Final

Last Name: Chakraborty

First Name: Dhruv

Student ID: 204-962-098

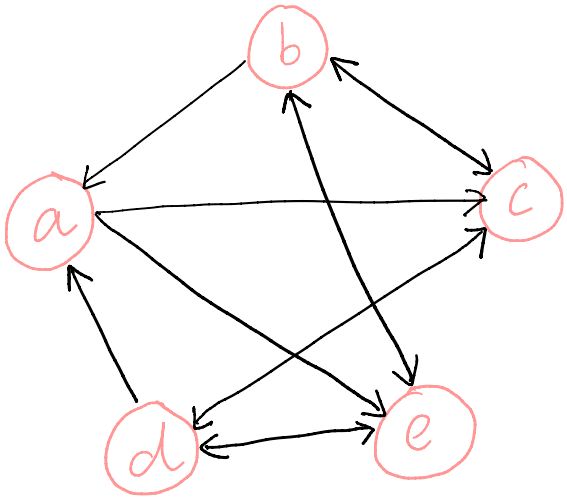
Instructions: This is an open book exam, so you may use your notes, lecture slides and videos, and content on Piazza. The only thing you are not allowed to do is ask someone else for help. This includes asking your classmates questions over Piazza, Group.me, email etc; consulting with other friends or family members; and using online Q& A services.

Please do not write below this line.

Question	Points	Score
1	20	
2	10	
3	10	
4	15	
5	15	
6	10	
Total:	80	

1. Consider the directed graph with adjacency matrix $A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$ a b c d e

(a) (5 points) Sketch the graph. Number the vertices and draw in all directed edges, clearly showing their orientation.



(b) (5 points) Compute $\tilde{P}$ using $\alpha = 0.05$, where $\tilde{P}$ is as defined on slide 7 of Lecture 13.

We first calculate $P = A^T(D_{out})^{-1}$ where $D_{out} = \text{diag}(d_a^{out}, \ldots, d_e^{out})$.

$$\therefore P = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 \end{pmatrix} = \begin{pmatrix} 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 1/3 & 0 & 1/3 & 0 \end{pmatrix}.$$

Now we compute $\tilde{P}$ as follows:

$$\tilde{P} = \alpha P + (1-\alpha)\frac{1}{n}\vec{1}\vec{1}^T = \begin{pmatrix} 0.19 & 0.20667 & 0.19 & 0.20667 & 0.19 \\ 0.19 & 0.19 & 0.215 & 0.19 & 0.215 \\ 0.215 & 0.20667 & 0.19 & 0.20667 & 0.19 \\ 0.19 & 0.19 & 0.215 & 0.19 & 0.215 \\ 0.215 & 0.20667 & 0.19 & 0.20667 & 0.19 \end{pmatrix}$$

Question 1 continues on the next page...

(c) (5 points) Let $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k)}, \ldots$ be the iterates produced by the Power Method. Explain why $\lim_{k\to\infty} \boldsymbol{v}^{(k)} = \boldsymbol{v}_1$, the leading eigenvector of $\tilde{P}$.

We know that $\tilde{P}$ is column stochastic so that $\rho(\tilde{P}) = 1$. Moreover since $\tilde{P}$ is non-negative and irreducible, Perron-Frobenius applies and we have that $\lambda_1(\tilde{P}) = \rho(\tilde{P}) = 1$ and that there is a unique eigenvector $\vec{v}_1 > 0$ associated with $\lambda_1(\tilde{P})$. Supposing that $\tilde{P}$ has n distinct eigenvalues $\lambda_1 = 1 > |\lambda_2| > \ldots > |\lambda_n|$, we can write any $\vec{v} \in \mathbb{R}^n$ as a linear combination of $\tilde{P}$'s eigenvectors i.e. $\vec{v} = \beta_1\vec{v}_1 + \beta_2\vec{v}_2 + \ldots + \beta_n\vec{v}_n$. But notice that multiplying both sides by $\tilde{P}$ gives:

$$\vec{v}^{(0)} = \tilde{P}\vec{v} = \tilde{P}(\beta_1\vec{v}_1 + \ldots + \beta_n\vec{v}_n) = \beta_1\tilde{P}\vec{v}_1 + \ldots + \beta_n\tilde{P}\vec{v}_n$$
$$= \beta_1\lambda_1\vec{v}_1 + \ldots + \beta_n\lambda_n\vec{v}_n$$

and similarly, $\vec{v}^{(k)} = \tilde{P}^k\vec{v} = \beta_1\lambda_1^k\vec{v}_1 + \ldots + \beta_n\lambda_n^k\vec{v}_n$.

But note that $\lambda_1 = 1 > |\lambda_2| > \ldots > |\lambda_n| \Rightarrow$ as $k \to \infty$
$\vec{v}^{(k)} = \beta_1(1)^k\vec{v}_1 + \ldots + \beta_n\lambda_n^k\vec{v}_n$

$\therefore \lim_{k\to\infty} \vec{v}^{(k)} = \vec{v}_1$ (since we are okay with scalar multiple of eigenvector)

(d) (5 points) Use PageRank to rank the vertices of this graph from most to least important. You may use any method you like (including built-in functions in Python or Matlab) to compute $\boldsymbol{v}_1$.

Using the pagerank() function in the networkx module of python [nx.pagerank(nx.from_numpy_matrix(A, create_using=nx.DiGraph()))]

results in the following ranking:

1. c, e (both have score = 0.21355)
2. b, d (both have score = 0.21152)
3. a (score = 0.14986)

i.e. $\vec{v}_1 = \begin{pmatrix} 0.14986 \\ 0.21152 \\ 0.21355 \\ 0.21151 \\ 0.21355 \end{pmatrix}$.

2. (10 points) Convert the following linear programming problem to standard form:

$$\begin{aligned} \text{minimize } & 2x_1 + x_2 \\ \text{subject to: } & x_1 + x_2 \leq 3 \\ & x_1 + 2x_2 \leq 5 \\ & x_1 \geq 0 \text{ and } x_2 \geq 0 \end{aligned}$$

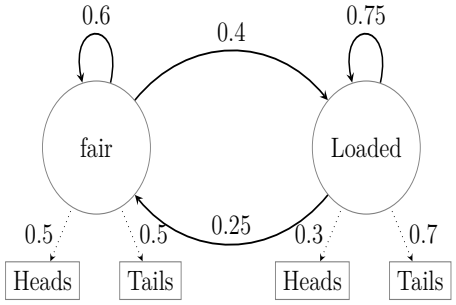
We introduce 2 slack variables $x_3 = 3 - x_1 - x_2$ and $x_4 = 5 - x_1 - 2x_2$ s.t. when $x_3 \geq 0$ we have $3 - x_1 - x_2 \geq 0$ $\Rightarrow x_1 + x_2 \leq 3$ and when $x_4 \geq 0$, $5 - x_1 - 2x_2 \geq 0$ so $x_1 + 2x_2 \leq 5$.

Now let $A = \begin{pmatrix} -1 & -1 & -1 & 0 \\ -1 & -2 & 0 & -1 \end{pmatrix}$ and $\vec{b} = \begin{pmatrix} -3 \\ -5 \end{pmatrix}$ so that we can write our problem as minimizing $2x_1 + x_2$ with $A\vec{x} = \vec{b}$ and $\vec{x} \geq 0$ $\left(\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}\right)$. Finally let $\vec{c} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

and we can write our problem is standard form as minimizing $\vec{c}^T\vec{x}$ subject to $\vec{x} \geq 0$ and $A\vec{x} = \vec{b}$.

UID: 204-962-098

3. (10 points) Consider the dishonest casino problem modeled by the Hidden Markov Model (HMM) given below. Suppose we observe the sequence of emissions $\boldsymbol{Y}_6 =$



$HTTTHT$. Use Viterbi's algorithm to find the most probable sequence of hidden states. (Here, the hidden states are of course "F" and "L"). Assume an initial distribution of $\mathbb{P}[x_1 = F] = 0.5$ and $\mathbb{P}[x_1 = L] = 0.5$. Your answer should include the dynamic programming table you construct as part of the algorithm.

Using $A_{ij} = P(x_{t+1} = i \mid x_t = j)$, $E_{ij} = P(y_t = e_j \mid x_t = i)$ and the given diagram we have $A = \begin{pmatrix} 0.6 & 0.25 \\ 0.4 & 0.75 \end{pmatrix}$ and $E = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}$.

We use the recursion $f(i,k) = E_{i y_k} \max_r A_{ir} f(r, k-1)$ and $P(x_1 = F) = 0.5$, $P(x_1 = L) = 0.5$ to generate the following table:

$f(F,1) = P(y_1 \mid x_1 = F) P(x_1 = F) = P(H \mid F) P(F) = (0.5)(0.5) = 0.25$

$f(L,1) = P(y_1 \mid x_1 = L) P(x_1 = L) = P(H \mid L) P(L) = (0.3)(0.5) = 0.15$

$f(F,2) = P(y_2 \mid x_2 = F) \max \begin{cases} 0.6\,(0.25) \\ 0.25\,(0.15) \end{cases} = (0.5)(0.6)(0.25) = 0.075$

$f(L,2) = P(y_2 \mid x_2 = L) \max \begin{cases} 0.4\,(0.25) \\ 0.75\,(0.15) \end{cases} = (0.7)(0.75)(0.15) = 0.07875$

The full table below was generated by code, and a screenshot of this is appended to the exam.

	k=1	k=2	k=3	k=4	k=5	k=6
$x_t = F$	0.25	0.075	0.0225	0.008269	0.004341	0.0013
$x_t = L$	0.15	0.07875	0.04134	0.0217	0.004884	0.0027

Using traceback we find the most probable sequence of states

$X = LLLLLL$

4. We say that a function $f(\boldsymbol{x})$ is *concave* if $-f(\boldsymbol{x})$ is convex.

(a) (5 points) Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ is concave. Prove that every stationary point of $f(\boldsymbol{x})$ is a global maximum.

Let x^* be a stationary point of $f(x)$ s.t. $\nabla f(x^*) = 0$ and let $y \in \mathbb{R}^d$ be any other point. Let $g(x) = -f(x)$ so that $g(x)$ is convex. Then we know the tangent line at x is always below $g(x)$'s graph i.e. we have $g(y) \geq g(x^*) + \nabla g(x^*)(y - x^*) \Rightarrow g(y) \geq g(x^*)$ $\because \nabla g(x^*) = 0$ ($= -\nabla f(x^*)$). multiplying both sides of the inequality by -1 flips the sign to give us $-g(y) \leq -g(x^*) \Rightarrow \underline{f(y) \leq f(x^*)}$.

$\therefore$, x^* is a global maximum.

(b) (5 points) Show that $f(x) = 4 + 3x - 2x^4$ is concave.

Consider $g(x) = -f(x) = 2x^4 - 3x - 4$.

Then $g'(x) = 8x^3 - 3 \Rightarrow g''(x) = 24x^2$.

Note that $g''(x) \geq 0$ i.e. $g''(x)$ is non-negative (enough for positive semi-definiteness in $\mathbb{R}$)

so that $g(x)$ is convex.

$\therefore$ $f(x) = -g(x)$ is concave.

(c) (5 points) Write down pseudocode for a gradient-based algorithm to solve $\arg\max_{x \in \mathbb{R}} f(x)$ where $f(x)$ is as in part (b). Explicitly compute any gradients you may use.

Input: α = step size; K = num iterations; x_1 = initial pt

for $k = 1, \ldots, K$ do

compute $f'(x_k) = 3 - 8x^3$

Let $x_{k+1} = x_k + \alpha f'(x_k)$

// go in direction of steepest increase

end for

output x_K

UID: 204-962-098

5. Consider the problem of partitioning a data set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ into 3 classes: $X = X_1 \cup X_2 \cup X_3$. In class we discussed how one can view this as an optimization problem by defining a partition vector $\Pi \in \{1, 2, 3\}^n$ which encodes a partition as:

$$X_a = \{\boldsymbol{x}_i : \ \Pi_i = a\}$$

Recall that we introduced the following quality score:

$$Q(\Pi) = \sum_{a=1}^{3} \sum_{j:\Pi_j = a} \|\mu_a - \mathbf{x}_j\|_2^2 \quad \text{where } \mu_a = \frac{1}{|X_a|} \sum_{\boldsymbol{x}_i \in X_a} \boldsymbol{x}_i$$

(a) (5 points) Suppose that $\Pi^\star = \underset{\Pi \in \{1,2,3\}^n}{\arg\min}\, Q(\Pi)$. Why will $\Pi^\star$ correspond to a good partition of the data? Justify your answer.

Finding a $\Pi^* = \underset{\Pi \in \{1,2,3\}^n}{\text{argmin}}\, Q(\Pi)$ corresponds to a good partition $\because X_a = \{x_i : \Pi_i = a\}$ encodes all possible partitions and $Q(\Pi)$ returns the sum of the Euclideam distance between each point and the corresponding centroid. Other distance functions (kernels) used here would lead to potentially different but also good partitions.

(b) (5 points) Your friend suggests finding $\Pi^\star$ using gradient descent applied to $Q(\Pi)$. Why is this not going to work? Justify your answer.

This won't work because finding a Π that minimizes $Q(\Pi)$ is not a convex problem and moreover is a discrete optimation problem. Gradient descent and similar approaches are only guaranteed to work for convex functions.

(c) (5 points) Write pseudocode for the k-means algorithm (also known as Lloyd's algorithm) applied to this problem of partitioning X into three clusters (*i.e.* $k = 3$).

Input data points $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$, K = num iterations
randomly initialize $\mu_1, \mu_2, \mu_3 \in \mathbb{R}^d$
for $k = 1, \ldots, K$ do
  assign x_i to cluster X_a if it is closest to μ_a
  reset $\mu_a = \frac{1}{|X_a|} \sum_{x_j \in X_a} x_j$
end for
return μ_1, μ_2, μ_3.

6. In class we studied the logistic regression model for binary classification. Recall that we defined the loss functions as:

$$\ell_i(\boldsymbol{\theta}, b) := -y^{(i)} \log(\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)} + b)) - (1 - y^{(i)}) \log\left(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)} + b)\right)$$
$$L(\boldsymbol{\theta}, b) = \sum_{i=1}^{N} \ell_i(\boldsymbol{\theta}, b)$$

We showed that one can find the best parameters, $\boldsymbol{\theta}^\star, b^\star$, by minimizing the loss function given above. That is, $\boldsymbol{\theta}^\star, b^\star = \arg\min_{\boldsymbol{\theta}, b} L(\boldsymbol{\theta}, b)$.

(a) (5 points) Why did we need to assume that the training data pairs $(\boldsymbol{x}^{(i)}, y^{(i)})$ are independent? *(Hint: it has to do with the likelihood function).*

We need to assume the training data pairs are independent so that we could get the likelihood function in terms of the individual data points: $P(\vec{y} \mid X; \theta, b) = \prod_{i=1}^{N} P(y^{(i)} \mid \vec{x}^{(i)}; \theta, b)$. We use the log of this product to get a convex function to minimize. Without independence we do not have a nice function.

(b) (5 points) It is frequently useful to consider the *regularized* loss function:

$$\tilde{L}(\boldsymbol{\theta}, b) = \sum_{i=1}^{N} \ell_i(\boldsymbol{\theta}, b) + \|\boldsymbol{\theta}\|_2^2.$$

Compute $\nabla_{\boldsymbol{\theta}} \tilde{L}$. Show all your work *(even though some of it will overlap with work you have done in a homework assignment).*

We first find $\nabla_\theta \ell_i(\theta, b) = \nabla_\theta \left[-y^{(i)} \log(\sigma(\theta^T x^{(i)} + b)) - (1 - y^{(i)}) \log(1 - \sigma(\theta^T x^{(i)} + b)) \right]$

$$= -y^{(i)} x^{(i)} \left[\frac{\sigma(\theta^T x^{(i)} + b)(1 - \sigma(\theta^T x^{(i)} + b))}{\sigma(\theta^T x^{(i)} + b)} \right] + (1 - y^{(i)}) x^{(i)} \left[\frac{\sigma(\theta^T x^{(i)} + b)(1 - \sigma(\theta^T x^{(i)} + b))}{1 - \sigma(\theta^T x^{(i)} + b)} \right]$$

$$\nabla_\theta \ell_i(\theta, b) = x^{(i)} \left(\sigma(\theta^T x^{(i)} + b) - y^{(i)} \right)$$

and $\nabla_\theta (\|\theta\|_2^2) = 2\theta$ so

$$\nabla_\theta \tilde{L} = \sum_{i=1}^{N} x^{(i)} \left(\sigma(\theta^T x^{(i)} + b) - y^{(i)} \right) + 2\theta.$$

[49]

```
def viterbi(y, A, E, pi):
    K = A.shape[0]
    T = len(y)
    T1 = np.empty((K, T), 'd')
    T2 = np.empty((K, T), 'B')

    T1[:, 0] = pi * E[:, y[0]]
    T2[:, 0] = 0

    for i in range(1, T):
        T1[:, i] = np.max(T1[:, i - 1] * A.T * E[np.newaxis, :, y[i]].T, 1)
        T2[:, i] = np.argmax(T1[:, i - 1] * A.T, 1)

    x = np.empty(T, 'B')
    x[-1] = np.argmax(T1[:, T - 1])
    for i in reversed(range(1, T)):
        x[i - 1] = T2[x[i], i]

    return x, T1, T2

y = np.array([0,1,1,1,0,1])
A = np.array([[0.6, 0.25],[0.4, 0.75]])
E = np.array([[0.5,0.5], [0.3,0.7]])
init = np.array([0.5, 0.5])

viterbi(y, A, E, init)
```

```
(array([1, 1, 1, 1, 1, 1], dtype=uint8),
 array([[0.25      , 0.075     , 0.0225    , 0.00826875, 0.00434109,
         0.00130233],
        [0.15      , 0.07875   , 0.04134375, 0.02170547, 0.00488373,
         0.00256396]]),
 array([[0, 0, 0, 1, 1, 0],
        [0, 1, 1, 1, 1, 1]], dtype=uint8))
```