IE531: Algorithms for Data Analytics
End-Term Exam
Spring, 2018

1. *(10 points)* (**Gibbs Sampling**) Consider the 9-state Markov Chain shown in figure 1, with the desired stationary probability distribution of

$$\begin{array}{lll} \pi(x_{1,1}) = \frac{8}{37} & \pi(x_{1,2}) = \frac{6}{37} & \pi(x_{1,3}) = \frac{4}{37} \\ \pi(x_{2,1}) = \frac{3}{37} & \pi(x_{2,2}) = \frac{4}{37} & \pi(x_{2,3}) = \frac{2}{37} \\ \pi(x_{3,1}) = \frac{4}{37} & \pi(x_{3,2}) = \frac{4}{37} & \pi(x_{3,3}) = \frac{2}{37} \end{array}$$

Using the *Gibbs Sampling Procedure* find the first row[1] of the 9×9 probability matrix $\mathbf{P}$ such that if $\mathbf{A} = \lim_{k \to \infty} P^k$, then

$$\mathbf{A} = \begin{pmatrix} \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \\ \pi(x_{1,1}) & \pi(x_{1,2}) & \pi(x_{1,3}) & \pi(x_{2,1}) & \pi(x_{2,2}) & \pi(x_{2,3}) & \pi(x_{3,1}) & \pi(x_{3,2}) & \pi(x_{3,3}) \end{pmatrix}$$

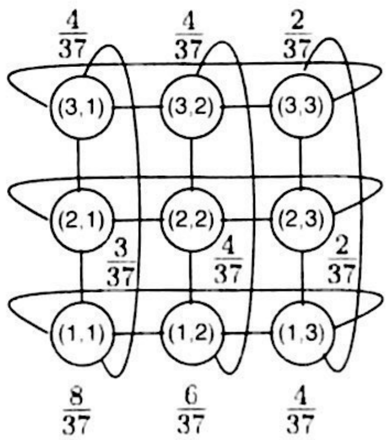

Figure 1: The Markov Chain for problem 1.

[1] I am not asking for the 91 entries here, it would take too long. Just give me the 9 values of the first-row. **Of these 4 will be zero, and five will be non-zero. You get two points for each correct answer – make sure you are working with the correct choice of the first row, here.**

2. (*10 points*) (**Approximate Solution to the** FREQUENT **Problem**) We have a stream $\langle a_1, a_2, \ldots, a_n \rangle$ where each $a_i \in \{1, 2, \ldots m\}$. We have the associated frequency vector $(f_1, f_2, \ldots f_m)$, where $\sum_{i=1}^{n} f_i = n$. We wish to maintain an approximation $(\widehat{f_1}, \widehat{f_2}, \ldots \widehat{f_m})$ to the frequency vector $(f_1, f_2, \ldots f_m)$, such that $\forall j, |f_j - \widehat{f_j}| \leq \epsilon n$, for a desired $\epsilon > 0$.

   In section 1.2 of my notes on the material in Chapter 6 of the text, I introduced you a solution to the above problem that kept track of l-many keys (and their associated counters) where $l \ll n$. This procedure required us to maintain at least one empty key (with its counter zeroed-out) at all instants. This was accomplished by subtracting the median of the counter-values from each counter.

   We noted in class that it was an overkill to subtract the median from all counter-values, and we could have used the minimum in place of the median. Derive an expression for the value of l (i.e. the number of keys-with-their-counters) that would accomplish the objective when we use the minimum counter-value instead of the median.

3. (*10 points*) (**Chaining**) Suppose we built a storage-scheme for n-many objects in a set S by a hash-table of size m using *Chaining*.

   (a) (*1 point*) Can $n > m$?

   (b) (*1 point*) What is a *False-Negative Error*? What is a *False-Positive Error*? Could we them in our scheme?

   (c) (*2 points*) Derive an expression for the probability of seeing an i-long chain.

   (d) (*2 points*) Derive an expression for the average number of empty-slots in the hash-table.

   (e) (*2 points*) Derive an expression for the average number of collisions in the hash-table.

   (f) (*2 points*) Derive an upper-bound for the probability that the longest-chain will have a length of i.

4. (*10 points*) (**Deletions in Open-Addressing**) Suppose we built a storage-scheme for n-many objects in a set S by a hash-table of size m using *Open Addressing*.

   (a) (*2 points*) Can $n > m$?

   (b) (*2 points*) Assuming we have not deleted anything, can we have any False-Positive or False-Negative errors?

   (c) (*2 points*) Suppose we used a naive scheme of erasing/deleting the hash-table entry for an object whenever it is to be removed (or sold), can we have any False-Positive or False-Negative errors?

   (d) (*2 points*) Is there an alternative to the naive scheme of problem 4c for erasing/deleting objects? (Just three/four sentences with a cogent logical thought is sufficient for this problem)

(e) (*2 points*) Let us suppose we did what ever you say we should do in your solution to problem 4d which is better than the naive scheme of erasing/deleting objects. Following this, how do we place new-items in the hash-table? (Just three/four sentences with a cogent logical thought is sufficient for this problem)

5. (*10 points*) (**Deletions in Bloom Filters**) Suppose we built an n-long *Bloom Filter* for m-many objects, using k-many hash-functions. We observed that for regular/vanilla Bloom Filters, there can be no False Negative Errors, while it possible to have False Positive Errors. We showed in class that the

$$Prob\{\text{False Positive Error}\} = (1 - e^{-km/n})^k.$$

and if we set $k = \frac{n}{m} \ln 2 \approx (0.7n)/m$, then

$$Prob\{\text{False Positive Error}\} = (1 - e^{-km/n})^k = (1/2)^k = (1/2)^{(\ln 2)(n/m)} = (0.615)^{n/m}.$$

If $Prob\{\text{False Positive Error}\} = p$, then[2]

$$p = (0.615)^{n/m} \Rightarrow \frac{n}{m} = \frac{\log_2 p}{\log_2 0.615} = 0.7 \log_2 p.$$

(a) (*2 points*) Suppose we used a regular/vanilla Bloom Filter, and we naively deleted the k-many bit positions for any object that we thought was in our inventory, but could not find it at the spot is should be. Can we have False Negative Errors after this operation? Explain.

(b) (*4 points*) Explain how a *Counting Bloom Filter* is different from regular/-vanilla Bloom Filters. What is the process by which objects get inserted in this filter? What is the process by which objects get deleted from this filter?

(c) (*4 points*) Without getting into the nitty-gritty mathematical formulae, tell me (in words) how the *number of bits per counter* for a Counting Bloom Filter is decided. What was the logic behind saying 4-bits are sufficient per counter for a typical Counting Bloom Filter.

6. (*10 points*) (**Random Sampling of an Unknown Stream**) Describe a process by which you can choose a single-sample that is uniformly-random from a stream $\langle a_1, a_2, \ldots, a_n \rangle$, when you have no ia priori idea of what n is going to be. Or stated differently, you will only know what n is after you have seen the last member of the stream.

(**PS**: I am looking for (1) a sampling-algorithm, and (2) a proof that if so far we have seen $\langle a_1, a_2, \ldots, a_k \rangle$ where $k \leq n$, then the algorithm would have picked any member of the stream with equal probability (i.e with probability $\frac{1}{k}$.)

[2] As an illustration, we took the case of $p \leq \frac{1}{1000}$ then $\frac{n}{m} \approx 0.7 \log_2 (1/1000) \approx 7$ and $k = 0.7 \times 7 \approx 5$. That is, if we are happy with a 1/1000 chance of a false positive error, then $k \approx 5$ and $n/m \approx 7$, which is quite good

7. *(10 points)* **(Length-Squared Sampling)** Suppose $\mathbf{A}$ (resp. $\mathbf{B}$) is an $m \times n$ (resp. $n \times p$) matrix, and we wish to compute the product $\mathbf{AB}$ using sampling. That is, we pick the k-th column of $\mathbf{A}$ (i.e. in MATLAB-notation, we pick $\mathbf{A}(:,k)$) with probability p_k, and multiply it with the k-th row of $\mathbf{B}$, divide the result by p_k, and present

$$\mathbf{X} = \frac{1}{p_k}\mathbf{A}(:,k)\mathbf{B}(k,:)$$

as the "aproximation" to $\mathbf{AB}$.

   (a) *(3 points)* Show that $\mathbf{X}$ is an unbiased-estimator of $\mathbf{AB}$. That is, show $E\{\mathbf{X}\} = \mathbf{AB}$.

   (b) *(3 points)* Present an expression for the variance of $\mathbf{X}$.

   (c) *(4 points)* Using your expression for problem 7b, compute the best choice of p_k that minimizes the variance of $\mathbf{X}$. Assume $\mathbf{B} = \mathbf{A}^T$ for this part of the problem.

8. *(10 points)* **(The Perceptron Algorithm)** In a 2-class problem using two features, we have the following four training vectors

$$\mathcal{P} = \{(0,0)^T, (2,0)^T\}$$
$$\mathcal{N} = \{(0,1)^T, (2,2)^T\}$$

   Use the Perceptron Criterion with $\rho_k = 1$ for all k to determine the solution vector $\mathbf{w}$. Use the all-zero vector as the starting vector as the initial-value for $\mathbf{w}$. Show the weight-vector after each iteration.

9. *(10 points)* **(Linear Separability)** We have training data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathcal{R}^d$, and let us suppose each $\mathbf{x}_i$ has a label $y_i \in \{-1, +1\}$. The set of data points $\mathbf{x}_i$ with $y_i = +1$ (resp. $y_i = -1$) will be referred to as the set of *positive* (resp. *negative*) examples, and will be denoted by $\mathcal{P}$ (resp. $\mathcal{N}$).

   In class we looked at Pedroso and Murata's LP that can be used to check if the data is linearly separable/non-separable. We suppose there is a hyperplane $H_{\mathbf{w},b} : \mathbf{w}^T\mathbf{x} + b = 0$ that separates the positive from negative examples. For non-separable data we require

   (a) $\forall \mathbf{x}_i \in \mathcal{P}, \mathbf{w}^T\mathbf{x}_i + b \geq +1 - \xi_i$,

   (b) $\forall \mathbf{x}_i \in \mathcal{N}, \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i$,

   (c) $\xi_i \geq 0, \forall i$, and

   with the objective function

$$\underset{\mathbf{w},b,\xi}{\text{minimize}} \left\{ \|\mathbf{w}\|^2 + \left(\sum_i \xi_i\right) \right\}.$$

Following some observations involving *conjugate-norms*, this resulted in the following LP

$$\begin{array}{ll} \text{minimize} & a + C\left(\sum_{i\in\mathcal{P}} \xi_i^+ + \sum_{i\in\mathcal{N}} \xi_i\right) \\ & \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i^+, \forall i \in \mathcal{P}, \\ & \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i^-, \forall i \in \mathcal{N}, \\ \text{subject to} & a \geq \mathbf{w}_j, \forall j \in \{1, 2, \ldots, d\}, \\ & a \geq -\mathbf{w}_j, \forall j \in \{1, 2, \ldots, d\}, \\ & a, b \in \mathcal{R}; \mathbf{w} \in \mathcal{R}^d; \forall i, \xi_i^+ > 0; \forall i, \xi_i^- > 0 \end{array}$$

Use the above LP formulation to investigate the Linear Separability of the following 2-class data sets.

(a) (*5 points*) Determine if the following 2-class data set is linearly separable, where

$$\mathcal{P} = \{(0,1)^T, (1,0)^T\}$$
$$\mathcal{N} = \{(0,0)^T, (1,1)^T\}$$

(b) (*5 points*) Determine if the following 2-class data set is linearly separable, where

$$\mathcal{P} = \{(-1,-1)^T, (0,0)^T, (1,1)^T\}$$
$$\mathcal{N} = \{(-1,1)^T, (1,-1)^T\}$$

(**PS**: Any attempt as showing these (rather obvious) problems by any other method will not get you any points! To paraphrase Obi-Wan Kenobi – "*Use the LP, Luke!*")

10. (*10 points*) (**Decision Regions of a "Neural Network" constructed from Linear Discriminants**) We will use the *committee machine* shown in figure 2 for a 2-class classification problem. A test vector $\mathbf{x} = (x_1, x_2)^T$ goes through the three linear discriminant functions to produce three outputs y_1, y_2 and y_3. These outputs are *binarized* to yield z_1, z_2 and z_3 respectively. The value of z_i is $+1$ (resp. -1) if the input y_i is positive (resp. not positive). The discriminant n is obtained as $z_1 + z_2 + z_3$. If $n > 0$, we choose class $\mathcal{P}$, otherwise we choose class $\mathcal{N}$. Derive and sketch the decision regions implied by this committee machine for the 2-class problem.

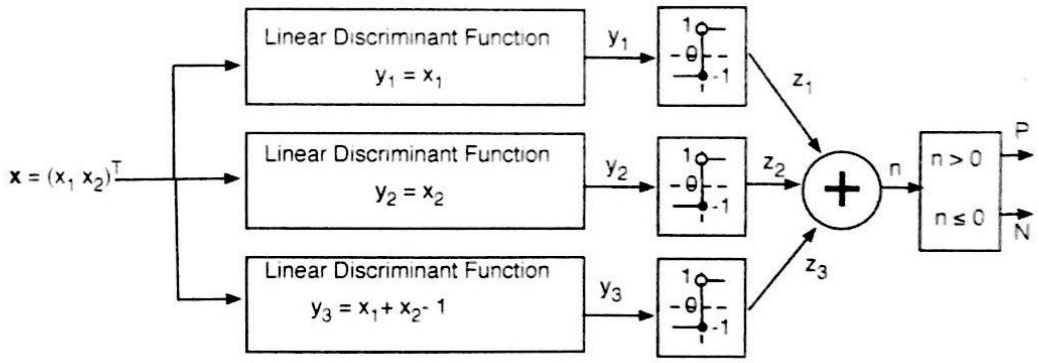

Figure 2: The Committee Machine for Problem 10.