D. Lecture #7. Markov-Chain Monte-Carlo.

Markov Chain Monte Carlo (MCMC) methods belong to the class of algorithms for sampling from a probability distribution based on constructing a Markov chain that converges to the target steady distribution.

Examples and flavors of MCMC are many (and some are quite similar) – heat bath, Glauber dynamics, Gibbs sampling, Metropolis Hastings, Cluster algorithm, Warm algorithm, etc – in all these cases we only need to know transition probability between states while the actual stationary distribution may be not known or, more accurately, known up to the normalization factor, also called the partition function. Below, we will discuss in details two key examples: Gibbs sampling & Metropolis-Hastings.

1. Gibbs Sampling

Assume that the direct sampling is not feasible (because there are too many variables and computations are of "exponential" complexity — more on this latter). The main point of the Gibbs sampling is that given a multivariate distribution it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution. Then we create a chain: start from a current sample of the vector x, pick a component at random, compute probability for this component/variable conditioned to the rest, and sample from this conditional distribution. (The conditional distribution is for a simple component and thus it is easy.) We continue the process till convergence, which can be identified (empirically) by checking if estimation of the histogram or observable(s) stopped changing.

Algorithm 2 Gibbs Sampling

Input: Given $p(x_i|x_{\sim i} = x \setminus x_i), \quad \forall i \in \{1, \cdots, N\}$. Start with a sample $x^{(t)}$.

loop Till convergence
- Draw an i.i.d. i from $\{1, \cdots, N\}$.
- Generate a random $x_i \sim p\left(x_i|x^{(t)}_{\sim i}\right)$.
- $x^{(t)}_i = x_i$.
- $\forall j \in \{1, \cdots, N\} \setminus i: \quad x^{(t)}_j \leftarrow x^{(t)}_j$.
- Output $x^{(t+1)}$ as the next sample.

end loop

Question (specific): Does the Gibbs sampling obey the Detailed Balance? Show it.

Question (generic): if one wants to prove convergence of an MCMC, how one can do it? (What is the expectation/average to study?)

2. Metropolis-Hastings Sampling

Metropolis-Hastings sampling is an MCMC method which explores efficiently the detailed balance, i.e. reversibility of the underlying Markov Chain. The algorithm also uses sampling from the conditional probabilities and smart use of the rejection strategy. Assume that the probability of any state x from which one wants to sample (call it the target distribution) is explicitly known up to the normalization constant, Z, i.e. $p(x) = \pi(x)/Z$, where $Z = \sum_x \pi(x)$. Let us also introduce the so-called proposal distribution, $p(x'|x)$, and assume that drawing a sample proposal x' from the current sample x is (computationally) easy.

Note that the Gibbs sampling previously introduced can be considered as the Metropolis-Hastings without rejection (thus it is a particular case).

Exercise: Arguing in terms of transitions between states show that the algorithm maintains the DB.

The proposals (conditional probabilities) may vary. Details are critical (change mixing time), especially for large system. There is a (heuristic) rule of thumb: **lower bound on number of iterations of MH**. If the largest distance between the states is L, the MH will mix in time

$$T \approx (L/\varepsilon)^2 \tag{II.41}$$

where ε is the typical step size of the random walk.

Question: How can one reason the quadratic behavior in Eq. (II.41)? What would acceleration from quadratic to linear mean? (Diffusive vs ballistic regime.)

Algorithm 3 Metropolis-Hastings Sampling

Input: Given $\pi(x)$ and $p(x'|x)$. Start with a sample x_t.

loop Till convergence
  Draw a random $x' \sim p(x'|x_t)$.
  Compute $\alpha = \frac{p(x_t|x')\pi(x')}{p(x'|x_t)\pi(x_t)}$.
  Draw random $\beta \in U([0,1])$, uniform i.i.d. from $[0,1]$.
  if $\beta < \min\{1, \alpha\}$ **then**
    $x_t \leftarrow x'$ [accept]
  else
    x' is ignored [reject]
  end if
  x_t is recordered as a new sample
end loop

Mixing may be extremely slow if the proposal distribution is not selected carefully. Let us illustrate how slow MCMC can be on a simple example. (See Section 29 of the McKay book for extra details.) Consider the following target distribution over N states

$$\pi(x) = \begin{cases} 1/N & x \in \{0, \cdots, N-1\} \\ 0 & \text{otherwise} \end{cases} \tag{II.42}$$

and proposal distribution over $N+2$ states (extended by -1 and N)

$$p(x'|x) = \begin{cases} 1/2 & x' = x \pm 1 \\ 0 & \text{otherwise} \end{cases} \tag{II.43}$$

Notice that the rejection can only occur when the proposed state is $x' = -1$ or $x' = N$.

A more sophisticated example of the Glauber algorithm (version of MH) on the example of the Ising Model is to be discussed next.

3. Glauber Sampling of Ising Model

Let us return to the special version of MH developed specifically for the Ising model - the Glauber dynamics/algorithms:

Algorithm 4 Glauber Sampling

Input: Ising model on a graph. Start with a sample σ

loop Till convergence
  Pick a node i at random.
  $-\sigma_i \leftarrow \sigma_i$
  Compute $\alpha = \exp\left(\sigma_i \left(\sum_{j \in \mathcal{V}: \{i,j\} \in \mathcal{E}} J_{ij}\sigma_j - 2h_i\right)\right)$.
  Draw random $\beta \in U([0,1])$, uniform i.i.d. from $[0,1]$.
  if $\alpha < \beta < 1$ **then**
    $-\sigma_i \leftarrow \sigma_i$ [reject]
  end if
  Output: σ as a sample
end loop

Question: What is the proposal distribution turning the MH sampling into the Glauber sampling (for the Ising model)?

Exercise [Advanced]: Consider running parallel dynamics, based on the Glauber algorithm, i.e. at every moment of time update all variables in parallel according to the Glauber Sampling rule applied to the previous state. What is the resulting stationary distribution? Is it different from the Ising model? Does the algorithm satisfy the DB conditions?

For useful additional reading on sampling and computations for the Ising model see `https://www.physik.uni-leipzig.de/~janke/Paper/lnp739_079_2008.pdf`. MCMC recitation will focus on discussion of the Glauber algorithm.

4. Exactness and Convergence

MCMC algorithm is called (casually) exact if one can show that the generated distribution "converges" to the desired stationary distribution. However, "convergence" may mean different things.

The strongest form of convergence – called **exact independence test** (warning - this is our 'custom' term) – states that at each step we generate an independent sample from the target distribution. To prove this statement means to show that empirical correlation of the consecutive samples is zero in the limit when Nnumber of samples $\to \infty$:

$$\lim_{N\to+\infty} \frac{1}{N} \sum_{n=0}^{N} \sum_{n=1}^{N} f(x_n) g(x_{n-1}) \to \mathbb{E}\left[f(x)\right] \mathbb{E}\left[g(x)\right], \tag{II.44}$$

where $f(x)$ and $g(x)$ are arbitrary functions (however such that respective expectations on the rhs of Eq. (**??**) are well-defined).

A weaker statement – call it **asymptotic convergence** – suggests that in the limit of $N \to \infty$ we reconstruct the target distribution (and all the respective existing moments):

$$\lim_{N\to+\infty} \frac{1}{N} \sum_{n=0}^{N} \sum_{n=1}^{N} f(x_n) \to \mathbb{E}\left[f(x)\right], \tag{II.45}$$

where $f(x)$ is an arbitrary function such that the expectation on the rhs is well defined.

Finally, the weakest statement – call it **parametric convergence** – corresponds to the case when one arrives at the target estimate only in a special limit with respect to a special parameter. It is common, e.g. in statistical/theoretical physics and computer sicence, to study the so-called thermodynamic limit, where the number of degrees of freedom (for example number of spins/variables in the Ising model) becomes infinite:

$$\lim_{s\to s_*} \lim_{N\to+\infty} \frac{1}{N} \sum_{n=0}^{N} \sum_{n=1}^{N} f_s(x_n) \to \mathbb{E}\left[f_{s_*}(x)\right]. \tag{II.46}$$

For additional math (but also intuitive as written for physicists) reading on the MCMC (and in general MC) convergence see `http://statweb.stanford.edu/~cgates/PERSI/papers/mixing.pdf` and also [3].

5. Exact Monte Carlo Sampling (Did it converge yet?)

(This part of the lecture is a bonus material - we discuss it only if time permits.)

The material follows Chapter 32 of D.J.C. MacKay book. Some useful set of modern references and discussions are also available at `http://dimacs.rutgers.edu/~dbwilson/exact/`.

As mentioned already the main problem with MCMC methods is that one needs to wait (and sometimes for too long) to make sure that the generated samples (from the target distribution) are i.i.d. If one starts to form a histogram (empirical distribution) too early it will deviate from the target distribution. One important question in this regards is: For how long shall one run the Markov Chain before it has 'converged'? To answer this question (prove) it is very difficult, in many cases not possible. However, there is a technique which allows to check the **exact convergence**, for some cases, and do it on the fly - as we run MCMC.

This smart technique is the Propp-Wilson exact sampling method, also called **coupling from the past**. The technique is based on a combination of three ideas:

- The main idea is related to the notion of the **trajectory coalescence**. Let us observe that if starting from different initial conditions the MCMC chains share a single random number generator, then their trajectories in the phase space can coalesce; and having coalesced, will not separate again. This is clearly an indication that the initial conditions are forgotten.

  Will running all the initial conditions forward in time till coalescence generate exact sample? Apparently not. One can show (sufficient to do it for a simple example) that the point of coalescence does not represent an exact sample.

- However, one can still achieve the goal by **sampling from a time T_0 in the past**, up to the present. If the coalescence has occurred the present sample is an unbiased sample; and if not we restart the simulation from the time T_0 further into the past, reusing the same random numbers. The simulation is repeated till a coalescence occur at a time before the present. One can show that the resulting sample at the present is exact.

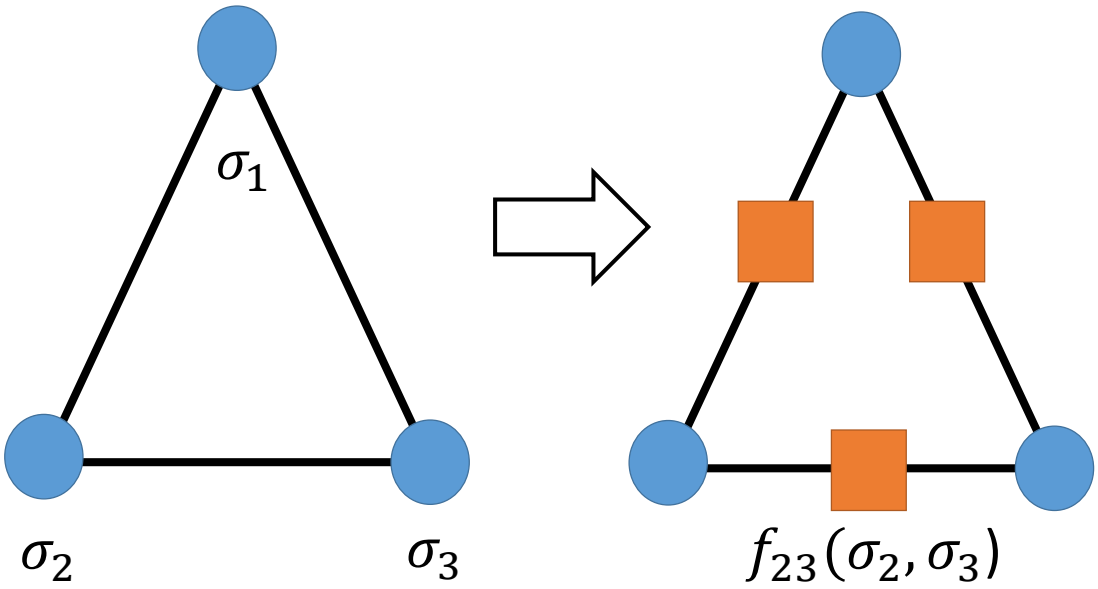

FIG. 7: Factor Graph Representation for the (simple case) with pair-wise factors only. In the case of the Ising model: $f_{12}(\sigma_1, \sigma_2) = \exp\left(-J_{12}\sigma_1\sigma_2 + h_1\sigma_1 + h_2\sigma_2\right)$.

- One problem with the scheme is that we need to test it for all the initial conditions - which are too many to track. Is there a way to **reduce the number of necessary trials**. Remarkably, it appears possible for sub-class of probabilistic models the so-called **'attractive'** models. Loosely speaking and using 'physics' jargon - these are **'ferromagnetic'** models - which are the models where for a stand alone pair of variables the preferred configuration is the one with the same values of the two variables. In the case of attractive model monotonicity (sub-modularity) of the underlying model suggests that the paths do not cross. This allows to only study limiting trajectories and deduce interesting properties of all the other trajectories from the limiting cases.

III. THEME # 3: GRAPHICAL MODELS

A. Lecture #8. Exact & Approximate Inference.

This lecture and the following lecture largely follow the materials of the mini-course on *Graphical Models of Statistical Inference: Belief Propagation & Beyond*, given at Skoltech in September of 2016 (also planned for October of 2017). See slides and lecture notes at `https://sites.google.com/site/mchertkov/courses`.

1. *From Ising Model to (Factor) Graphical Models*

Brief reminder of what we have learned so far about the Ising Model. It is fully described by Eqs. (II.37,II.38). The weight of a "spin" configuration is given by Eq. (II.37). Let us not pay much of attention for now to the normalization factor Z and observe that the weight is nicely factorized. Indeed, it is a product of pair-wise terms. Each term describes "interaction" between spins. Obviously we can represent the factorization through a graph. For example, if our spin system consists only of three spins connected to each other, then the respective graph is a triangle. Spins are associated with nodes of the graphs and "interactions", which may also be called (pair-wise) factors, are associated with edges.

It is useful, for resolving this and other factorized problems, to introduce a bit more general representation — in terms of graphs where both factors and variables are associated with nodes/vertices. Transformation to the factor-graph representation for the three spin example is shown in Fig. (7).

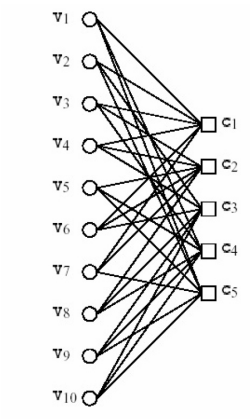

FIG. 8: Tanner graph of a linear code, represented with $N = 10$ bits, $M = 5$ checks, and $L = N - M = 5$ information bits. This code selects 2^5 codewords from 2^10 possible patterns. This adjacency, parity-check matrix of the code is given by Eq. (III.2).

Ising Model, as well as other models discussed later in the lectures, can thus be stated in terms of the general factor-graph framework/model

$$P(\sigma) = Z^{-1} \prod_{a \in \mathcal{V}_f} f_a(\sigma_a), \quad \sigma_a \doteq (\sigma_i | i \in \mathcal{V}_n, \quad (i,a) \in \mathcal{E}) \,, \tag{III.1}$$

where $(\mathcal{V}_f, \mathcal{V}_n, \mathcal{E})$ is the bi-partite graph built of factors and nodes.

The factor graph language (representation) is more general. We will see it next - discussing another interesting problem from Information Theory - decoding of error-correction codes.

2. *Decoding of Graphical Codes as a Factor Graph problem*

First, let us discuss decoding of a graphical code. (Our description here is terse, and we advise interested reader to check the book by Richardson and Urbanke [8] for more details.) A message word consisting of L information bits is encoded in an N-bit long code word, $N > L$. In the case of binary, linear coding discussed here, a convenient representation of the code is given by $M \geq N - L$ constraints, often called parity checks or simply, checks. Formally, $\boldsymbol{\varsigma} = (\varsigma_i = 0, 1 | i = 1, \cdots, N)$ is one of the 2^L code words iff $\sum_{i \sim \alpha} \varsigma_i = 0 \ (\mathrm{mod}\ 2)$ for all checks $\alpha = 1, \cdots, M$, where $i \sim \alpha$ if the bit if the bit i contributes the check α, and $\alpha \sim i$ will indicate that the check α contains bit i. The relation between bits and checks is often described in terms of the $M \times N$ parity-check matrix $\boldsymbol{H}$ consisting of ones and zeros: $H_{i\alpha} = 1$ if $i \sim \alpha$ and $H_{i\alpha} = 0$ otherwise. The set of the codewords is thus defined as $\Xi^{(\mathrm{CW})} = (\varsigma | \boldsymbol{H}\boldsymbol{\varsigma} = \mathbf{0} \ (\mathrm{mod}\ 2))$. A bipartite graph representation of $\boldsymbol{H}$, with bits marked as circles, checks marked as squares, and edges corresponding to respective nonzero elements of $\boldsymbol{H}$, is usually called (in the coding theory) the Tanner graph of the code, or parity-check graph of the code. (Notice that, fundamentally, code is defined in terms of the set of its codewords, and there are many parity check matrixes/graphs parameterizing the code. We ignore this unambiguity here, choosing one convenient parametrization $\boldsymbol{H}$ for the code.) Therefore the bi-partite Tanner graph of the code is defined as $\mathcal{G} = (\mathcal{G}_0, \mathcal{G}_1)$, where the set of nodes is the union of the sets associated with variables and checks, $\mathcal{G}_0 = \mathcal{G}_{0;v} \cup \mathcal{G}_{0;e}$ and only edges connecting variables and checks contribute $\mathcal{G}_1$.

For a simple example with 10 bits and 5 checks, the parity check (adjacency) matrix of the code with the Tanner graph shown in Fig. (8) is

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} . \tag{III.2}$$

Another example of a bigger code and respective parity check matrix are shown in Fig. (9). For this example, $N = 155$, $L = 64$, $M = 91$ and the Hamming distance, defined as the minimum l_0-distance between two distinct codewords, is 20.

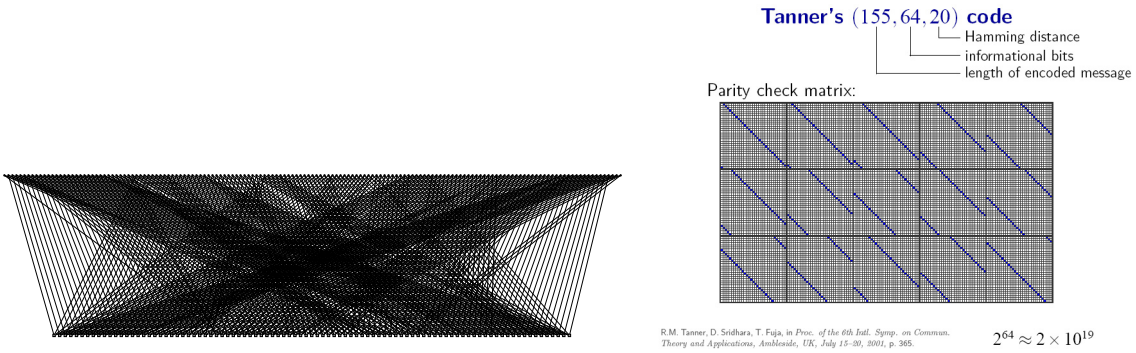

FIG. 9: Tanner graph and parity check matrix of the $(155, 64, 20)$ Tanner code, where $N = 155$ is the length of the code (size of the code word), $L = 64$ and the Hamming distance of the code, $d = 20$.

Assume that each bit of the transmitted signal is changed (effect of the channel noise) independently of others. It is done with some known conditional probability, $p(x|\sigma)$, where $\sigma = 0, 1$ is the valued of the bit before transmission, and x is its changed/distorted image. Once $\boldsymbol{x} = (x_i | i = 1, \cdots, N)$ is measured, the task of the Maximum-A-Posteriori (MAP) decoding becomes to reconstruct the most probable codeword consistent with the measurement:

$$\boldsymbol{\sigma}^{(MAP)} = \arg \min_{\boldsymbol{\sigma} \in \Xi^{(\mathrm{CW})}} \prod_{i=1}^{N} p(x_i|\sigma_i). \tag{III.3}$$

More generally, the probability of a codeword $\boldsymbol{\varsigma} \in \Xi^{(\mathrm{CW})}$ to be a pre-image for $\boldsymbol{x}$ is

$$\mathcal{P}(\boldsymbol{\sigma}|\boldsymbol{x}) = (Z(\boldsymbol{x}))^{-1} \prod_{i \in \mathcal{G}_{0;v}} g^{(\mathrm{channel})}(x_i|\varsigma_i), \quad Z(\boldsymbol{x}) = \sum_{\boldsymbol{\varsigma} \in \Xi^{(\mathrm{CW})}} \prod_{i \in \mathcal{G}_{0;v}} g^{(\mathrm{channel})}(x_i|\varsigma_i), \tag{III.4}$$

where $Z(\boldsymbol{x})$ is thus the partition function dependent on the detected vector $\boldsymbol{x}$. One may also consider the signal (bit-wise) MAP decoder

$$\forall\, i: \quad \varsigma_i^{(s-MAP)} = \arg \max_{\varsigma_i} \sum_{\boldsymbol{\varsigma} \in \Xi^{(\mathrm{CW})}}^{\boldsymbol{\varsigma} \setminus \varsigma_i} \mathcal{P}(\boldsymbol{\varsigma}|\boldsymbol{x}). \tag{III.5}$$

3. *Partition Function. Marginal Probabilities. Maximum Likelihood.*

The partition function in Eq. (III.1) is the normalization factor

$$Z = \sum_{\sigma} \prod_{a \in \mathcal{V}_f} f_a(\sigma_a), \quad \sigma_a \doteq (\sigma_i | i \in \mathcal{V}_n, \quad (i, a) \in \mathcal{E}), \tag{III.6}$$

where $\sigma = (\sigma_i \in \{0, 1\} \in \mathcal{V}_n)$. Here, we assume that the alphabet of the elementary random variable is binary, however generalization to the case of a higher alphabet is straightforward.

We are interested to 'marginalize' Eq. (III.1) over a subset of variables, for example over all the elementary/nodal variable but one

$$P(\sigma_i) \doteq \sum_{\sigma \setminus \sigma_i} P(\sigma). \tag{III.7}$$

Expectation of σ_i computed with the probability Eq. (III.7) is also called (in physics) 'magnetization' of the variable.

Exercise: Does a partition function oracle sufficient for computing $P(\sigma_i)$?

Exercise: What is the relation in the case of the Ising model between $P(\sigma_i)$ and $Z(h)$?

Another object of interest is the so-called Maximum Likelihood. Stated formally, is the most probable state of all represented in Eq. (III.1):

$$\sigma_* = \arg \max_{\sigma} P(\sigma). \tag{III.8}$$

Exercise: Consider Ising model at a temperature, T, where $J \to J/T$ and $h \to h/T$ in Eq. (II.37). How can one extract ML (III.8) from $Z(T)$?

All these objects are difficult to compute. "Difficulty" - still stated casually - means that the number of operations needed is exponential in the system size (e.g. number of variables/spins in the Ising model). This is in general, i.e. for a GM of a general position. However, for some special cases, or even special classes of cases, the computations may be much easier than in the worst case. Thus, ML (III.8) for the case of the so-called ferromagnetic (attractive, sub-modular) Ising model can be computed with efforts polynomial in the system size. Note that the partition function computation (at any nonzero temperatures) is still exponential even in this case, thus illustrating the general statement - computing Z or $P(\sigma_i)$ is a more difficult problem than computing σ_*.

A curious fact. Ising model (ferromagnetic, anti-ferromagnetic or glassy) when the "magnetic field" is zero, $h = 0$, and the graph is planar, represents a very unique class of problems for which even computations of Z are easy. In this case the partition function is expressed via determinant of a finite matrix, while computing determinant of a size N matrix is a problem of $O(N^3)$ complexity (actually $O(N^{3/2})$ in the planar case).

In the general (difficult) case we will need to relay on approximations to make computations scalable. And some of these approximations will be discussed later in the lecture. However, let us first prepare for that - restating the most general problem discussed so far – computation of the partition function, Z – as an optimization problem.

4. Kublack-Leibler Formulation & Probability Polytope

We will go from counting (computing partition function is the problem of weighted counting) to optimization by changing description from states to probabilities of the states, which we will also call beliefs. $b(\sigma)$ will be a belief - which is our probabilistic guess - for the probability of state σ. Consider it on the example of the triangle system shown in Fig. (7). There are 2^3 states in this case: $(\sigma_1 = \pm 1, \sigma_2 = \pm 1, \sigma_3 = \pm 1)$, which can occur with the probabilities, $b(\sigma_1, \sigma_2, \sigma_3)$. All the beliefs are positive and together should some to unity. We would like to compare a particular assignment of the beliefs with $P(\sigma)$, generally described by Eq. (III.1). Let us recall a tool which we already used to compare probabilities - the Kublack-Leibler (KL) divergence (of probabilities) discussed in Lecture #2:

$$D(b\|P) = \sum_{\sigma} b(\sigma) \log\left(\frac{b(\sigma)}{P(\sigma)}\right) \tag{III.9}$$

Note that the KL divergence (III.9) is a convex function of the beliefs (remember, there are 2^3 of the beliefs in the our enabling three node example) within the following polytope – domain in the space of beliefs bounded by linear constraints:

$$\forall \sigma: \quad b(\sigma) \geq 0, \tag{III.10}$$

$$\sum_{\sigma} b(\sigma) = 1. \tag{III.11}$$

Moreover, it is straightforward to check (please do it at home!) that the unique minimum of $D(b\|P)$ is achieved at $b = P$, where the KL divergence is zero:

$$P = \arg\min_{b} D(b\|P), \quad \min_{b} D(b\|P) = 0. \tag{III.12}$$

Substituting Eq. (III.1) into Eq. (III.12) one derives

$$\log Z = -\min_{b} \mathcal{F}(b), \quad \mathcal{F}(b) \doteq \sum_{\sigma} b(\sigma) \log\left(\frac{\prod_a f_a(\sigma_a)}{b(\sigma)}\right), \tag{III.13}$$

where $F(b)$, considered as a function of all the beliefs, is called (configurational) free energy (where configuration is one of the beliefs). The terminology originates from statistical physics.

To summarize, we did manage to reduce counting problem to an optimization problem. Which is great, however so far it is just a reformulation – as the number of variational degrees of freedom (beliefs) is as much as the number of terms in the original sum (the partition function). Indeed, it is not the formula itself but (as we will see below) its further use for approximations which will be extremely useful.

5. Variational Approximations. Mean Field.

The main idea is to reduce the search space from exploration of the $2^N - 1$ dimensional beliefs to their lower dimensional, i.e. parameterized with fewer variables, proxy/approximation. What kind of factorization can one suggest for the multivariate

(N-spin) probabilities/belliefs? The idea of postulating independence of all the N variables/spins comes to mind:

$$b(\sigma) \to b_{MF}(\sigma) = \prod_i b_i(\sigma_i) \tag{III.14}$$

$$\forall i \in \mathcal{V}_i, \;\; \forall \sigma_i : \quad b_i(\sigma_i) \geq 0 \tag{III.15}$$

$$\forall i \in \mathcal{V}_i : \quad \sum_{\sigma_i} b_i(\sigma_i) = 1. \tag{III.16}$$

Clearly $b_i(\sigma_i)$ is interpreted within this substitution as the single-node marginal belief (estimate for the single-node marginal probability).

Substituting b by b_{MF} in Eq. (III.13) one arrives at the MF estimation for the partition function

$$\log Z_{mf} = -\min_{b_{mf}} \mathcal{F}(b_{mf}), \quad \mathcal{F}(b_{mf}) \doteq \sum_a \sum_{\sigma_a} \left(\prod_{i \sim a} b_i(\sigma_i) \right) \log f_a(\sigma_a) - \sum_i \sum_{\sigma_i} b_i(\sigma_i) \log(b_i(\sigma_i)). \tag{III.17}$$

Exercise: Show that $Z_{mf} \geq Z$. Exercise: Show that $\mathcal{F}(b_{mf})$ is a convex function of its (vector) argument.

To solve the variational problem (III.17) constrained by Eqs. (III.14,III.15,III.16) is equivalent to searching for the (unique) stationary point of the following MF Lagrangian

$$\mathcal{L}(b_{mf}) \doteq \mathcal{F}(b_{mf}) + \sum_i \lambda_i \sum_{\sigma_i} b_i(\sigma_i) \tag{III.18}$$

Exercise: Write down equations defining the stationary point of $\mathcal{L}(b_{mf})$. Suggest an iterative algorithm converging to the stationary point.

The fact that Z_{mf} gives an upper bound on Z is a good news. However, in general the approximation is very crude, i.e. the gap between the bound and the actual value is large. The main reason for that is clear - by assuming that the variables are independent we have ignored significant correlations.

In the next lecture we will analyze what, very frequently, provides a much better approximation for ML inference - the so called Belief Propagation approach.