CS 228, Winter 2019 Final Exam

This exam is worth 100 points. You have 3 hours to complete it. You may consult notes, books, and use a laptop but no communication or network access is allowed. Good luck!

Stanford University Honor Code

The Honor Code is the University's statement on academic integrity written by students in 1921. It articulates University expectations of students and faculty in establishing and maintaining the highest standards in academic work:

- The Honor Code is an undertaking of the students, individually and collectively:
  - that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  - that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.
- The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking usual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.
- While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

Signature

I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Stanford University Honor Code.

Name / SUNet ID:

Signature:

Question	**Score**	**Question**	**Score**
1	/ 21	5	/ 23
2	/ 26		
3	/ 14		
4	/ 16		

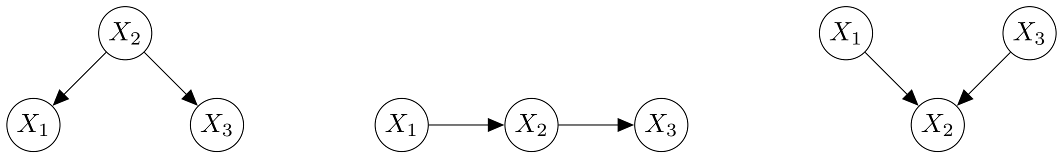
Total score: **/ 100**

Note: Partial credit will be given for partially correct answers. Zero points will be given to answers left blank.

1. **[21 points total] Conceptual Short Answers**

   Each question requires an answer of about one sentence. Longer answers will not positively affect your grade.

   a) **[2 points total]** Consider the three Bayes Nets shown below, answer each of the following with True or False. If True, give a short justification. If False, give a counter-example or one-sentence explanation.

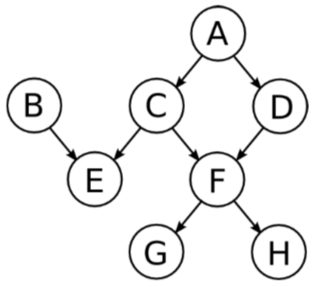


A B C

   i. **[1 points]** True or false: Any distribution that can be represented by Bayesian network A can be represented by Bayesian network B.
      Answer: True. The two models have the same conditional independences.

   ii. **[1 points]** True or false: Any distribution that can be represented by Bayesian network A can be represented by Bayesian Network C.
      Answer: False. Graph C has the additional conditional independence $X_1 \perp X_3$.

   b) **[3 points total]** State and justify whether the following independence properties are True or False for all distributions $P(A, B, C, D, E, F, G, H)$ that factorize according to the graph given below:



   i. **[1 points]** $B \perp G \mid E$
      Answer: False. Information can flow from B to G ($B - E - C - F - G$).

   ii. **[1 points]** $C \perp D \mid G, A$
      Answer: False. Since we condition on G, and G is a descendant of F, F is not blocking.

   iii. **[1 points]** $B \perp D \mid C$
      Answer: True. E and its descendants are not observed, so it is blocking.

   c) **[2 points total]** (Circle the correct answer/s). Let G and G' be two minimal I-maps for a distribution p. Then

   i. G and G' must be equal
   ii. G and G' must have the same number of edges
   iii. G and G' must be I-equivalent
   iv. None of the above

Answer: iv. none of the above

d) **[2 points total]** In graph theory, a pseudoforest is an undirected graph in which every connected component has at most one cycle. What is the maximum treewidth of a pseudoforest? Explain your answer.

Note: A connected component is a sub-graph where any pair of vertices are connected.

Answer: 2, because we can first eliminate all the non-cycle branches, and then the remaining cycle graph has treewidth 2.

e) **[2 points total]** Figure 1 shows a simple Markov chain and its transition probabilities.

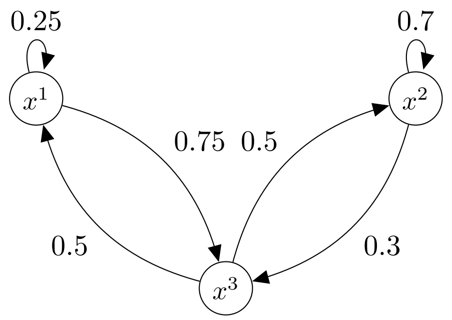


Figure 1: Markov Chain $\mathcal{T}$. There are three states (x^1, x^2, x^3) with corresponding transition probabilities. For example, the probability of entering state x^3 from x^2 is 0.3.

Which of the following is a stationary distribution of the Markov chain? Explain your choice.

a. $\pi_1(x^1) = 0.14,\quad \pi_1(x^2) = 0.35,\quad \pi_1(x^3) = 0.21$
b. $\pi_2(x^1) = 0.1,\quad \pi_2(x^2) = 0.4,\quad \pi_2(x^3) = 0.5$
c. None of the above

Answer: By definition the stationary distribution π must satisfy the following three equations:

$$\pi(x^1) = 0.25\pi(x^1) + 0.5\pi(x^3)$$
$$\pi(x^2) = 0.7\pi(x^2) + 0.5\pi(x^3)$$
$$\pi(x^3) = 0.75\pi(x^1) + 0.3\pi(x^2)$$

as well as the one asserting that it is a legal distribution:

$$\pi(x^1) + \pi(x^2) + \pi(x^3) = 1$$

It is straightforward to verify that the system has a unique solution:

$$\pi(x^1) = 0.2, \pi(x^2) = 0.5, \pi(x^3) = 0.3.$$

f) **[3 points total]** Consider a probability distribution p over n binary variables $\mathbf{X} = (X_1, \ldots, X_n)$ such that

$$p(\mathbf{X} = \mathbf{x}) \propto \begin{cases} e^M & \mathbf{x} = \mathbf{0} \\ 1 & \text{otherwise} \end{cases}$$

where $M \in \mathbb{R}$ is a parameter. You decide to run a Gibbs sampler to sample from p, using random (uniform) initialization. Assume n is a large number. How long should the burn-in period be when $M = 0$? Would you need a very long burn-in period if $M = n$? Would you need a very long burn-in period if $M = -n$? Explain your answers.

Answers: When $M = 0$ it is already in equilibrium. When $M = n$ a long burn in is needed because it is highly unlikely that $p(X = 0)$ is reached before exponential time. So $\rho(p(x^t), p^*(x))$ will be large by any metric (total variation or KL).

For $M = -n$ we are almost converging to begin with. So don't need a long burn in period. (Unless they mention convergence in ∞ norm, then we give credit for answering 'need a long burn in').

g) **[4 points total]** We have a MRF over a set of variables $\mathbf{x} = (x_1, \dots, x_k)$, whose probability is defined by

$$p_\theta(\mathbf{x}) = \frac{1}{Z_p} e^{\theta_1 f_1(\mathbf{x}) + \theta_2 f_2(\mathbf{x}) + \cdots + \theta_k f_k(\mathbf{x})}$$

where $f_1, \dots, f_k$ are known functions, and $\theta = (\theta_1, \dots, \theta_k)$ are the real valued parameters. We would like to approximate this distribution by

$$q_\phi(\mathbf{x}) = \frac{1}{Z_q} e^{\phi_1 g_1(x_1) + \cdots + \phi_k g_k(x_k)}$$

where $g_1, \dots, g_k$ are known functions, and $\phi = (\phi_1, \dots, \phi_k)$ are parameters we would like to optimize. Note that compared to p_θ, the difference is that each g_i now only depends on one variable x_i rather than all of $\mathbf{x}$.

i. **[1 points]** Can we compute Z_q in polynomial time (with respect to k)? Justify your answer.
Answer: Yes, we can compute Z_q in polynomial time. This is because Z_q is equal to the product of the partition function for univariate distribution $p(x_i) \propto e^{\phi_i g_i(x_i)}$

ii. **[3 points]** Suppose we have two candidate parameters ϕ, ϕ', and we would like to know which one is better by computing $\mathrm{KL}(q_\phi(\mathbf{x}) \| p_\theta(\mathbf{x})) - \mathrm{KL}(q_{\phi'}(\mathbf{x}) \| p_\theta(\mathbf{x}))$. Given samples $\mathbf{x}[1], \dots, \mathbf{x}[m] \sim q_\phi(\mathbf{x})$ and $\mathbf{x}'[1], \dots, \mathbf{x}'[m] \sim q_{\phi'}(\mathbf{x})$, derive a Monte Carlo estimator for the above expression. All terms in your estimator should be computable in time polynomial in n, k.
Answer: Note: it's also fine to replace all integrals below with sums.

$$\begin{aligned}\mathrm{KL}(q_\phi(\mathbf{x}) \| p_\theta(\mathbf{x})) &= \int q_\phi(\mathbf{x}) \left(\sum_i (\phi_i g_i(\mathbf{x}) - \theta_i f_i(\mathbf{x})) + \log Z_p - \log Z_{q_\phi} \right) d\mathbf{x} \\ &= \int q_\phi(\mathbf{x}) \left(\sum_i (\phi_i g_i(\mathbf{x}) - \theta_i f_i(\mathbf{x})) \right) d\mathbf{x} + \log Z_p - \log Z_{q_\phi}\end{aligned}$$

After subtracting the two KL divergences, the $\log Z_p$ will cancel out, we can compute $\log Z_q$, and we can estimate

$$\int q_\phi(\mathbf{x}) \sum_i (\phi_i g_i(\mathbf{x}) - \theta_i f_i(\mathbf{x})) \approx \frac{1}{m} \sum_j q_\phi(\mathbf{x}[j]) \sum_i (\phi_i g_i(\mathbf{x}[j]) - \theta_i f_i(\mathbf{x}[j]))$$

h) **[3 points total]** Suppose we have a distribution $p_\theta(x)$ with unknown parameter(s) θ, where $x \in \mathbb{R}$. We also have a dataset $x_1, \dots, x_n \in \mathbb{R}$, and to store it more efficiently we only store the following quantities

$$\sum_{i=1}^n x_i, \qquad \sum_{i=1}^n x_i^2 \tag{1}$$

For each of the following distribution classes (i)-(iii), we would like to find the parameter $\theta = (\theta_1, \theta_2)$ that maximizes the likelihood of observing data $x_1, \dots, x_n$ under $p_\theta(x)$.

i. $p_\theta(x) \propto e^{\theta_1 x}$
ii. $p_\theta(x) \propto e^{\theta_1 x + \theta_2 x^{-1}}$
iii. $p_\theta(x) \propto e^{\theta_1 (x - \theta_2)^2}$

Can we get the same maximum likelihood estimation given only the quantities in Eq.(1) rather than the entire dataset? You should answer the question for (i), (ii), (iii) separately. A short Yes/No answer is expected.

Answers:

i. Yes
ii. No
iii. Yes

2. **[26 points total] Application of Graphical Models to Stereo Correspondence**

In the dense stereo correspondence problem, we are given two grayscale images of the same scene, taken from different points of view. Let $\mathcal{I}^{(L)}$, $\mathcal{I}^{(R)} \in \mathbb{R}^{M \times M}$ denote the images taken from the left and right views, respectively. Our goal is to estimate a *disparity map* that indicates how much every object in the image shifts horizontally between the two images. For each pixel in the left image, the corresponding pixel in the right image shifts to the left. The closer an object is to the camera, the more it shifts. You can convince yourself of these facts by looking at Figure 2, which gives an example of a pair of stereo images and a disparity map[1]. Disparity maps are useful for distance estimation and 3-D scene reconstruction tasks.

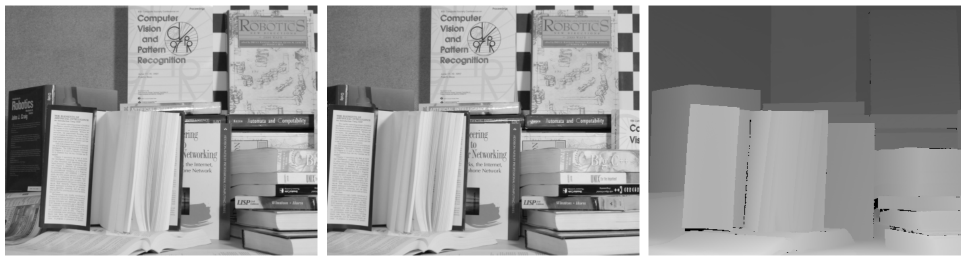


Figure 2: (Left, Middle) Pair of rectified stereo images $\mathcal{I}^{(L)}$, $\mathcal{I}^{(R)}$. (Right) True disparity map.

For each pixel i in the left image, we use d_i to denote how many pixels it shifts. d_i is a discrete random variable taking on values $\{0, \ldots, N-1\}$, where we assume that each pixel shift to the left by at most $N-1$ pixels horizontally. We also assume that the images are perfectly aligned in the vertical direction, so objects do not shift vertically between the two images.

If we shift pixel i in the left image d_i pixels, we will get a pixel in the right image, which we denote $i - d_i$. Our hope is that this pixel $i - d_i$ will have similar intensity to our original pixel i. Formally, we define the local cost c_i of assigning disparity d_i to pixel i as the squared difference in intensity between pixel i in the left image and pixel $i - d_i$ in the right image:

$$c_i(d_i) = \left(\mathcal{I}_i^{(L)} - \mathcal{I}_{(i-d_i)}^{(R)}\right)^2.$$

Assume that you are given precomputed vectors $\mathbf{c}_i = [c_i(0), \ldots, c_i(N-1)]$ for every pixel i.

Because disparity maps tend to be smooth, we also define a smoothness cost parameterized by θ. Let $\mathcal{E}$ be the set of all neighboring pixel pairs. For $(i,j) \in \mathcal{E}$, the smoothness cost is

$$s_{ij}(d_i, d_j; \theta) = \begin{cases} 0 & d_i = d_j \\ \theta & d_i \neq d_j \end{cases}.$$

Thus, a disparity assignment $\mathbf{d} = (d_1, \cdots, d_{M \times M})$ to all pixels in the left image has total cost (or *energy*)

$$F(\mathbf{d}; \mathbf{c}, \theta) = \sum_i c_i(d_i) + \sum_{(i,j) \in \mathcal{E}} s_{ij}(d_i, d_j; \theta).$$

In this question, we will explore how to compute a disparity map $\mathbf{d}$ and how to learn the best parameter θ.

a) **[2 points]** Recall (from HW 4) that the relationship between the energy function $F(x)$ and the probability distribution $P(x)$ is

$$F(x) = -\log P(x) + C$$

[1]Images are reproduced from the Middlebury Stereo Vision Dataset at `http://vision.middlebury.edu/stereo/`. D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN, June 2007.

for some constant C. Note that lower energy states have higher probability. Write out the equation for the probability distribution $P(\mathbf{d};\mathbf{c},\theta)$. Include the equation for the partition function.

Answer:

$$P(\mathbf{d};\mathbf{c},\theta) = \frac{1}{Z(\mathbf{c},\theta)} \exp\left(-F(\mathbf{d};\mathbf{c},\theta)\right)$$
$$Z(\mathbf{c},\theta) = \sum_{\mathbf{d}\in Val(\mathbf{d})} \exp\left(-F(\mathbf{d};\mathbf{c},\theta)\right)$$

b) [**2 points**] We can graphically represent $P(\mathbf{d};\mathbf{c},\theta)$ with a factor graph as shown in Figure 3. What are the factors ψ_i and σ_{ij} in terms of c_i and s_{ij}, respectively?

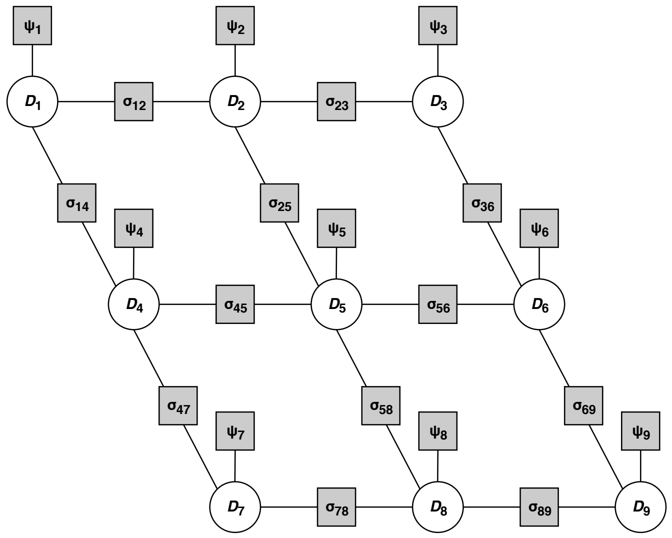


Figure 3: Factor graph representing the probability distribution over a disparity map for a 2-D grayscale image.

Answer:

$$\psi_i(d_i) = e^{-c_i(d_i)} \qquad \sigma_{ij}(d_i,d_j) = e^{-s_{ij}(d_i,d_j;\theta)}$$

c) **[6 points]** Suppose we are given the ground truth disparity map $\mathbf{d}_0$, and we would like to find the best θ by minimizing negative log-likelihood $L(\theta; \mathbf{d}_0, \mathbf{c}) = -\log P(\mathbf{d}_0; \mathbf{c}, \theta)$. Show that its derivative can be written as

$$\frac{d}{d\theta} L(\theta; \mathbf{d}_0, \mathbf{c}) = \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) - \mathbb{E}_{\mathbf{d} \sim P(\mathbf{d}; \mathbf{c}, \theta)} \left[\frac{d}{d\theta} F(\mathbf{d}; \mathbf{c}, \theta) \right]$$

Answer:

$$\begin{aligned} L(\theta; \mathbf{d}_0, \mathbf{c}) &= -\log P(\mathbf{d}_0; \mathbf{c}, \theta) \\ &= F(\mathbf{d}_0; \mathbf{c}, \theta) + \log Z(\mathbf{c}, \theta) \end{aligned}$$

$$\begin{aligned} \frac{d}{d\theta} L(\theta; \mathbf{d}_0, \mathbf{c}) &= \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) + \frac{1}{Z(\mathbf{c}, \theta)} \frac{dL}{d\theta} Z(\mathbf{c}, \theta) \\ &= \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) + \frac{1}{Z(\mathbf{c}, \theta)} \sum_{\mathbf{d} \in Val(\mathbf{d})} \frac{d}{d\theta} \exp[-F(\mathbf{d}; \mathbf{c}, \theta)] \\ &= \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) - \frac{1}{Z(\mathbf{c}, \theta)} \sum_{\mathbf{d} \in Val(\mathbf{d})} \exp[-F(\mathbf{d}; \mathbf{c}, \theta)] \frac{d}{d\theta} F(\mathbf{d}; \mathbf{c}, \theta) \\ &= \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) - \sum_{\mathbf{d} \in Val(\mathbf{d})} P(\mathbf{d}; \mathbf{c}, \theta) \frac{d}{d\theta} F(\mathbf{d}; \mathbf{c}, \theta) \\ &= \frac{d}{d\theta} F(\mathbf{d}_0; \mathbf{c}, \theta) - \mathbb{E}_{\mathbf{d} \sim P(\mathbf{d}; \mathbf{c}, \theta)} \left[\frac{d}{d\theta} F(\mathbf{d}; \mathbf{c}, \theta) \right] \end{aligned}$$

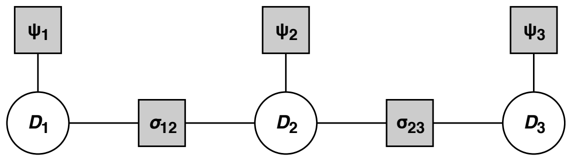


Figure 4: Factor graph (Hidden Markov Model) representing distribution over a disparity map for a single 1-D scanline (row of an image).

d) **[8 points]** Computing the term $\mathbb{E}_{\mathbf{d}\sim P(\mathbf{d};\mathbf{c},\theta)}\left[\frac{d}{d\theta}F(\mathbf{d};\mathbf{c},\theta)\right]$ can be difficult in the full factor graph in Figure 3. However, if we only create disparity maps for one "scanline" (row of an image) at a time as shown in Figure 4, we can compute this quantity in polynomial time. Show that you can compute this expectation in $O(MN^2)$ time where M is the number of pixels per row and each pixel can take on N possible disparity values.

If you use message passing, you should descibe which messages you compute (You don't have to write out the exact formulas for computing them), and write down the expression for the desired quantities as a function of the messages. If your algorithm is $\Theta(M^2N^2)$ complexity you can get 5 points.

Hint: HW3 Q2 is a related problem you can refer to. In addition, you can use the following results:

$$\frac{\partial}{\partial\theta}s_{ij}(d_i,d_j;\theta) = \mathbf{1}(d_i \neq d_j)$$

where $\mathbf{1}(d_i \neq d_j)$ is 1 if $d_i \neq d_j$ and 0 otherwise. For any random vector $\mathbf{X} = [X_1,\dots,X_n]^T$,

$$\mathbb{E}_{\mathbf{x}\sim P(\mathbf{X})}[f(x_i,x_j)] = \mathbb{E}_{(x_i,x_j)\sim P(X_i,X_j)}[f(x_i,x_j)]$$

where $P(X_i,X_j)$ is the marginal distribution over (X_i,X_j).

Answer:

$$\begin{aligned}
\mathbb{E}_{\mathbf{d}\sim P(\mathbf{d};\mathbf{c},\theta)}\left[\frac{d}{d\theta}F(\mathbf{d};\mathbf{c},\theta)\right] &= \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{\mathbf{d}\sim P(\mathbf{d};\mathbf{c},\theta)}\left[\frac{d}{d\theta}s_{ij}(d_i,d_j;\theta)\right] \\
&= \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{(d_i,d_j)\sim P(d_i,d_j;\mathbf{c},\theta)}\left[\frac{d}{d\theta}s_{ij}(d_i,d_j;\theta)\right] \\
&= \sum_{(i,j)\in\mathcal{E}} \sum_{d_i}\sum_{d_j} P(d_i,d_j;\mathbf{c},\theta)\mathbf{1}[d_i \neq d_j]
\end{aligned}$$

Since this is a chain-structured factor graph, we can perform variable elimination from both ends of the chain to compute the entire distribution $P(d_i,d_j;\mathbf{c},\theta)$ in $O(MN^2)$ time. There are $M-1$ pairs $(i,j)\in\mathcal{E}$, so we can naively compute all such distributions in $O(M^2N^2)$ time. However, by caching the intermediate factors created during variable elimination (as explained in HW3 Q2c), we can compute all pairwise distributions in $O(MN^2)$ time.

The first sum is over $M-1$ pairs, and the two other summations are over N^2 values for (d_i,d_j), so the summations also take $O(MN^2)$ time. Thus, in total, the expectation can be calculated exactly in $O(MN^2)$ time.

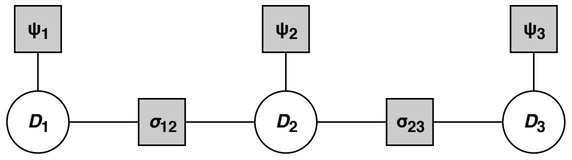


Figure 4: Same figure as for part (d), reproduced here for convenience.

e) [**4 points total**] Instead of computing the expectation $\mathbb{E}_{\mathbf{d}\sim P(\mathbf{d};\mathbf{c},\theta)}\left[\frac{d}{d\theta}F(\mathbf{d};\mathbf{c},\theta)\right]$ exactly, you consider taking samples from the distribution $P(\mathbf{d};\mathbf{c},\theta)$ and computing a Monte-Carlo estimate of the expectation.

Your algorithm consists of three steps:

Step 1. Run belief propagation over the factor graph (Figure 4).

Step 2. Starting at one end of the chain, you sample $d_1 \sim P(D_1;\mathbf{c},\theta)$

Step 3. Sample $d_2 \sim P(D_2 \mid D_1 = d_1;\mathbf{c},\theta)$, and so on, following the chain rule.

Is this sampling method more efficient or less efficient than computing the expectation exactly, as in part (d)? Explain.

Answer: Sampling a joint assignment takes $O(MN^2)$ time (e.g. in HW4). Sampling just a single complete assignment to **d** requires the same time complexity as computing the expectation exactly. Therefore, the sampling method is less efficient than the exact computation, especially since we may need many samples to reduce the variance of our estimator.

f) [**2 points**] We return to the setting of the 2-D factor graph (Figure 3). Because it is not tree-structured, we use Loopy BP to sample from $P(\mathbf{d};\mathbf{c},\theta)$. Are the samples $\mathbf{d}^{(1)},\mathbf{d}^{(2)},\ldots$ guaranteed to yield an unbiased Monte-Carlo estimate for

$$\mathbb{E}_{\mathbf{d}\sim P(\mathbf{d};\mathbf{c},\theta)}\left[\frac{d}{d\theta}F(\mathbf{d};\mathbf{c},\theta)\right]$$

Explain in 3 or fewer sentences.

Answer: Loopy BP is not guaranteed to converge to the true distribution $P(\mathbf{d};\mathbf{c},\theta)$, so the samples we generate are not guaranteed to be from this desired distribution. Therefore, the Monte-Carlo estimate is likely to be biased. However, in practice, sampling using Loopy BP performs reasonably well.

g) [**2 points**] Assume we have learned the optimal parameter θ^{MLE}. Given a new pair of images and corresponding **c**, we would like to infer the depth map. Suppose by the procedure in (f) we draw a few samples $\mathbf{d}^{(1)},\mathbf{d}^{(2)},\ldots$ from $P(\mathbf{d};\mathbf{c},\theta^{MLE})$. Propose a simple strategy to pick the best sample as the outcome of our algorithm.

Answer: Use the one such that $F(\mathbf{d}^{(i)};\mathbf{c},\theta)$ is smallest. (Or equivalently, with largest P).

3. **[14 points total] Bayesian learning and Structure learning**

   Your friendly TAs give you a dataset over three binary variables (A, B, C) and ask you to use structure learning and Bayesian learning to find the best graphical model G to represent the distribution over these variables. The dataset $\mathcal{D}$ is given in Table 1. You first hypothesize a Bayes network G shown in Figure 6.

   (a) **[1 points]** Based on graph G, what is the maximum likelihood estimate for $P(A = 0)$?
   Answer: 5/8

   (b) **[1 points]** Let $P(B = 1) = \theta_B$, where θ_B is a parameter we would like to estimate. Assume your friend uses a prior distribution of $Beta(3, 3)$ for θ_B. You are more confident than your friend that variable B is equally likely to be 0 or 1. Which prior distribution would you choose, $Beta(1, 1)$ or $Beta(5, 5)$? Justify your choice in one or two sentences.

   Answer: $Beta(5, 5)$. If you are more confident, you should incorporate more pseudo-counts into the prior.

   (c) **[2 points]** You change your mind and decide that $Beta(2, 3)$ is a more reasonable prior for θ_B. Based on graph G and the dataset, compute the Bayesian posterior probability $P(B = 1|\mathcal{D})$?

   Note: The density of $Beta(\alpha, \beta)$ is $p(x) \propto x^{\alpha-1}(1 - x)^{\beta-1}$. α is the pseudo-count for when the variable equals 1, and β is the pseudo-count for when the variable equals 0.

   Answer: $\frac{4+2}{8+2+3} = 6/13$

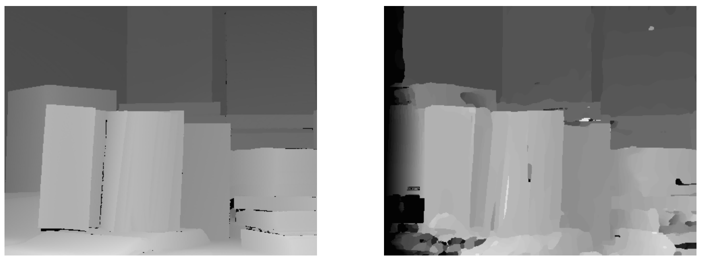


Figure 5: For fun, we show the output of a similar model with slightly more complex local and smoothness cost functions. The local cost function compares a patch of pixel intensities, and the pairwise cost function assigns different penalties depending on pixel intensities.
(Left) Ground truth disparity map. (Right) Estimated disparity map.

Sample	A	B	C
1	1	0	1
2	1	0	0
3	0	1	0
4	0	1	1
5	0	0	0
6	1	1	1
7	0	1	0
8	0	0	1

Table 1: A dataset over three binary variables.

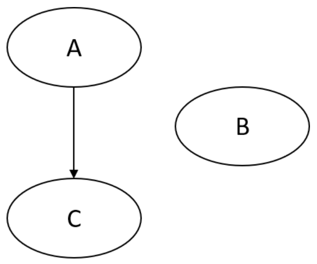


Figure 6: Graph G

(d) **[3 points]** Now you want to use local search to search for the optimal graphical model over the full dataset D. You **reverse the connection between A and C** in graph G and obtain G', but it seems that log-likelihood of the dataset $\ell_G(\theta;\mathcal{D})$ stays the same with maximum likelihood estimates, i.e. $\ell_G(\theta_G^{ML};\mathcal{D}) = \ell_{G'}(\theta_{G'}^{ML};\mathcal{D})$. Why is this?
Answer:

$$\begin{aligned}
\ell_G(\theta_G^{ML};\mathcal{D}) - \ell_{G'}(\theta_{G'}^{ML};\mathcal{D}) &= \left(\sum_A M[A]\log\frac{M[A]}{M} + \sum_A\sum_C M[A,C]\log\frac{M[A,C]}{M[A]}\right)\\
&- \left(\sum_C M[C]\log\frac{M[C]}{M} + \sum_C\sum_A M[A,C]\log\frac{M[A,C]}{M[C]}\right)\\
&= \left(\sum_A\sum_C M[A,C]\log\frac{M[A]}{M} + \sum_A\sum_C M[A,C]\log\frac{M[A,C]}{M[A]}\right)\\
&- \left(\sum_C\sum_A M[A,C]\log\frac{M[C]}{M} + \sum_C\sum_A M[A,C]\log\frac{M[A,C]}{M[C]}\right)\\
&= 0
\end{aligned}$$

Alternative answer: The two graphs are I-equivalent. They have the same log likelihood.

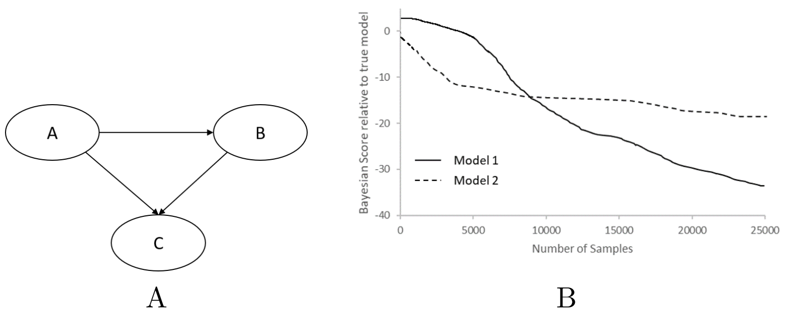


Figure 7: **(A)** Graph G_{fc}. **(B)** Relative Bayesian score of two graphical models over different number of training samples.

(e) Now suppose your TAs instead gave you a much larger dataset $\mathcal{D}$ of 25000 training examples. You have more data, so in addition to the original graph G, you also consider a graph with more edges G_{fc}, as shown in Figure 7A.

To decide which graph is a better model for the data, you use a Bayesian structure learning approach. You start with a prior over possible graph structures, $P(G) \propto e^{-5|G|}$,

where $|G|$ is the number of edges in G. Given a graph G, you have a prior over the parameters of G given by $P(\theta|G)$, where every free parameter of the Bayes Network G has prior $Beta(5,5)$.
To learn the structure, you use the **Bayesian Score**, which is defined as

$$Score(G;\mathcal{D}) = \log P(G \mid \mathcal{D})$$

where $P(G \mid \mathcal{D})$ is the posterior probability over graph structures given the dataset $\mathcal{D}$

$$P(G \mid \mathcal{D}) \propto P(G) \int P(\mathcal{D} \mid \theta, G)\, P(\theta \mid G)\, d\theta. \tag{2}$$

Your instructor has the true model G^* (but not the true parameter, so as before, you assume a prior $Beta(5,5)$ for every free parameter). The **Relative Bayesian Score** of a graph G is defined as $Score(G;\mathcal{D}) - Score(G^*;\mathcal{D})$.

i. [**2 points**] Figure 7 plots the relative Bayesian score for G and G_{fc} as a function of the amount of training data available. For Model 1 and Model 2 in Figure 7B, indicate which corresponds to G and which corresponds to G_{fc}. What is your reason?
Answer: Model 1 corresponds to G and Model 2 corresponds to G_{fc}. The reason is since G is simpler, the larger value for $\log P(G)$ initially dominate. However, because it is misspecified, with more data, the $\log \int p(\mathcal{D}|\theta)P(\theta|G)$ term dominates and it becomes worse.

ii. [**5 points**] We investigate behavior of relative Bayesian score as we increase the size of our training data. For Model 1 in Figure 7B, as the number of training examples in $\mathcal{D}$ goes to $+\infty$, will its relative Bayesian score go to $-\infty$ or approach some finite value? To get credit you must justify your answer (formal proof not needed).
You may assume that G^* is a perfect map for the true distribution (G^*, θ^*), and G^* is neither G nor G_{fc}. (In fact, this should be obvious from Figure 7.)
Hint: think about what it requires for the relative log likelihood to either reach a constant or go to $-\infty$.
Answers: Note: Any answer that arrive at the correct conclusion, and give sensible justification (e.g. a model that cannot represent a true distribution has small data likelihood, so posterior is small) is fine.
It will go to $-\infty$. The key idea is that with infinite data, the 'insufficient' model will be invalidated and $P(G|D)$ will go to zero.

$$\log P(G_{\text{model 1}}|\mathcal{D}) - \log P(G^*|\mathcal{D}) = C + \log \frac{\int p(\mathcal{D}|\theta, G)p(\theta|G)d\theta}{\int p(D|\theta, G^*)p(\theta|G^*)d\theta}$$

where C is a contant irrelavent to the size of the dataset. So we only have to verify that

$$\frac{\int p(\mathcal{D}|\theta, G)p(\theta|G)d\theta}{\int p(D|\theta, G^*)p(\theta|G^*)d\theta} \to 0$$

this can be verified by $\frac{p(\theta^*|G)}{p(\theta^*|G^*)} = 0$ where θ^* is the true parameter. Intuitively, this means that G cannot represent the true distribution, θ^* while G^* can represent the true distribution θ^*.
Semi-rigorous proof: Not part of the test, but placed here as a sanity check that the intuitive answer is correct. This proof can be made fully rigorous by making it measure theoretic and use PAC bounds instead of strong law of large numnbers.
Let G^*, θ^* be true model and parameter, denote its distribution as p^*. Note for different graphs G, θ belong to different spaces. To avoid this technical difficulty, we

assume that all θ are defined relative to the fully connected graph. Any other graph define prior distributions $p(\theta|G)$ on some lower dimensional subspace. Denote $\Theta(G)$ as the parameters a graph can represent. We let G_0 denote the original graph (i.e. the one in Figure 6).
Let d be a joint assignment of all variables, then by the strong law of large numbers, under distribiton p^*, for any graph and parameter pair (G, θ), almost surely

$$\lim_{n\to\infty}\sum_{i=1}^{n}\log p(d_i|\theta, G) \to n\mathbb{E}_{p^*(d)}[\log p(d|\theta, G)]$$

Then

$$\frac{\int p(\mathcal{D}|\theta, G)p(\theta|G)}{\int p(\mathcal{D}|\theta, G^*)p(\theta|G^*)} \to \frac{\int_\theta e^{n\mathbb{E}_{p^*(d)}[\log p(d|\theta,G)]}p(\theta|G)}{\int_\theta e^{n\mathbb{E}_{p^*(d)}[\log p(d|\theta,G^*)]}p(\theta|G^*)}$$

For any graph G, denote suboptimality score as

$$s(\theta, G) = \mathbb{E}_{p^*(d)}[\log p^*(d)] - \mathbb{E}_{p^*(d)}[\log p(d|\theta, G)]$$

The set of ϵ-suboptimal parameter as

$$N(G, \epsilon) = \{\theta \in \Theta(G) : s(\theta, G) \leq \epsilon\}$$

Then it is easy to see that there exists a ϵ_0 such that $N(G_0, \epsilon_0) = \emptyset$ because G_0 is not an I-Map for the true distribution, while for any $\epsilon > 0$, $N(G^*, \epsilon) \neq \emptyset$. Let $\neg N(G_0, \epsilon)$ denote all parameters θ representable by G not in $N(G_0, \epsilon)$ Therefore for any $\epsilon < \epsilon_0$

$$\begin{aligned}\frac{\int_\theta e^{n\mathbb{E}_{p^*(d)}[\log p(d|\theta,G)]}p(\theta|G)}{\int_\theta e^{n\mathbb{E}_{p^*(d)}[\log p(d|\theta,G^*)]}p(\theta|G^*)} &= \frac{\int_\theta e^{-ns(\theta,G)p(\theta|G)}}{\int_\theta e^{-ns(\theta,G^*)p(\theta|G^*)}} \\ &\leq \frac{\int_{\neg N(G_0,\epsilon)} e^{-n\epsilon_0}p(\theta|G)}{\int_{N(G^*,\epsilon)} e^{-n\epsilon}} \leq \frac{e^{n(\epsilon-\epsilon_0)}}{p(N(G^*,\epsilon)|G^*)} \to^{n\to\infty} 0\end{aligned}$$

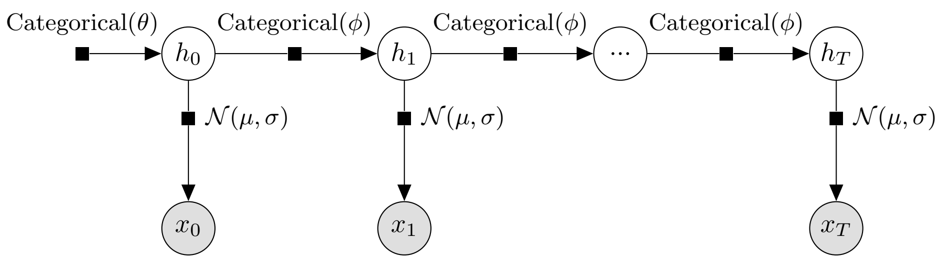


Figure 8: The graphical representation of the Hidden Markov Model

4. **[16 points total] Expectation Maximization**

In this problem we will derive EM update for the Hidden Markov Model (HMM) shown in Figure 8. This model is parametrized by $\theta \in \mathbb{R}^K$, $\phi \in \mathbb{R}^{K\times K}$, $\mu \in \mathbb{R}^K$ and $\sigma \in \mathbb{R}^K$ with

$$\begin{aligned} h_0 &\sim \text{Categorical}(\theta) && \text{(i.e. } P(h_0 = u; \theta) = \theta_u, \forall u \in \{1, \ldots, K\}) \\ (h_i \mid h_{i-1} = u) &\sim \text{Categorical}(\phi_u) && \text{(i.e. } P(h_i = v | h_{i-1} = u; \phi) = \phi_{uv}, \forall u, v \in \{1, \ldots, K\}) \\ (x_i \mid h_i = u) &\sim \mathcal{N}(\mu_u, \sigma_u) && (\mathcal{N} \text{ is the Gaussian distribution}) \end{aligned}$$

In what follows, take $\mathbf{x} = (x^{(1)}, \ldots, x^{(N)})$ to be our training data set with N sequences of length $T+1$, where each training data $x^{(n)} \in \mathbb{R}^{T+1}, n = 1, \ldots, N$. Thus for the n-th sample $x^{(n)}$ and a full assignment to the hidden variables $h^{(n)} = (h_0^{(n)}, \ldots, h_T^{(n)})$ we have:

$$P(x^{(n)}, h^{(n)}) = P(h_0^{(n)}; \theta) \left(\prod_{i=1}^{T} P(h_i^{(n)} \mid h_{(i-1)}^{(n)}; \phi) \right) \left(\prod_{i=0}^{T} \mathcal{N}(x_i^{(n)}; \mu_{h_i^{(n)}}, \sigma_{h_i^{(n)}}) \right)$$

where $\mathcal{N}(x_i; \mu_u, \sigma_u)$ is the Gaussian density function

$$\mathcal{N}(x_i; \mu_u, \sigma_u) = \frac{1}{\sqrt{2\pi\sigma_u^2}} e^{(x_i - \mu_u)^2 / 2\sigma_u^2}$$

(a) **[7 points]** Write the MLE of each parameter $(\theta, \phi, \mu, \sigma)$ if all variables are observed, i.e., we observe $(x^{(n)}, h^{(n)})$ for $n = 1, \ldots, N$.

Answer:

$$\theta_k : \frac{1}{N} \sum_n 1[h_0^{(n)} = k]$$

$$\phi_{k,k'} : \frac{\sum_n \sum_j 1[h_j^{(n)} = k, h_{(j+1)}^{(n)} = k']}{\sum_n \sum_j 1[h_j^{(n)} = k]}$$

$$\mu_k : \frac{\sum_n \sum_j X_j^{(n)} 1[h_j^{(n)} = k]}{\sum_n \sum_j 1[h_j^{(n)} = k]}$$

$$\sigma_k : \frac{\sum_n \sum_j (X_j^{(n)} - \mu_k)^2 1[h_j^{(n)} = k]}{\sum_n \sum_j 1[h_j^{(n)} = k]}$$

(b) **[2 points]** Now assume that h is not observed. Given some input data sample $x^{(n)}$, what efficient algorithm can you use to compute $P(h_i^{(n)} = u \mid x^{(n)}), \forall 0 \leq i \leq T$ and $P(h_i^{(n)} = u, h_{i+1}^{(n)} = v \mid x^{(n)}), \forall 0 \leq i < T$, and $u, v \in \{1, \dots, K\}$?

Answer: Mesasage passing

(c) **[7 points]** Given parts (a) and (b), derive the M-step for updating the parameters when h is not observed. Please simplify your notation by denoting

$$w_i^{(n)}(u) := P(h_i^{(n)} = u \mid x^{(n)})$$
$$w_i^{(n)}(u,v) := P(h_i^{(n)} = u, h_{i+1}^{(n)} = v \mid x^{(n)})$$

Answer:

$$\theta_u^{t+1} := \frac{1}{N} \sum_i w_0^{(n)}(u)$$

$$\phi_{u,v}^{t+1} := \frac{\sum_n \sum_j w_j^{(n)}(u,v)}{\sum_n \sum_j w_j^{(n)}(u)}$$

$$\mu_u^{t+1} := \frac{\sum_n \sum_j w_j^{(n)}(u) X_j^{(n)}}{\sum_n \sum_j w_j^{(n)}(u)}$$

$$\sigma_u^{t+1} := \frac{\sum_n \sum_j w_j^{(n)}(u)(X_j^{(n)} - \mu_u^t)^2}{\sum_n \sum_j w_j^{(n)}(u)}$$

5. **[23 points total]** In a Sudoku puzzle, the goal is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (see Figure 9) contain all of the digits from 1 to 9.

   Let $\mathbf{x} = (x_1, \ldots, x_{81})$ be a vector that represents the 81 numbers in the grid (the order of $(x_1, \ldots, x_{81})$ is irrelevent to this question). We call $\mathbf{x}$ a valid solution if it satisfies the above constraints.

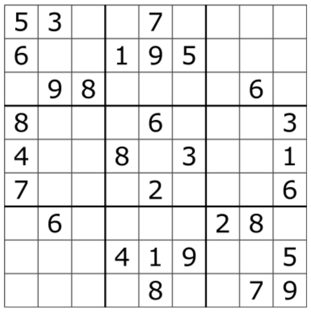


Figure 9: A Sudoku puzzle.

   (a) **[2 points total]** You want to model the Sudoku game using a graphical model. Compactly specify a probability distribution $p(\mathbf{x})$ using an undirected graphical model such that

$$p(\mathbf{x}) \propto \begin{cases} 1 & \mathbf{x} \text{ is a solution} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

   Use a few sentences to describe which variables are in each clique or factor. Your graphical model should not have cliques / factors with more than 9 variables.

   Answer: A possible answer is to use undirected graphical model, where for each constraint over 9 variables we place a potential $\phi(x)$ over the nine variables, such that $\phi(x) = 1$ if and only if they contain values 0-9.

   (b) **[1 points total]** What is the meaning of the normalization constant for Equation 3? In other words, what fact about the game does the normalization constant tell us?

   Answer: The number of solutions to the puzzle.

   (c) **[1 points total]** Let X^{UL} denote all the variables in the upper-left 3×3 subgrid. Under your graphical model, what is the Markov Blanket of X^{UL}?

   Answer: The set of all variables on the left three columns and top three columns (excluding X^{UL} itself).

   (d) **[2 points total]** When playing Sudoku, you are given a partially filled square (like Figure 9). You are asked to complete it to get a valid solution. Sudoku puzzles are typically designed so that there is a unique valid solution. Use your model from (a) and one of the algorithms you have seen in class to check if exactly one solution (completion) exists. (It's okay if your algorithm takes a long time to run, but you shouldn't enumerate all possibilities.)

   Answer: One possible answer is:

   Use variable elimination to compute the partition function conditioned on the existing numbers. Check if partition function is 1.

   (e) **[4 points total]** Now you'd like to count the total number of distinct, valid solutions to Sudoku (starting from an empty grid). Your first approach is to use importance sampling. Let q be a proposal distribution. Write down an IS estimator $\widehat{S}$ for the total number of solutions using M samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)} \sim q(\mathbf{x})$.

Notation: use $\tilde{p}$ to denote unnormalized probability

$$\tilde{p}(\mathbf{x}) \propto \begin{cases} 1 & \mathbf{x} \text{ is a solution} \\ 0 & \text{otherwise} \end{cases}$$

and use Z to denote the number of valid solutions.

Answer:

$$\mathbb{E}_{q(\mathbf{x})}\left[\frac{\tilde{p}(\mathbf{x})}{q(\mathbf{x})}\right] = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) = Z_p$$

The estimator is thus $1/M \sum_i \tilde{p}(\mathbf{x}^{(i)})/q(\mathbf{x}^{(i)})$.

(f) **[3 points total]** Suppose q is uniformly distributed over all possible $\mathbf{x}$ (not just the valid ones).

i. **[1 points]** Write down an analytic expression for $q(\mathbf{x})$.

ii. **[1 points]** If you draw infinite samples, does importance sampling under this proposal give you the right answer?

iii. **[1 points]** Is this IS scheme likely work in practice? Explain your answer.

Answer: $q(\mathbf{x}) = 1/9^{81}$. It is valid, since the weights are well defined, and never $+\infty$. It is unlikely to work in practice as importance weight is very likely 0, which means the sample is wasted.

(g) **[3 points total]** Designing a good proposal is too hard after all, so you decide to give MCMC a try. Is Gibbs sampling a good choice? Explain why or why not.

Answer: No, Gibbs sampling is a poor choice because each variable has a unique possible values given other variables.

(h) **[7 points total]** Suppose you have access to an efficient black-box algorithm that can produce (exact) samples from $p(\mathbf{x})$ or any conditional distribution $p(\mathbf{x} \mid e)$ for any choice of evidence e. For example, you can draw samples from $p(\mathbf{x} \mid x_2 = 6, x_7 = 3)$. Try to find the most efficient algorithm you can to estimate the number of distinct valid solutions (assuming that you start from an empty square). Ideally:

1) The number of collected samples and computation time should be a low degree polynomial in the size of the grid. For example, it's fine if you collect $81^3 * 100$ samples and take $81^3 * 200$ computer operations, but it is not acceptable to collect 2^{81} samples or take 81^{15} steps to run.

2) Your estimate should be as accurate as possible (e.g., be close to the true value with high probability).

Use a few sentences to argue why your algorithm achieves the above objectives. Proofs are not necessary.

Answer: First estimate $p(x_1 = c_1) = \mathbb{E}_{p(x)}[\mathcal{I}(x_1 = c_1)]$ from samples, pick the c_1 such that $p(x_1 = c_1)$ is largest, record this number as p_1 (in fact, can pick any c_1 such that $p(x_1 = c_1) \neq 0$). Then similarly estimate $p(x_2 = c_2 | x_1 = c_1)$, and pick the c_2 such that this is largest, record this number as p_2. Repeat this process, the total number of solutions, can then be estimated as $\log Z = \log \frac{1}{\prod_{i=1}^k p_i} = -\sum_i \log p_i$.

Short argument for correctness: We can estimate each p_i accurately. Since p_i is not too small (e.g. if we pick largest, $p_i \geq 1/9$), we can estimate $\log p_i$ accurately. Thus we can estimate their sum accurately.

As a fun side fact, the best algorithm should require $O(81^2/\epsilon^2 \log 1/\delta)$ samples and produce estimates between $\log Z \pm \epsilon$ with probability $1 - \delta$ for any $\epsilon > 0, \delta > 0$.

The number of valid Sudoku squares was calculated exactly by Felgenhauer and Jarvis in 2005. There are 6,670,903,752,021,072,936,960 solutions.