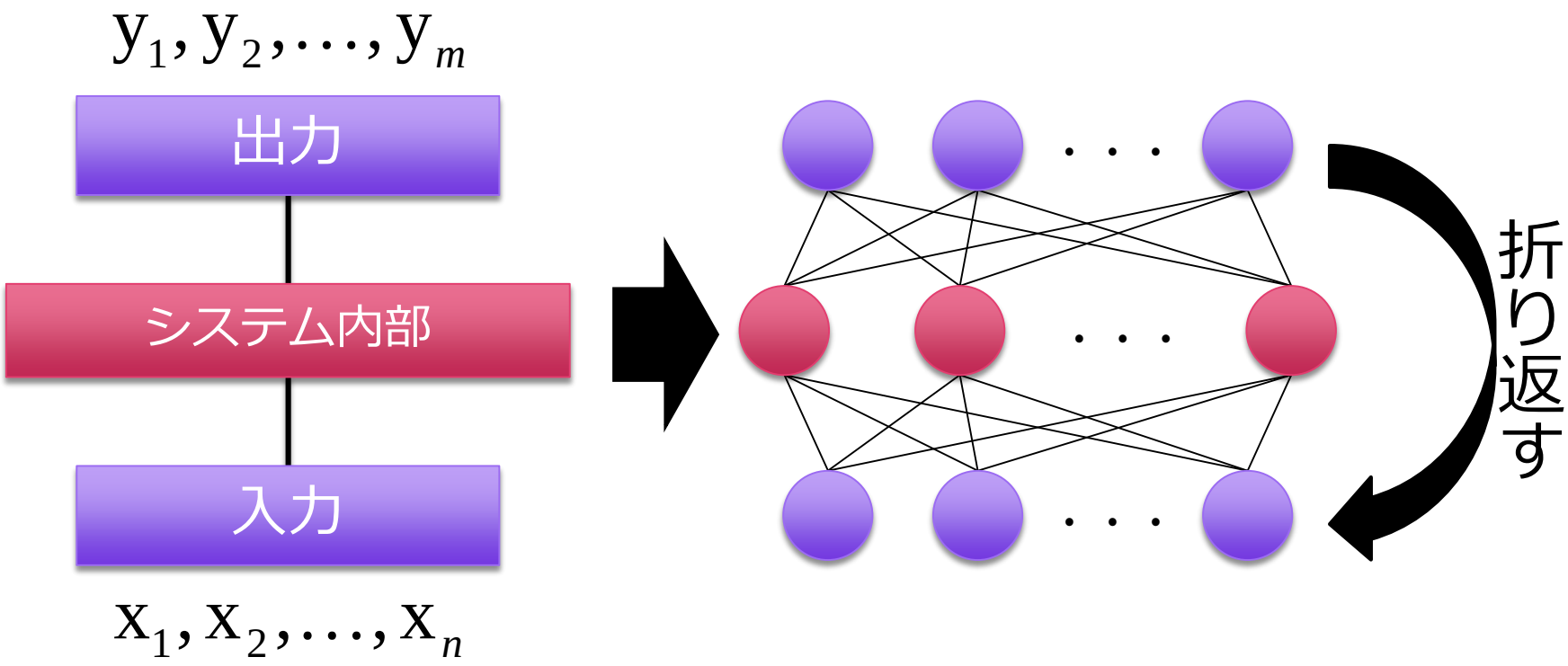
確率モデルベースのシステムでは
出力 $\mathbf{y}$ は確率的にサンプリングされる
ので
サンプリング毎に違った値となる

我々が欲しいのは唯一の修復画像なので
出力を何か一つに決めなければならない

推定法（出力を一つに決める方法）は様々だが

MMSE（最小平均２乗誤差）推定では
サンプリングの期待値を出力とする

RBM との架け橋

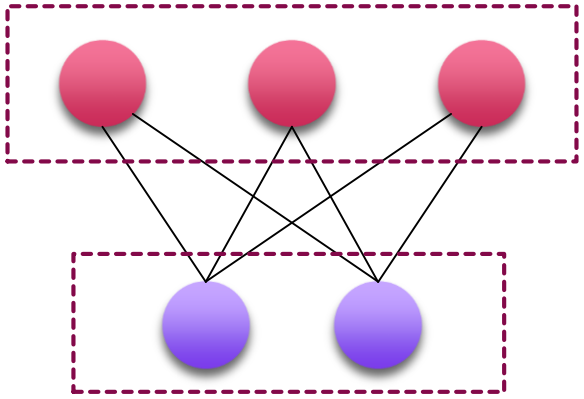


背景（再掲）

制限ボルツマンマシンは２部グラフ上に定義される確率モデル

隠れ層 H
隠れ変数 $\boldsymbol{h}$: データと関連しない

可視層 V
可視変数 $\boldsymbol{v}$: データと関連する



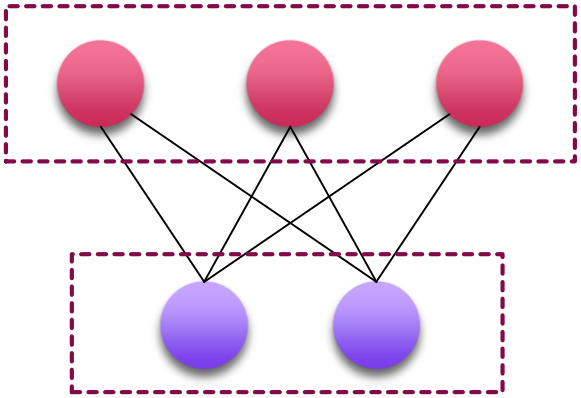
$$\boldsymbol{h} = \{h_j \mid j \in H\}$$

$$\boldsymbol{v} = \{v_i \mid i \in V\}$$

制限ボルツマンマシン (RBM)　→　可視変数：離散　隠れ変数：離散

ガウス型制限ボルツマンマシン (GRBM)　→　可視変数：**連続**　隠れ変数：離散

背景（再掲）



学習フェーズ

得られたデータを用いて最適なパラメータの値を決定

e.g. コントラスティブ・ダイバージェンス

推定フェーズ

学習により得られたモデルを用いた入出力値推定

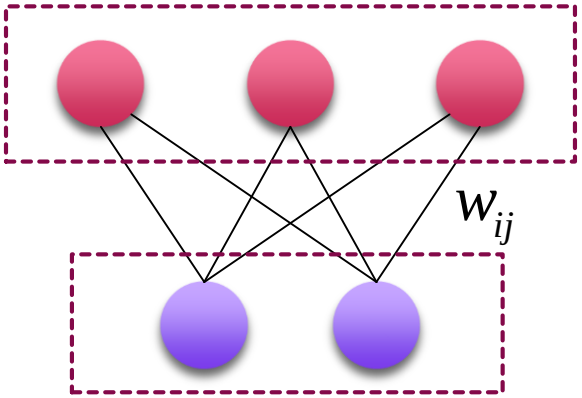
出力変数の期待値が出力値となる（MMSE推定法）

ギブスサンプリング
や
平均場近似法

目的

ガウス型制限ボルツマンマシン上での
平均場近似を基礎とした
推定アルゴリズム（期待値近似アルゴリズム）
の性能を**情報理論的に調べる**

ガウス型制限ボルツマンマシン



$$\boldsymbol{h} = \left\{ h_j \in \mathcal{X} \mid j \in H \right\}$$ 離散変数

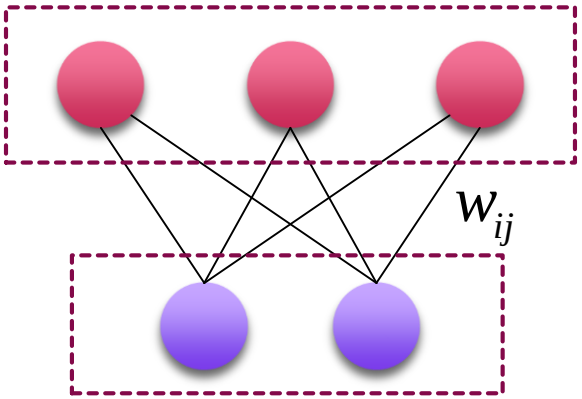
$$\boldsymbol{v} = \left\{ v_i \in (-\infty, \infty) \mid i \in V \right\}$$ 連続変数

エネルギー関数

$$E(\boldsymbol{v}, \boldsymbol{h}; \theta) := \sum_{i \in V} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j \in H} c_j h_j - \sum_{i \in V} \sum_{j \in H} \frac{w_{ij}}{\sigma_i^2} v_i h_j$$

$$P(\boldsymbol{v}, \boldsymbol{h} \mid \theta) := \frac{1}{Z(\theta)} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)\right)$$

ガウス型制限ボルツマンマシン



b_i, c_j : 可視変数，隠れ変数の
バイアスパラメータ
σ_i : 可視変数の分散パラメータ
w_{ij} : 結合パラメータ

θ

エネルギー関数

$$E(\boldsymbol{v},\boldsymbol{h};\theta) := \sum_{i\in V}\frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j\in H} c_j h_j - \sum_{i\in V}\sum_{j\in H}\frac{w_{ij}}{\sigma_i^2} v_i h_j$$

$$P(\boldsymbol{v},\boldsymbol{h}\,|\,\theta) := \frac{1}{Z(\theta)}\exp\left(-E(\boldsymbol{v},\boldsymbol{h};\theta)\right)$$

期待値計算の組み合わせ爆発

$$\mathrm{E}\left[v_i \mid \theta\right] = \int_{-\infty}^{\infty} d\boldsymbol{v} \sum_{\boldsymbol{h}\in\mathcal{X}^{|H|}} v_i P\left(\boldsymbol{v},\boldsymbol{h} \mid \theta\right), \quad \mathrm{E}\left[h_j \mid \theta\right] = \int_{-\infty}^{\infty} d\boldsymbol{v} \sum_{\boldsymbol{h}\in\mathcal{X}^{|H|}} h_j P\left(\boldsymbol{v},\boldsymbol{h} \mid \theta\right)$$

$$\int_{-\infty}^{\infty} d\boldsymbol{v}\left(\cdots\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\left(\cdots\right) dv_1 dv_2 \cdots dv_{|V|} \quad \Leftarrow \text{多重積分}$$

$$\sum_{\boldsymbol{h}\in\mathcal{X}^{|H|}}\left(\cdots\right) = \sum_{h_1\in\mathcal{X}}\sum_{h_2\in\mathcal{X}}\cdots\sum_{h_{|H|}\in\mathcal{X}}\left(\cdots\right) \quad \Leftarrow \text{多重和}$$

計算量は変数の数に対して指数的　→　NP困難

近似が必須！→ **平均場近似**

平均場近似の枠組み

多次元結合分布 $P(\boldsymbol{x}) = P(x_1, x_2, \ldots, x_n)$

に対して

テスト分布 $T(\boldsymbol{x}) = \prod_{i=1}^{n} \rho_i(x_i)$

各変数についての統計的独立性を仮定した分布（**平均場制約**）

を準備する

カルバック・ライブラー情報量

$$K[T] := \sum_{\boldsymbol{x}} T(\boldsymbol{x}) \ln \frac{T(\boldsymbol{x})}{P(\boldsymbol{x})} \geq 0$$

を最小にする T が元の P の平均場近似分布となる

平均場近似の枠組み

カルバック・ライブラー情報量は
異なる分布間の距離的な尺度

⇩

元の分布 P にもっとも近い近似分布 T が得られる

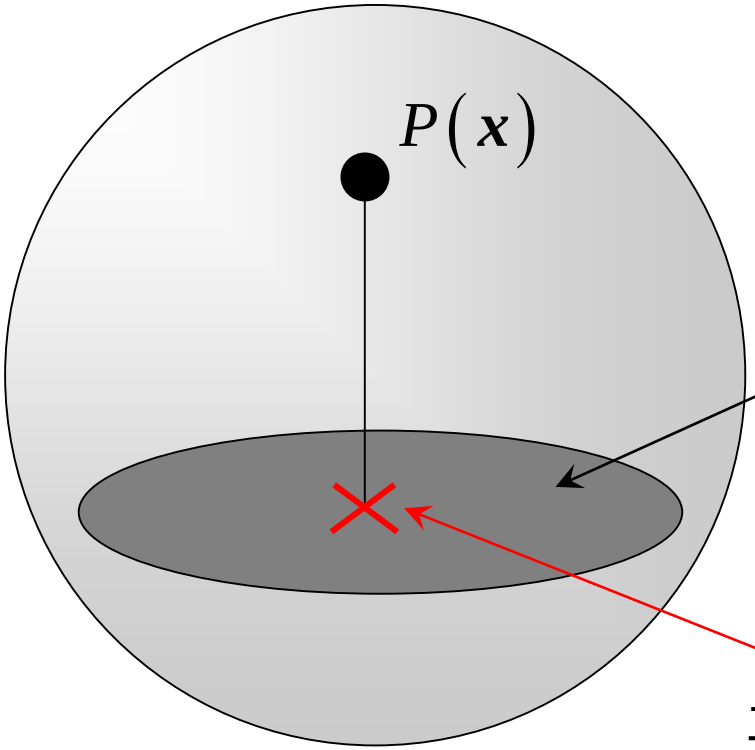
カルバック・ライブラー情報量

$$K[T] := \sum_{x} T(x) \ln \frac{T(x)}{P(x)} \geq 0$$

を最小にする T が元の P の平均場近似分布となる

カルバック・ライブラー情報量

カルバック・ライブラーダイバージェンス
カルバック・ライブラー擬距離
などとも呼ばれる



GRBM に対する平均場近似

2 種類の平均場近似が定式化可能

Type Ⅰ 平均場近似

$P(\boldsymbol{v},\boldsymbol{h}|\theta)$ に対して平均場近似

（全変数に対する平均場近似）

Type Ⅱ 平均場近似

$P(\boldsymbol{h}|\theta)=\int_{-\infty}^{\infty} d\boldsymbol{v}P(\boldsymbol{v},\boldsymbol{h}|\theta)$ に対して平均場近似

（周辺分布に対する平均場近似）

GRBM に対する平均場近似

2 種類の平均場近似が定式化可能

Type Ⅰ 平均場近似

$$K_1[T_1] := \int_{-\infty}^{\infty} d\boldsymbol{v} \sum_{\boldsymbol{h}\in\mathcal{X}^{|H|}} T_1(\boldsymbol{v},\boldsymbol{h}) \ln \frac{T_1(\boldsymbol{v},\boldsymbol{h})}{P(\boldsymbol{v},\boldsymbol{h}|\theta)}$$ の最小化

Type Ⅱ 平均場近似

$$K_2[T_2] := \int_{-\infty}^{\infty} d\boldsymbol{v} \sum_{\boldsymbol{h}\in\mathcal{X}^{|H|}} P(\boldsymbol{v}|\boldsymbol{h},\theta) T_2(\boldsymbol{h}) \ln \frac{P(\boldsymbol{v}|\boldsymbol{h},\theta) T_2(\boldsymbol{h})}{P(\boldsymbol{v},\boldsymbol{h}|\theta)}$$

の最小化

GRBM に対する平均場近似

２種類の平均場近似が定式化可能

Type Ⅰ 平均場近似

Type Ⅱ 平均場近似

計算量としては全く同等の近似計算アルゴリズムを導く

$$O(|V||H|)$$

Which method is better ??

定理（情報理論的立場）

任意の GRBM に対して

$$\min K_1[T_1] \geq \min K_2[T_2]$$

が常に成り立つ

カルバック・ライブラー情報量の観点で

Type Ⅱ 平均場近似の方が**より元の分布に近い**近似分布を与える

※この定理はより強い定理の系として導かれる
※元の定理は任意の確率モデルで上記と類似の不等式が成り立つことを主張している

定理（統計物理的立場）

$$\min K_1[T_1] = F_1(\theta) - F(\theta)$$

平均場自由エネルギー　真の自由エネルギー

$$\min K_2[T_2] = F_2(\theta) - F(\theta)$$

平均場自由エネルギー　真の自由エネルギー

任意の GRBM に対して

$$F_1(\theta) \geq F_2(\theta) \geq F(\theta)$$

が常に成り立つ

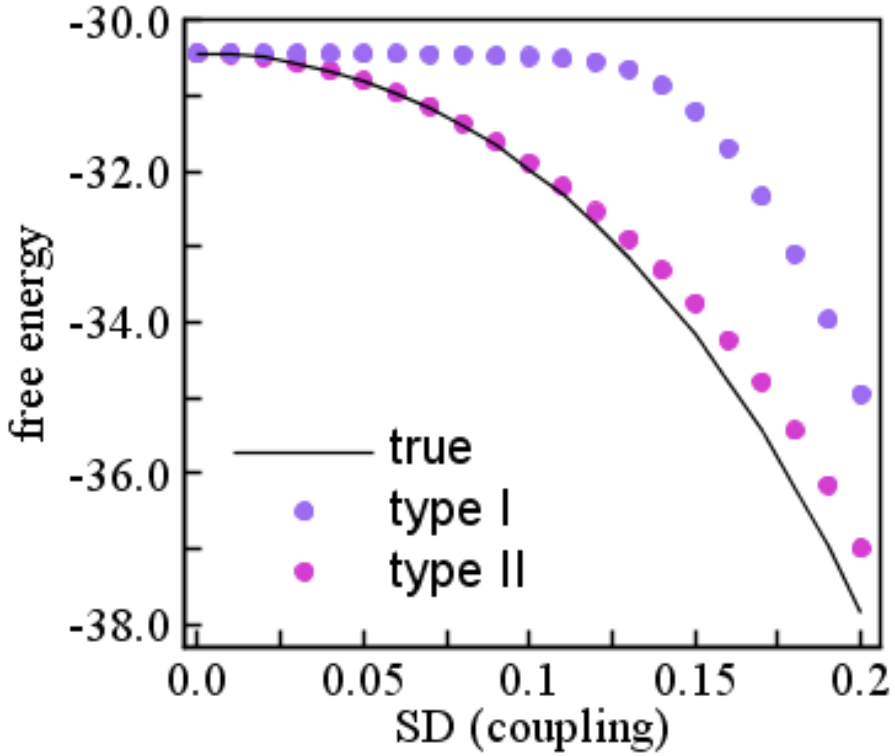
Type Ⅱ 平均場自由エネルギーの方が**真の自由エネルギーに近い**

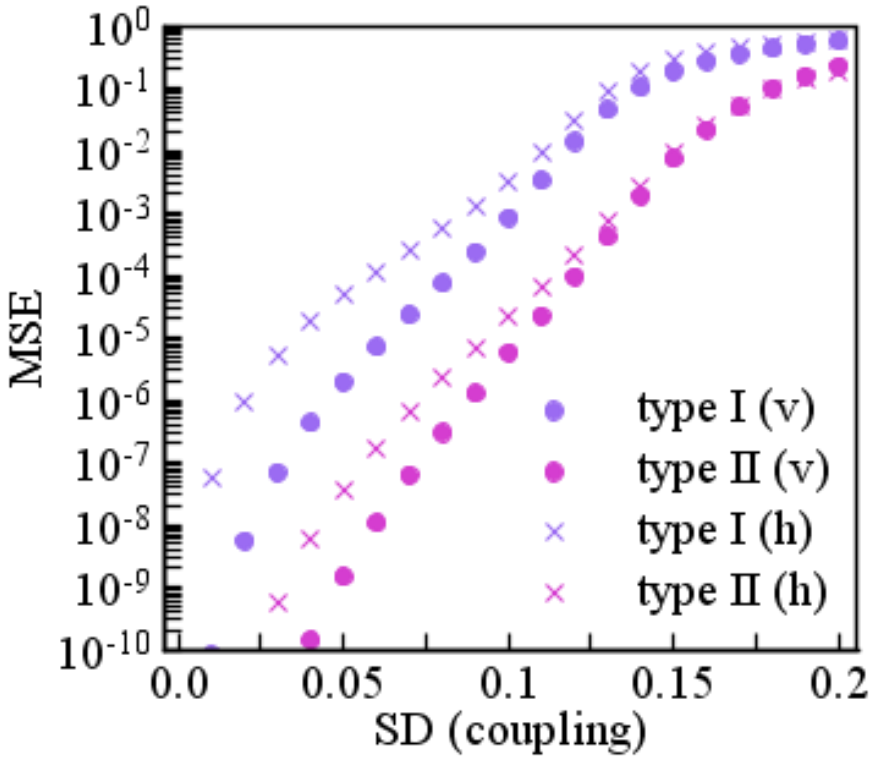
数値実験による定量的比較

$$|V| = 12,\ |H| = 24,\ \sigma_i^2 = 1$$

$$\mathcal{X} \in \{+1, -1\}$$

$b_i,\ c_j \leftarrow \mathcal{N}\left(0, 0.1^2\right)$で生成, $w_{ij} \leftarrow \mathcal{N}\left(0, \mathrm{SD}^2\right)$で生成





前半のまとめ

GRBM 上での平均場近似による推定アルゴリズム

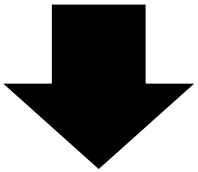
- 計算量的に同等な２種類のアルゴリズム（Type Ⅰ & Ⅱ）を定式化
- 解析的・数値的比較から Type Ⅱ の方が常に高性能であることが分かった
- 平均場法による推定アルゴリズムに対する重要な知見

Takahashi and Yasuda: *J. Phys. Soc. Jpn.*, Vol.85, 2016.

確率伝搬法

マルコフ確率場の計算処理は大変

計算量爆発の問題はどう頑張っても解決できない



真の値を得ることは諦めて近似値で良しとしましょうよ

高効率な近似計算法の発展

確率伝搬法（belief propagation; BP）

近似計算法の発展が現在の MRF 実装の楚となっている

確率伝搬法？

人工知能研究の中で生まれた計算技術 [Pearl, 82]

物性物理理論の中にも実は同様の手法がある [Bethe, 35]

Bethe 近似

情報科学分野と物理分野の狭間で発展してきた

強力な近似計算技法

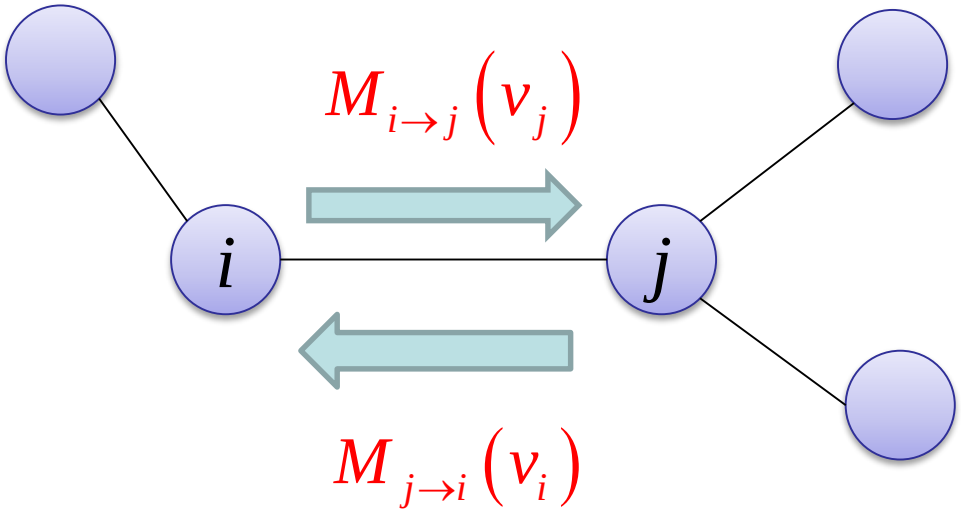
動的計画法的に再帰式を解く形式の計算アルゴリズム

確率伝搬法の作り方

標記の簡便さのため MRF を次の形式に変形しておく

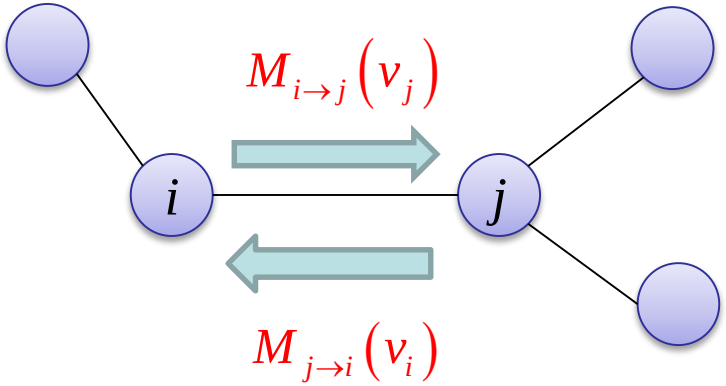
$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_{i\in V}\Phi_i(v_i) + \sum_{\{i,j\}\in E}\Psi_{\{i,j\}}(v_i, v_j)\right) = \frac{1}{Z}\prod_{i\in V}\phi_i(v_i)\prod_{\{i,j\}\in E}\varphi_{\{i,j\}}(v_i, v_j)$$

ノードからノードへの**メッセージ**と呼ばれる量を定義する



メッセージ伝搬則

- メッセージはすべてのリンク上に定義される
- メッセージは双方向に伝搬する
- メッセージは行き先側のノードの変数の関数



メッセージ伝搬則（message-passing rule）

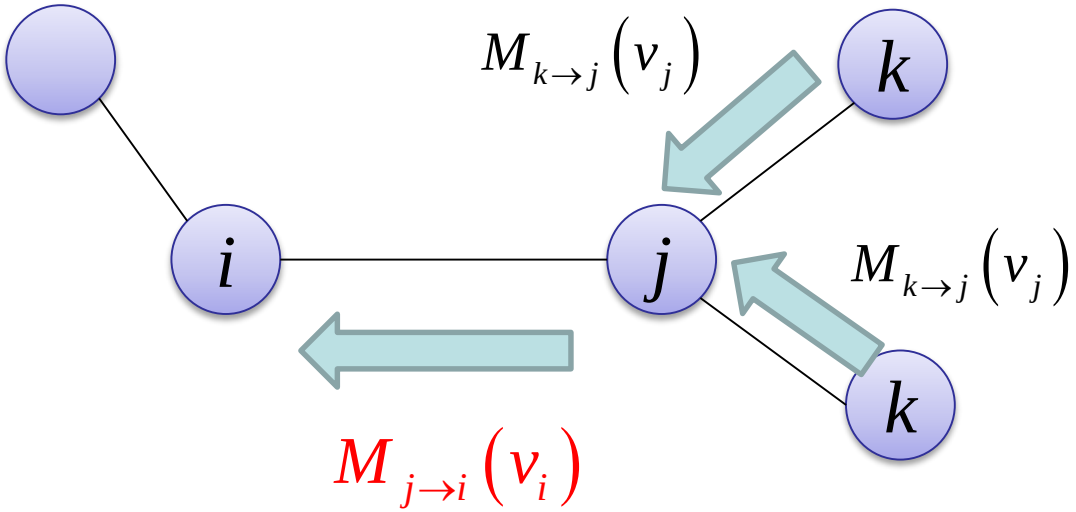
$$M_{j\to i}(v_i) \propto \sum_{x_j} \phi_j(v_j)\varphi_{\{i,j\}}(v_i, v_j) \prod_{k\in\partial(j)\setminus\{i\}} M_{k\to j}(v_j)$$

メッセージはこの連立非線形方程式を満足する必要がある

メッセージ伝搬則のグラフ的解釈

メッセージ伝搬則（message-passing rule）

$$M_{j\to i}(v_i) \propto \sum_{x_j} \phi_j(v_j)\varphi_{\{i,j\}}(v_i, v_j) \prod_{k\in\partial(j)\setminus\{i\}} M_{k\to j}(v_j)$$



行き先以外のノードからやってくる

メッセージを集めて

行き先に送る

メッセージ伝搬則の式はリンクの本数の2倍の数存在する
それを連立方程式と見立てて数値的に解く

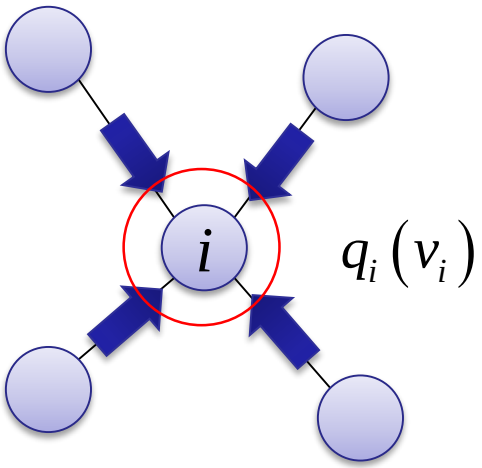
メッセージが求まったら

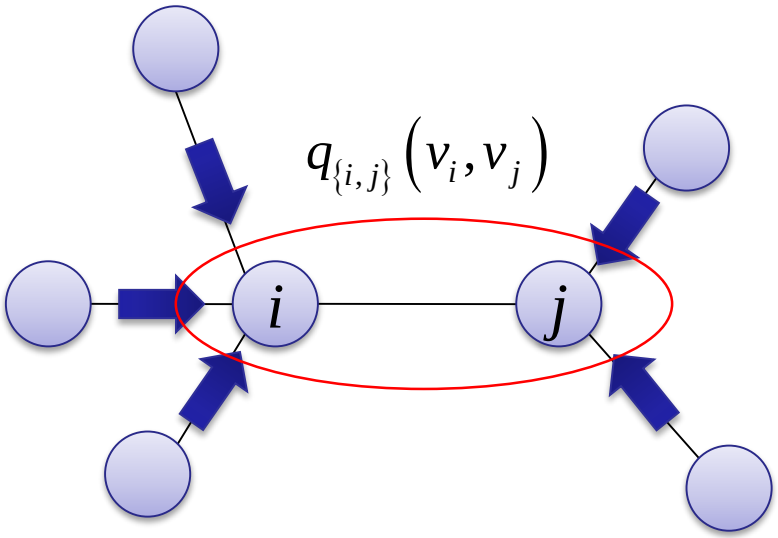
求めたメッセージを用いて**近似周辺確率**を計算する

$$q_i(v_i) = \frac{1}{z_i}\phi_i(v_i)\left(\prod_{j\in\partial(i)} M_{j\to i}(v_i)\right)$$

z_i, $z_{\{i,j\}}$ ← 規格化定数

$$q_{\{i,j\}}(v_i, v_j) = \frac{1}{z_{\{i,j\}}}\phi_i(v_i)\phi_j(v_j)\varphi_{\{i,j\}}(v_i, v_j)\left(\prod_{k\in\partial(i)\setminus\{j\}} M_{k\to i}(v_i)\right)\left(\prod_{l\in\partial(j)\setminus\{i\}} M_{l\to j}(v_j)\right)$$





確率伝搬法完了

求めた近似周辺確率を用いて**期待値**を次の要領で近似する

$$\mathrm{E}\left[f(v_i) \right] = \sum_{\boldsymbol{v}} f(v_i) P(\boldsymbol{v}) \approx \sum_{v_i} f(v_i) q_i(v_i)$$

$$\mathrm{E}\left[f(v_i, v_j) \right] = \sum_{\boldsymbol{v}} f(v_i, v_j) P(\boldsymbol{v}) \approx \sum_{v_i} \sum_{v_j} f(v_i, v_j) q_{\{i,j\}}(v_i, v_j)$$

1 変数の期待値のみでなく 2 変数の期待値も計算することができる！

確率伝搬法の手続きまとめ

メッセージ → 近似周辺確率 → 近似期待値

確率伝搬法について

グラフ構造が木構造（閉路を持たない構造）なら真の期待値を与える

リンクがスパースなグラフ上で非常に高性能

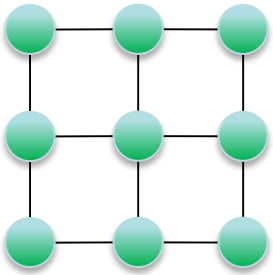
sum-product アルゴリズムという名でも知られている

確率伝搬法

が

MRF上での計算処理の基礎を成している

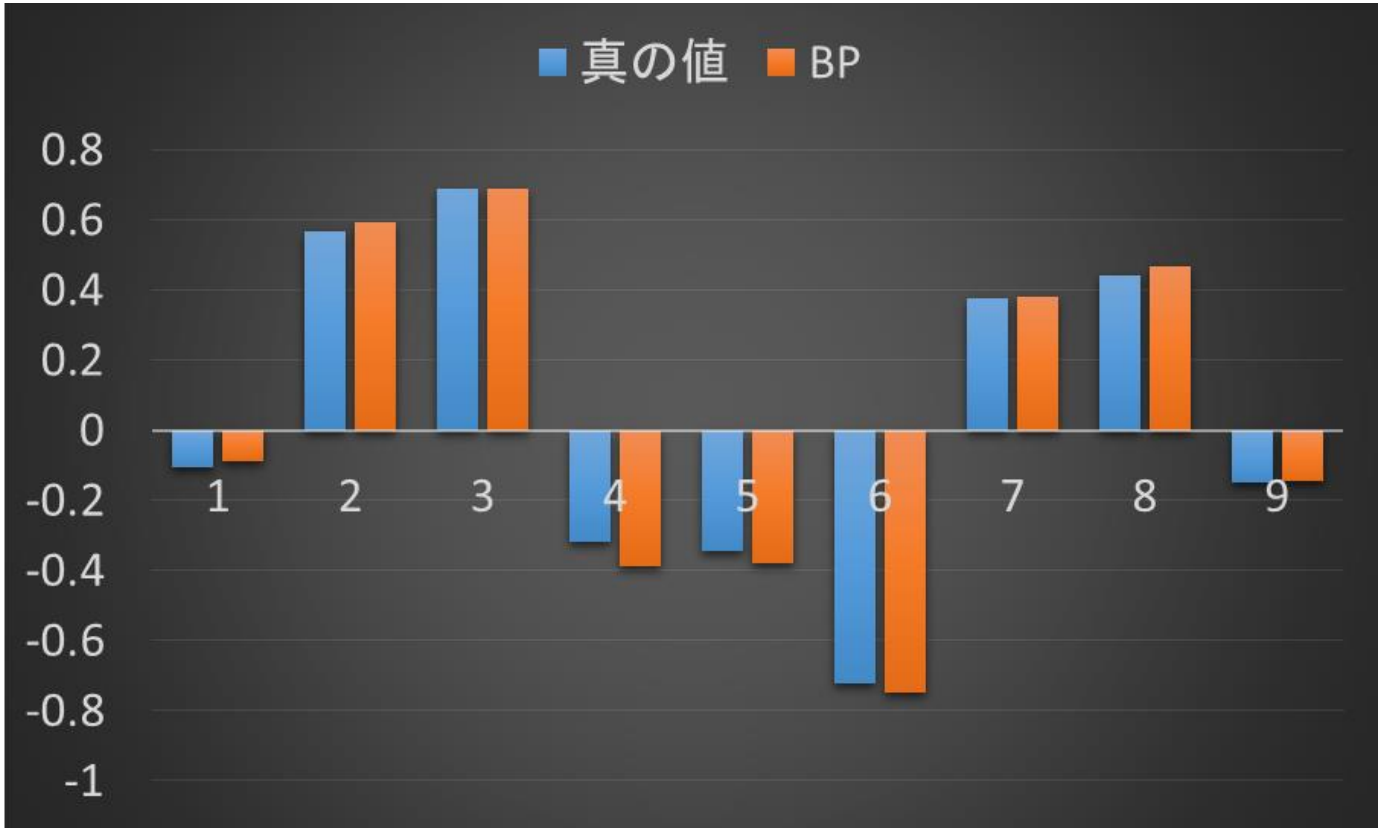
実際どんなもんか



$$P(\boldsymbol{v}\,|\,\boldsymbol{b},\boldsymbol{w}) = \frac{1}{Z}\exp\left(\sum_{i\in V} b_i v_i + \sum_{\{i,j\}\in E} w_{ij} v_i v_j\right) \quad v_i \in \{+1,-1\}$$

パラメータは平均 0 で標準偏差 0.4 のガウス乱数によりランダムに生成

$\mathrm{E}[v_i]$

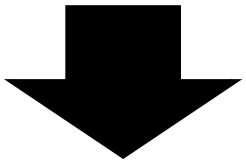


もう一つの確率伝搬法

sum-product アルゴリズムは MRF の期待値を計算する手法

確率伝搬法にはもう一つの手法がある！

MRF の確率を最大とする v の値を見つける手法



max-product アルゴリズム

max-product アルゴリズム

$$M_{j\to i}(v_i) \propto \sum_{v_j} \phi_j(v_j)\varphi_{\{i,j\}}(v_i, v_j) \prod_{k\in\partial(j)\setminus\{i\}} M_{k\to j}(v_j)$$

sum-product

max-product アルゴリズムの
メッセージ伝搬則（message-passing rule）

$$M_{j\to i}(v_i) \propto \max_{v_j} \left\{ \phi_j(v_j)\varphi_{\{i,j\}}(v_i, v_j) \prod_{k\in\partial(j)\setminus\{i\}} M_{k\to j}(v_j) \right\}$$

sum-product アルゴリズムのメッセージ伝搬則の**和**を

マックス演算に変更すると max-product アルゴリズム

max-product アルゴリズム

求めたメッセージを用いて近似最大値を計算する

$$v_i^{\mathrm{BP}} = \max_{v_i} \phi_i(v_i) \left(\prod_{j \in \partial(i)} M_{j \to i}(v_i) \right)$$

$$\boldsymbol{v}^{\mathrm{BP}} \approx \arg\max_{\boldsymbol{v}} P(\boldsymbol{v})$$

MRF を最大とする
変数の値の
近似値を得る

sum-product アルゴリズムと同様に
木構造のグラフ上なら真の値が求まる

確率伝搬法のまとめ

MRFの期待値を計算したい
→ sum-product アルゴリズム

MRFの最大値を計算したい
→ max-product アルゴリズム