Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.438 Algorithms For Inference
Fall 2014

Recitation 9: Loopy BP

1 General Comments

1. In terms of implementation, loopy BP is not different from sum-product for trees. In other words, you can take your implementation of sum-product algorithm for trees and apply it directly to a loopy graph. This is because sum-product/BP is a distributed, local algorithm: each node receives messages from its neighbours, does some local computation, and sends out messages to its neighbours. The nodes only interact with its neighbourhood and have no idea if the overall graph is loopy or not.

   So implementation of loopy BP is easy, the difficult part is analysing it. Recall that Sum-Product algorithm for trees is guaranteed to converge after a certain number of iterations, and the resulting estimates of node marginals are accurate. For loopy BP, this is no longer the case: the algorithm might not converge at all; even if it does converge, it doesn't necessarily result in correct estimates of marginals. The lecture notes asked three questions:

   First of all, is there any fixed point at all? Notice that if the algorithm does converge, it will converge to a fixed point by definition of a fixed point. The answer to this question is *yes*, assuming some regularity conditions (e.g. continuous functions, compact sets etc).

   Secondly, what are these fixed points? The short answer is that *these fixed points are in 1-1 correspondence to the extrema (i.e. maxima or minima) of the corresponding Bethe Approximation Problem.* We'll discuss how to formulate the Bethe Approximation Problem in more details later.

   Thirdly, will the loopy BP algorithm converge to a fixed point given a certain initialization? Or is there any initialization that makes the algorithm converge? Unfortunately, the answer is *not sure* in general. But for some special cases, e.g. graphs with one single loop, analysis is possible. In lecture, we introduced computation trees and attractive fixed points, both are methods for analysing convergence behaviour for loopy BP in special cases.

2. *Computing marginals and computing partition function Z are equivalent.* Both tasks are NP-hard for general distributions. But if we have an oracle that computes marginals for any given distribution, we've proved in last homework (Problem 5.3) that we can design a polynomial time algorithm that uses this oracle to compute partition function Z. Conversely, if we have an oracle that can compute partition function for any given distribution, we can use it to compute marginals in polynomial time as

well. Consider a distribution $p_{\mathbf{x}}(\mathbf{x})$, notice

$$p_{x_1}(x_1) = \sum_{x_2,...,x_n} p_{\mathbf{x}}(x_1, ..., x_n) = \frac{1}{Z} \sum_{x_2,...,x_n} p^*_{\mathbf{x}}(x_1, ..., x_n)$$

and $\frac{p^*_{\mathbf{x}}(x_1,...,x_n)}{\sum\limits_{x_2,...,x_n} p^*_{\mathbf{x}}(x_1,...,x_n)}$ is a distribution over variables $x_2, ..., x_n$. Thus the oracle can compute Z and $\sum\limits_{x_2,...,x_n} p^*_{\mathbf{x}}(x_1, ..., x_n)$ for us, and we can get the marginal distribution over x_1 in polynomial time.

In inference, marginal distributions usually make more sense. But in statistical physics community, where a large body of work on loopy BP algorithm comes from, log partition function is very important. In fact, many quantities, e.g. free energy, free entropy, internal energy etc, are defined in terms of log(Z).

2 Details

2.1 Variational Characterization of log(Z)

Remember our original goal is to compute log(Z) for a given distribution $p_{\mathbf{x}}(\mathbf{x})$. However, we will convert the problem into an equivalent optimization problem

$$\log(Z) = \sup_{\mu \in \mathcal{M}} F(\mu)$$

where $\mathcal{M}$ is the set of all distributions over $\mathbf{x}$. We'll define $F(\mu)$ and derive the conversion in a minute but there is a fancy term for converting a computation problem into an optimization problem: 'variational characterization'.

Let us first re-write $p_{\mathbf{x}}(\mathbf{x})$ as $p_{\mathbf{x}}(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x}))$. Given any distribution, you can always re-write it in this form (known as Boltzmann Distribution) by choosing the right $E(\mathbf{x})$. $E(\mathbf{x})$ has a physical meaning. If we think of each configuration $\mathbf{x}$ as a state, $E(\mathbf{x})$ is the energy corresponding to the state. The distribution indicates that a system is less likely to be in a state that has higher energy. Taking log on both sides and rearrange, we get

$$E(\mathbf{x}) = -\log(p_{\mathbf{x}}(\mathbf{x})) - \log(Z)$$

Now we define Bethe free entropy $F(\mu)$ as

$$F(\mu) = -\sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log(\mu(\mathbf{x})) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) E(\mathbf{x})$$

Notice the first term is the entropy of distribution $\mu(\mathbf{x})$ and the second term is the average energy with respect to distribution $\mu(\mathbf{x})$. Replace $E(\mathbf{x})$ with $p_{\mathbf{x}}(\mathbf{x})$ and $\log(Z)$, we get

$$\begin{aligned}
F(\mu) &= -\sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})\log(\mu(\mathbf{x})) - \sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})E(\mathbf{x}) \\
&= -\sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})\log(\mu(\mathbf{x})) + \sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})(\log(p_{\mathbf{x}}(\mathbf{x})) + \log(Z)) \\
&= -\sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})\log(\mu(\mathbf{x})) + \sum_{\mathbf{x}\in\mathcal{X}^N} \mu(\mathbf{x})\log(p_{\mathbf{x}}(\mathbf{x})) + \log(Z) \\
&= -D(\mu(\mathbf{x})||p_{\mathbf{x}}(\mathbf{x})) + \log(Z)
\end{aligned}$$

where $D(\cdot||\cdot)$ is the KL-divergence and thus always non-negative.

$\therefore \log(Z) \le F(\mu)$ and equality is achieved if and only if $\mu(\mathbf{x}) = p_{\mathbf{x}}(\mathbf{x})$

$$\therefore \log(Z) = \sup_{\mu\in\mathcal{M}} F(\mu)$$

Notice this optimization problem is still NP-hard for general distributions. It can be made computationally tractable if we're willing to optimize over a different set instead of $\mathcal{M}$. For instance, the basic idea of *mean field approximation* is to look at a subset of $\mathcal{M}$ whose elements are distributions under which $x_1, ..., x_n$ are independent. The resulting $\log(Z_{MF})$ is a lower bound of the actual $\log(Z)$. In Bethe approximation problem, we maximize $F(\mu)$ over a set $\hat{\mathcal{M}}$ that is neither a subset nor a superset of $\mathcal{M}$. We will discuss $\hat{\mathcal{M}}$ in more details in next section.

2.2 Locally Consistent Marginals

Recall that a distribution $p_{\mathbf{x}}(\mathbf{x})$ can be factorized in the form

$$p_{\mathbf{x}}(\mathbf{x}) = (\prod_{i\in V} p_{x_i}(x_i))(\prod_{(i,j)\in\mathcal{E}} \frac{p_{x_i,x_j}(x_i,x_j)}{p_{x_i}(x_i)p_{x_j}(x_j)})$$

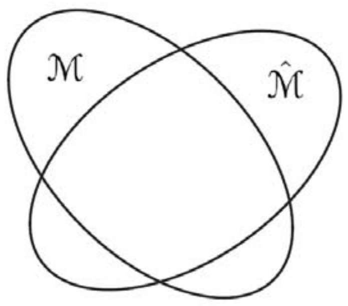
if and only if the distribution factorizes according to a tree graph.

The main idea of Bethe Approximation is instead of optimizing over the set of all distributions $\mathcal{M}$, we optimize $F(\mu)$ over $\hat{\mathcal{M}}$, which is defined as:

$$\hat{\mathcal{M}} = \{\mu_{\mathbf{x}}(\mathbf{x}) = (\prod_{i\in V} \mu_{x_i}(x_i))(\prod_{(i,j)\in\mathcal{E}} \frac{\mu_{x_i,x_j}(x_i,x_j)}{\mu_{x_i}(x_i)\mu_{x_j}(x_j)})\}$$

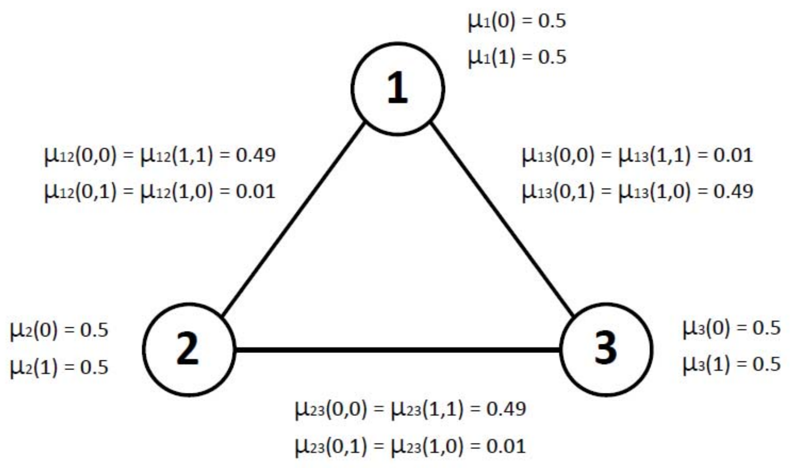
where $\{\mu_{x_i}(x_i)\}_{i\in V}$ and $\{\mu_{x_i,x_j}(x_i,x_j)\}_{(i,j)\in\mathcal{E}}$ are a set of locally consistent marginals, in other words:

$$\mu_{x_i}(x_i) \geq 0, \forall x_i$$
$$\mu_{x_i,x_j}(x_i, x_j) \geq 0, \forall x_i, x_j$$
$$\sum_{x_i} \mu_{x_i}(x_i) = 1, \forall i$$
$$\sum_{x_i} \mu_{x_i,x_j}(x_i, x_j) = \mu_{x_j}(x_j)$$
$$\sum_{x_j} \mu_{x_i,x_j}(x_i, x_j) = \mu_{x_i}(x_i)$$



In lecture, we discussed that $\hat{\mathcal{M}}$ is neither a superset nor a subset of $\mathcal{M}$, thus $\log(Z_{Bethe})$ is neither an upper bound nor a lower bound on $\log(Z)$.

1. There are elements in $\mathcal{M}$ that are not in $\hat{\mathcal{M}}$. Consider a distribution that factorizes according to a loopy graph.
2. There are elements in $\hat{\mathcal{M}}$ that are not in $\mathcal{M}$. Check out the following example.


$\mu_1(0) = 0.5$
$\mu_1(1) = 0.5$
1
$\mu_{12}(0,0) = \mu_{12}(1,1) = 0.49$
$\mu_{12}(0,1) = \mu_{12}(1,0) = 0.01$
$\mu_{13}(0,0) = \mu_{13}(1,1) = 0.01$
$\mu_{13}(0,1) = \mu_{13}(1,0) = 0.49$
$\mu_2(0) = 0.5$
$\mu_2(1) = 0.5$
2
3
$\mu_3(0) = 0.5$
$\mu_3(1) = 0.5$
$\mu_{23}(0,0) = \mu_{23}(1,1) = 0.49$
$\mu_{23}(0,1) = \mu_{23}(1,0) = 0.01$

It is easy to check that the $\{\mu_{x_i}(x_i)\}_{i\in V}$ and $\{\mu_{x_i,x_j}(x_i,x_j)\}_{(i,j)\in\mathcal{E}}$ are a set of locally consistent marginals. We claim they do not correspond to any actual distribution. Let us assume that they are the local marginals computed from some distribution q_{x_1,x_2,x_3}. Then we have

$$\begin{aligned}
q(0,0,0)+q(0,1,0) &= \mu_{13}(0,0)=0.01\\
q(0,0,1)+q(1,0,1) &= \mu_{23}(0,1)=0.01\\
q(0,0,0)+q(0,0,1) &= \mu_{12}(0,0)=0.49
\end{aligned}$$

Since q is a distribution, all q_{x_1,x_2,x_3} terms are non-negative. Thus the first two equations indicate $q(0,0,0)\leq 0.01$ and $q(0,0,1)\leq 0.01$. So the third equation cannot hold.

$\therefore$ The assumption is false and the set of locally consistent marginals does not correspond to any actual distribution.

2.3 Computation Trees

As mentioned before, computation tree is a method to analyse behaviour of Loopy BP in special cases such as graphs with a single loop. In this section, we will look at an example to get an idea how this method can be applied. Consider the graph below, and its computation tree rooted at node 1 and run for 4 iterations.

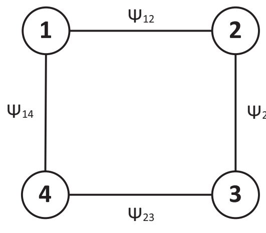


Figure 1: Original Graph

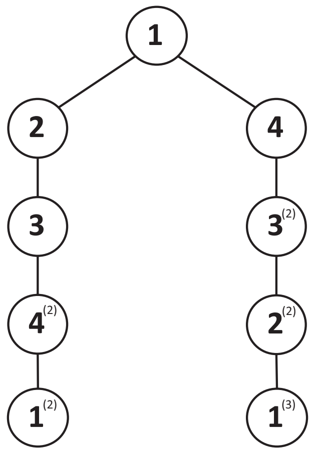


Figure 2: Computation Tree Rooted at Node 1, for 4 Iterations

Notice the messages satisfy the following equations:

$$\begin{pmatrix} m_{2\to1}^{(t)}(0) \\ m_{2\to1}^{(t)}(1) \end{pmatrix} = \begin{pmatrix} \psi_{23}(0,0) & \psi_{23}(1,0) \\ \psi_{23}(0,1) & \psi_{23}(1,1) \end{pmatrix} \begin{pmatrix} m_{3\to2}^{(t-1)}(0) \\ m_{3\to2}^{(t-1)}(1) \end{pmatrix} = A_{23} \begin{pmatrix} m_{3\to2}^{(t-1)}(0) \\ m_{3\to2}^{(t-1)}(1) \end{pmatrix}$$
$$\begin{pmatrix} m_{3\to2}^{(t-1)}(0) \\ m_{3\to2}^{(t-1)}(1) \end{pmatrix} = \begin{pmatrix} \psi_{34}(0,0) & \psi_{34}(1,0) \\ \psi_{34}(0,1) & \psi_{34}(1,1) \end{pmatrix} \begin{pmatrix} m_{4\to3}^{(t-2)}(0) \\ m_{4\to3}^{(t-2)}(1) \end{pmatrix} = A_{34} \begin{pmatrix} m_{4\to3}^{(t-2)}(0) \\ m_{4\to3}^{(t-2)}(1) \end{pmatrix}$$
$$\begin{pmatrix} m_{4\to3}^{(t-2)}(0) \\ m_{4\to3}^{(t-2)}(1) \end{pmatrix} = \begin{pmatrix} \psi_{14}(0,0) & \psi_{14}(0,1) \\ \psi_{14}(1,0) & \psi_{14}(1,1) \end{pmatrix} \begin{pmatrix} m_{1\to4}^{(t-3)}(0) \\ m_{1\to4}^{(t-3)}(1) \end{pmatrix} = A_{14}^T \begin{pmatrix} m_{1\to4}^{(t-3)}(0) \\ m_{1\to4}^{(t-3)}(1) \end{pmatrix}$$
$$\begin{pmatrix} m_{1\to4}^{(t-3)}(0) \\ m_{1\to4}^{(t-3)}(1) \end{pmatrix} = \begin{pmatrix} \psi_{12}(0,0) & \psi_{12}(1,0) \\ \psi_{12}(0,1) & \psi_{12}(1,1) \end{pmatrix} \begin{pmatrix} m_{2\to1}^{(t-4)}(0) \\ m_{2\to1}^{(t-4)}(1) \end{pmatrix} = A_{12} \begin{pmatrix} m_{2\to1}^{(t-4)}(0) \\ m_{2\to1}^{(t-4)}(1) \end{pmatrix}$$

$$\therefore \begin{pmatrix} m_{2\to1}^{(t)}(0) \\ m_{2\to1}^{(t)}(1) \end{pmatrix} = A_{23}A_{34}A_{14}^T A_{12} \begin{pmatrix} m_{2\to1}^{(t-4)}(0) \\ m_{2\to1}^{(t-4)}(1) \end{pmatrix} = M \begin{pmatrix} m_{2\to1}^{(t-4)}(0) \\ m_{2\to1}^{(t-4)}(1) \end{pmatrix}$$

In other words, the message from 2 to 1 at iteration t can be written as the product of a matrix and the message from 2 to 1 at iteration t-4. Thus we have

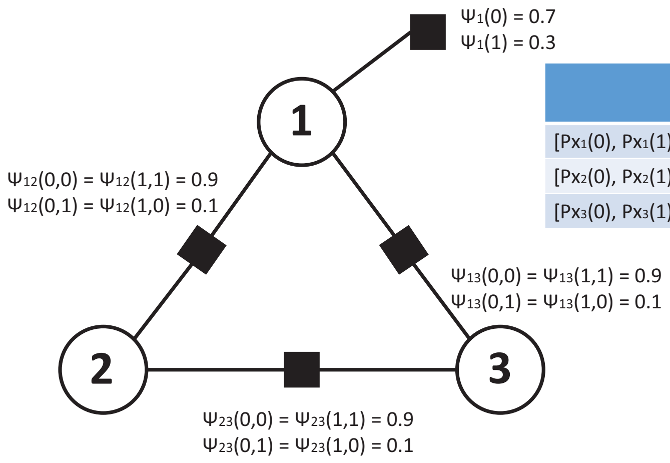
$$\begin{pmatrix} m_{2\to1}^{(4t)}(0) \\ m_{2\to1}^{(4t)}(1) \end{pmatrix} = M^t \begin{pmatrix} m_{2\to1}^{(0)}(0) \\ m_{2\to1}^{(0)}(1) \end{pmatrix}$$

Notice M is a positive matrix (i.e. each entry is positive). Apply Perron-Frobenious theorem, as $t \to \infty$, $M^t \to \lambda_1^k \mathbf{v}_1 \mathbf{u}_1^T$, where λ_1 is the largest eigenvalue of M, and v_1 and u_1 are the right and left eigenvalue of M corresponding to λ_1. Thus message from 2 to 1 will converge when number of iteration goes to infinity.

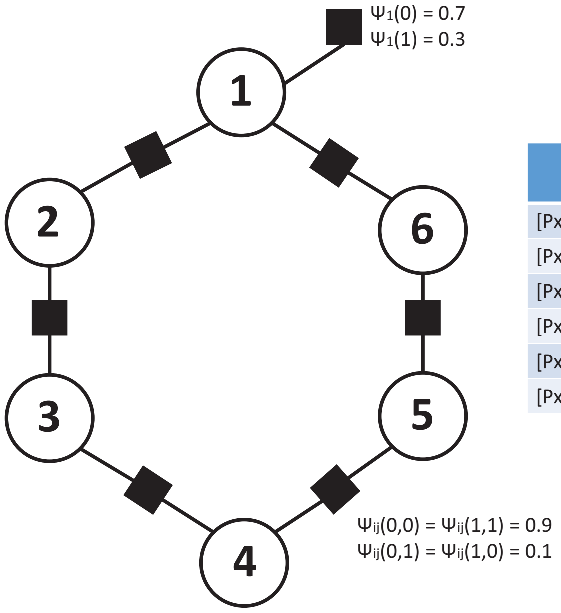
Due to the symmetry of the graph, similar arguments can be made for messages along other edges as well. So we have proved Loopy BP will converge on this graph.

2.4 Intuition: When Should We Expect Loopy BP to Work Well

Since Sum-Product is a distributed algorithm and it's exact on trees, intuitively we expect loopy BP to work well on graphs that are locally tree-like. In other words, if for any node, its neighbours are 'far apart' in the graph, we can think of the incoming messages as roughly independent (recall that incoming messages to each node are independent if and only if the graph is a tree). On the other hand, if a graph contains small loops, the messages around the loop will be amplified and result in marginal estimates very different from the true marginals. The numerical examples below hopefully will provide some intuition.


$\Psi_1(0) = 0.7$
$\Psi_1(1) = 0.3$
1
$\Psi_{12}(0,0) = \Psi_{12}(1,1) = 0.9$
$\Psi_{12}(0,1) = \Psi_{12}(1,0) = 0.1$
$\Psi_{13}(0,0) = \Psi_{13}(1,1) = 0.9$
$\Psi_{13}(0,1) = \Psi_{13}(1,0) = 0.1$
2
3
$\Psi_{23}(0,0) = \Psi_{23}(1,1) = 0.9$
$\Psi_{23}(0,1) = \Psi_{23}(1,0) = 0.1$

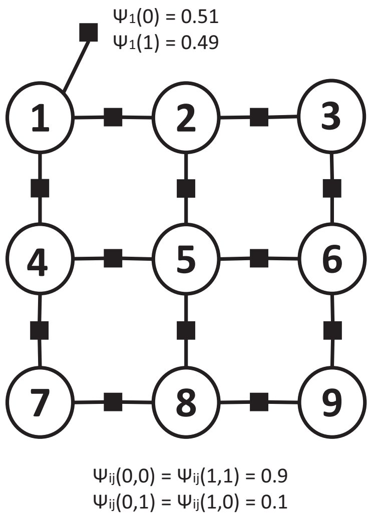
	Exact Marginals	Loopy BP Estimated Marginals
$[P_{x_1}(0), P_{x_1}(1)]$	[0.7, 0.3]	[0.9555, 0.0445]
$[P_{x_2}(0), P_{x_2}(1)]$	[0.6905, 0.3095]	[0.8829, 0.1171]
$[P_{x_3}(0), P_{x_3}(1)]$	[0.6905, 0.3095]	[0.8829, 0.1171]


$\Psi_1(0) = 0.7$
$\Psi_1(1) = 0.3$
1
2
6
3
5
4
$\Psi_{ij}(0,0) = \Psi_{ij}(1,1) = 0.9$
$\Psi_{ij}(0,1) = \Psi_{ij}(1,0) = 0.1$

	Exact Marginals	Loopy BP Estimated Marginals
$[P_{x_1}(0), P_{x_1}(1)]$	[0.7, 0.3]	[0.9027, 0.0973]
$[P_{x_2}(0), P_{x_2}(1)]$	[0.6787, 0.3213]	[0.7672, 0.2328]
$[P_{x_3}(0), P_{x_3}(1)]$	[0.6663, 0.3337]	[0.7487, 0.2513]
$[P_{x_4}(0), P_{x_4}(1)]$	[0.6623, 0.3377]	[0.7427, 0.2573]
$[P_{x_5}(0), P_{x_5}(1)]$	[0.6663, 0.3377]	[0.7487, 0.2513]
$[P_{x_6}(0), P_{x_6}(1)]$	[0.6787, 0.3213]	[0.7672, 0.2328]

Notice in both cases, the marginals estimated from Loopy BP algorithm is higher than the exact marginals. In loops, the incoming messages to a node are not independent. But the Loopy BP algorithm doesn't know that and still treat them as independent. Thus the messages will be counted more than once and result in an 'amplification' effect. Also notice that the 'amplification' is less severe for size-6 loop, because it is more 'locally tree-like'.

What Loopy BP dislikes more than small loops is closely-tied small loops. The following example hopefully explain this point pretty clearly.


$\Psi_1(0) = 0.51$
$\Psi_1(1) = 0.49$
1
2
3
4
5
6
7
8
9
$\Psi_{ij}(0,0) = \Psi_{ij}(1,1) = 0.9$
$\Psi_{ij}(0,1) = \Psi_{ij}(1,0) = 0.1$

	Exact Marginals	Loopy BP Estimated Marginals
$[P_{x_1}(0), P_{x_1}(1)]$	[0.5100, 0.4900]	[0.9817, 0.0183]
$[P_{x_2}(0), P_{x_2}(1)]$	[0.5096, 0.4904]	[0.9931, 0.0069]
$[P_{x_3}(0), P_{x_3}(1)]$	[0.5093, 0.4907]	[0.9811, 0.0189]
$[P_{x_4}(0), P_{x_4}(1)]$	[0.5096, 0.4904]	[0.9931, 0.0069]
$[P_{x_5}(0), P_{x_5}(1)]$	[0.5096, 0.4904]	[0.9992, 0.0008]
$[P_{x_6}(0), P_{x_6}(1)]$	[0.5095, 0.4905]	[0.9930, 0.0070]
$[P_{x_7}(0), P_{x_7}(1)]$	[0.5093, 0.4907]	[0.9811, 0.0189]
$[P_{x_8}(0), P_{x_8}(1)]$	[0.5095, 0.4905]	[0.9930, 0.0070]
$[P_{x_9}(0), P_{x_9}(1)]$	[0.5093, 0.4907]	[0.9810, 0.0190]