Take-home Exam

Deadline: January 12, 2024, 23:59

This document contains the instructions for the take-home exam in the "Advanced Probabilistic Machine Learning" course (SSY316). The exam is structured into four parts, and the corresponding datasets for each section can be accessed through the course Web site. Each part requires the submission of a project report. All components of the exam are to be completed in groups of two individuals, as previously registered. The total exam score is 100 points, with 20 points allocated to the first part and 30,30 and 20 points for the second, third, and fourth parts respectively.

---

1 Principal Component Analysis (PCA) of Genomes [20 pts]

In this part, you will run PCA on a real data set, and interpret the output. **Description:** Download the p4dataset2023.txt file from the course Web site. The data represented there is from the 1000 genomes project. Each of the 995 lines in the file represents an individual. The first three columns represent respectively the individual's unique identifier, their sex (M=male, F=female) and the population they belong to—see the file p4dataset2023decoding.txt for the decodings of these population tags. The subsequent 10101 columns of each line are a subsample of nucleobases from the individual's genome. We will be looking at the output of PCA on this dataset. PCA can refer to a number of related things, so to be explicit, in this section when we say "PCA" we mean

- The data should be centered (i.e., the sample mean subtracted out) but not normalized, so its alright if some dimensions have different variance than other dimensions.

- The output should be the normalized principal components (i.e., unit-length vectors)

Feel free to use a library implementation of PCA for the following questions. We recommend scikit learn's implementation. Note that with python scikit, you can specify how many principal components you want (this can save on computation time).

Exercises: First convert the data from the text file of nucleobases to a real-valued matrix (PCA needs a real-valued matrix). Specifically, convert the genetic data into a binary matrix X such that $X_{i,j} = 0$ if the i^{th} individual has column j's mode nucleobase[1] for their j^{th} nucleobase, and $X_{i,j} = 1$ otherwise. Note that all mutations appear as a 1, even if they are different mutations, so if the mode for column j is "G", then if individual i has an "A","T", or "C", then $X_{i,j}$ would be 1.

The first 3 columns of the data file provide meta-data, and should be ignored when creating the binary matrix X. We will examine genotypes to extract phenotype information.

1. (1 points) Say we ran PCA on the binary matrix X above. What would be the dimension of the returned vectors?

2. (4 points) We will examine the first 2 principal components of X. These components contain lots of information about our data set. Create a scatter plot with each of the 995 rows of X projected onto the first two principal components. In other words, the horizontal axis should be v1, the vertical axis v2, and each individual should be projected onto the subspace spanned by v1 and v2. Your plot must use a different color for each population and include a legend.

3. (5 points) In two sentences, list 1 or 2 basic facts about the plot created in part (b). Can you interpret the first two principal components?

[1] By "mode nucleobase", we just mean the most frequently occurring nucleobase in that position (across the 995 data points).

What aspects of the data do the first two principal components capture? Hint: think about history and geography.

4. (3 points) We will now examine the third principal component of X. Create another scatter plot with each individual projected onto the subspace spanned by the first and third principal components. After plotting, play with different labeling schemes (with labels derived from the meta-data) to explain the clusters that you see. Your plot must include a legend.

5. (4 points) Something should have popped out at you in the plot above. In one sentence, what information does the third principal component capture?

6. (3 points) In this part, you will inspect the third principal component. Plot the nucleobase index vs the absolute value of the corresponding value of the third principal component in that index. (The x-axis of your plot should go from 1 to 10101—you're literally just plotting the 10101 values in the third principal component.) What do you notice? What's a possible explanation? Hint: think about chromosomes (and if you don't know much biology, feel free to look through the wikipedia page on chromosomes....)

Deliverables: Scatter plot for part (b). Short discussion for part (c). Scatter plots for parts (d) and (f). One sentence answers for (e) and (f). Code for the whole section which doesn't have to be separated into parts.

2 Markov Chain Monte Carlo [30 pts]

Nowadays, statistical modelling of sport data has become an important part of sports analytics and is often a critical reference for the managers in their decision-making process. In this part, we will work on a real world example in

professional sports. Specifically, we are going to use the data from the 2013-2014 Premier League, the top-flight English professional league for men's football clubs, and build a predictive model on the number of goals scored in a single game by the two opponents. Bayesian hierarchical model is a good candidate for this kind of modeling task. We model each team's strength (both attacking and defending) as latent variables. Then in each game, the goals scored by the home team is a random variable conditioned on the attacking strength of the home team and the defending strength of the away team. Similarly, the goals scored by the away team is a random variable conditioned on the attack strength of the away team and the defense strength of the home team. Therefore, the distribution of the scoreline of a specific game is dependent on the relative strength between the home team A and the away team B, which also depends on the relative strength between those teams with their other opponents.

Table 1: 2013-2014 Premier League Teams

Index	0	1	2	3	4
Team	Arsenal	Aston Villa	Cardiff City	Chelsea	Crystal Palace
Index	5	6	7	8	9
Team	Everton	Fulham	Hull City	Liverpool	Manchester City
Index	10	11	12	13	14
Team	Manchester United	Newcastle United	Norwich City	Southampton	Stoke City
Index	15	16	17	18	19
Team	Sunderland	Swansea City	Tottenham Hotspurs	West Bromwich Albion	West Ham United

The Premier League has 20 teams, and we index them as in Table 1. Each

team would play 38 matches every season (playing each of the other 19 teams home and away), which totals 380 games in the entire season. For the g-th game, assume that the index of home team is $h(g)$ and the index of the away team is $a(g)$. The observed number of goals (y_{g0}, y_{g1}) of home and away team is modeled as independent Poisson random variables:

$$y_{gj}|\theta_{gj} \sim \text{Poisson}(\theta_{gj}), \quad j = 0, 1 \tag{1}$$

where $\theta = (\theta_{g0}, \theta_{g1})$ represents the scoring intensity in the g-th game for the team playing at home ($j = 0$) and away ($j = 1$), respectively. We put a log-linear model for the θs:

$$\log \theta_{g0} = home + att_{h(g)} - def_{a(g)} \tag{2}$$

$$\log \theta_{g1} = att_{a(g)} - def_{h(g)} \tag{3}$$

Note that team strength is broken into attacking and defending strength. And home represents home-team advantage, and in this model is assumed to be constant across teams. The prior on the home is a normal distribution:

$$home \sim \mathcal{N}(0, \tau_0^{-1}) \tag{4}$$

where we set the precision $\tau_0 = 0.0001$.
The team-specific attacking and defending effects are modeled as:

$$att_t \sim \mathcal{N}(\mu_{att}, \tau_{att}^{-1}) \tag{5}$$

$$def_t \sim \mathcal{N}(\mu_{def}, \tau_{def}^{-1}) \tag{6}$$

We use conjugate priors as the hyper-priors on the attack and defense means and precisions:

$$\mu_{att} \sim \mathcal{N}(0, \tau_1^{-1}) \tag{7}$$

$$\mu_{def} \sim \mathcal{N}(0, \tau_1^{-1}) \tag{8}$$

$$\tau_{att} \sim \text{Gamma}(\alpha, \beta) \tag{9}$$

$$\tau_{def} \sim \text{Gamma}(\alpha, \beta) \tag{10}$$

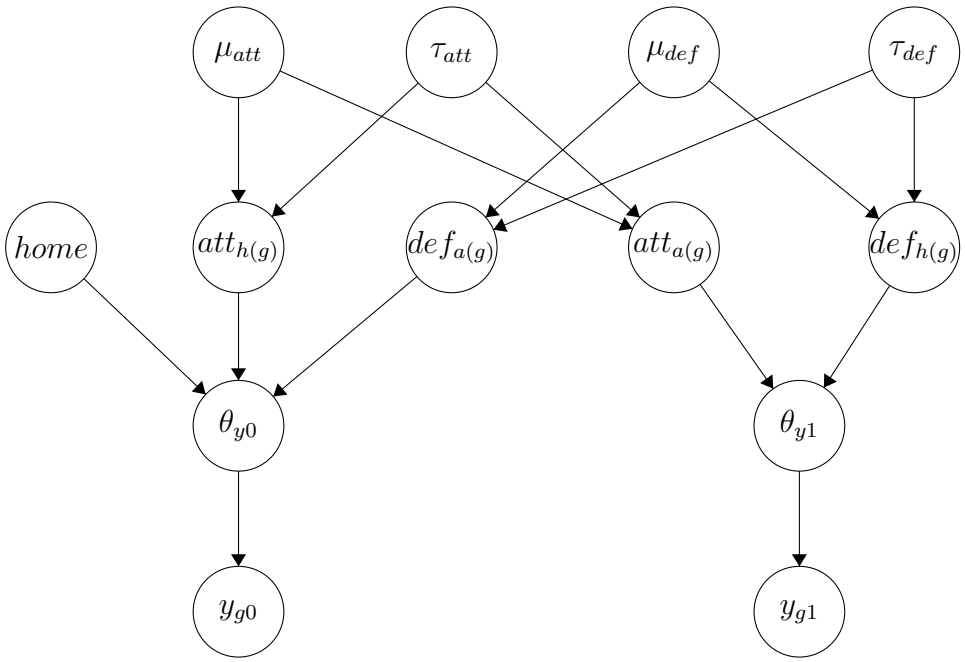

Figure 1: The DAG representation of the hierarchical Bayesian model.

where the precision $\tau_1 = 0.0001$, and we set parameters $\alpha = \beta = 0.1$.
This hierarchical Bayesian model can be represented using a directed acyclic graph as shown in Figure 1.
The goals of each game are $\mathbf{y} = \{y_{gj} | g = 0, 1, ..., 379, j = 0, 1\}$ are the observed variables, and parameters $\boldsymbol{\theta} = \{home, att_0, def_0, ..., att_{19}, def_{19}\}$ and hyper-parameters $\boldsymbol{\eta} = (\mu_{att}, \mu_{def}, \tau_{att}, \tau_{def})$ are unobserved variables that we need to make inference on. To ensure identifiability, we enforce a corner constraint on the parameters (pinning one team's parameters to 0,0). Here we use the first team as reference and assign its attacking and defending strength to be 0:

$$att_0 = def_0 = 0 \tag{11}$$

In this question, we want to estimate the posterior mean of the attacking and defending strength for each team, i.e. $\mathbb{E}_{p(\boldsymbol{\theta},\boldsymbol{\eta}|\mathbf{y})}[att_i]$, $\mathbb{E}_{p(\boldsymbol{\theta},\boldsymbol{\eta}|\mathbf{y})}[def_i]$, and $\mathbb{E}_{p(\boldsymbol{\theta},\boldsymbol{\eta}|\mathbf{y})}[home]$.

1. **[6 points]** Find the joint likelihood $p(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\eta})$.

2. **[6 points]** Write down the Metropolis-Hastings algorithm for sampling from posterior $p(\boldsymbol{\theta}, \boldsymbol{\eta}|\mathbf{y})$, and derive the acceptance function for a proposal distribution of your choice (e.g. isotropic Gaussian).

3. **[18 points]** Implement the Metropolis-Hastings algorithm to inference the posterior distribution. The data can be found from `https://chalmers.instructure.com/files/3108991/download?download_frd=1`, which contains a 380×4 matrix. The first column is the number of goals y_{g0} scored by the home team, the second column is the number of goals y_{g1} scored by the away team, the third column is the index for the home team $h(g)$, and the fourth column is the index for the away team $a(g)$.

   - Use an isotropic Gaussian proposal distribution $\mathcal{N}(0, \sigma^2 I)$ and use 0.1 as the starting point.
   - Run the MCMC chain for 5000 steps to burn in and then collect 5000 samples with t steps in between (i.e., run M-H for $5000t$ steps and collect only each t-th sample). This is called thinning, which reduces the autocorrelation of the MCMC samples introduced by the Markovian process. The parameter sets are $\sigma = 0.005, 0.05, 0.5$, and $t = 1, 5, 20, 50$.
   - Plot the trace plot of the burn in phase and the MCMC samples for the latent variable *home* using proposal distributions with different σ and t.
   - Estimate the rejection ratio for each parameter setting, report your results in a table.
   - Comment on the results. Which parameter setting worked the best for the algorithm?
   - Use the results from the optimal parameter setting:

(a) plot the posterior histogram of variable *home* from the MCMC samples.

(b) plot the estimated attacking strength $\mathbb{E}_{p(\boldsymbol{\theta},\boldsymbol{\eta}|\mathbf{y})}[att_i]$ against the estimated defending strength $\mathbb{E}_{p(\boldsymbol{\theta},\boldsymbol{\eta}|\mathbf{y})}[def_i]$ for each the team in one scatter plot. Please make sure to identify the team index of each point on your scatter plot using the index to team mapping in Table 1.

3 Variational Inference (VI) [30 pts]

3.1 A Review of Vanilla LDA [9 pts]

Latent Dirichlet Allocation (LDA) is a widely used model for extracting topics from text corpora. Given a collection of documents $\mathcal{D} = \{\boldsymbol{w}_d\}_{d=1}^{D}$, where $\boldsymbol{w}_d$ is the d^{th} document with words $\boldsymbol{w}_d = \{w_{di}\}_{i=1}^{N_d}$, LDA assumes each document is a mixture of K topics with mixture proportion $\boldsymbol{\theta}_d$. Each topic is a multinomial distribution parametrized by $\boldsymbol{\beta}_k$ over the vocabulary of V words. The generative process for LDA is as follows:

- For $k = 1 \rightarrow K$:
  - Draw topic $\boldsymbol{\beta}_k \sim \text{Dirichlet}(\boldsymbol{\eta})$.
- For each document $\boldsymbol{w}_d \in \mathcal{D}$:
  - Draw per-document mixture proportion $\boldsymbol{\theta}_d \sim \text{Dirichlet}(\boldsymbol{\alpha})$.
  - For each word $w_{di} \in \boldsymbol{w}_d$:
    - Sample a topic indicator $z_{di} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
    - Sample the word $w_{di} \sim \text{Multinomial}(\boldsymbol{\beta} z_{di})$

where $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are the parameters of the Dirichlet distributions for $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, respectively. The graphical model diagram of LDA is shown in Figure 2. Answer the following questions about LDA (no need to explain).

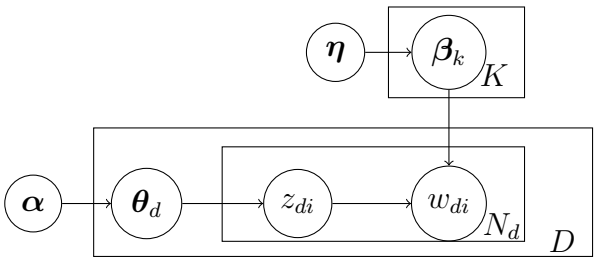

Figure 2: The graphical model diagram for LDA.

1. [**1 point**] True or False: Solving LDA involves posterior inference with non-conjugate distributions.

2. [**1 point**] True or False: Solving LDA using Variational EM involves non-convex optimization at E-step.

3. [**2 points**] Count the number of **model parameters** of the LDA using notations defined above.

4. [**2 points**] Write down the computational complexity of mean-field VI for LDA using Big O notation.

5. [**2 points**] Write down the memory complexity of mean-field VI for LDA using Big O notation.

6. [**1 point**] True or False: The vanilla LDA can be used to extract word embeddings.

3.2 More HMMs [21 pts]

Vanilla LDA assumes each word is independently generated given the topic vectors. To enhance this model by considering word order, we introduce a Hidden Markov Model (HMM). The HMM-LDA assumes a transition matrix $\boldsymbol{T}$ among K topics, denoted as $\boldsymbol{T}_{mn}$, representing the probability of topic m transitioning to topic n. With $\boldsymbol{T}$, we model sequential dependencies by letting word w_{di}, with probability $p(0 < p < 1)$, be sampled from topic z_{di}.

With probability $1 - p$, it transitions from the topic $z_{d(i-1)}$ of the previous word $w_{d(i-1)}$. The generative process is as follows:

- For each document $\boldsymbol{w}_d$:
  - Sample $p_d \sim \text{Beta}(\gamma)$.
  - Sample $\boldsymbol{\theta}_d \sim \text{Dirichlet}(\boldsymbol{\alpha})$.
  - For the first word (the head of HMM), sample $z_{d1} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$, $w_{d1} \sim \text{Multimomial}(\beta_{z_{d1}})$.
  - For each word w_{di} $(i > 1)$:
    - Sample $\delta_{di} \sim \text{Bernoulli}(p_d)$.
    - If $\delta_{di} = 1$: sample $z_{di} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$, then $w_{di} \sim \text{Multimomial}(\beta_{z_{di}})$.
    - If $\delta_{di} = 0$: sample $z_{di} \sim \text{Multinomial}(\boldsymbol{T}_{z_{d(i-1)},:})$, then $w_i \sim \text{Multimomial}(\beta_{z_{di}})$.

Where we use $\boldsymbol{T}_{m,:}$ to denote the m^{th} row of $\boldsymbol{T}$.
The posteriors of this model are still intractable, and we will use variational EM to perform posterior inference and estimate model parameters.

1. **[2 points]** Identify (a) the set of observed variables, (b) the set of hidden variables, (c) the set of model parameters to be estimated.

2. **[2 points]** Based on the altered generative process and Figure 2, draw the new graphical model for HMM-enhanced LDA (HMM-LDA).

3. **[1 points]** Write down the joint distribution defined by this generative process.

4. **[2 points]** Define variational distributions over hidden variables using proper distributions and variational parameters.

5. **[1 points]** Derive the Evidence Lower Bound (ELBO) using variational distributions.

6. **[3 points]** Derive the update rules for all variational parameters in the E step.

7. **[3 points]** Derive the update rules for all model parameters in the M step.

8. **[4 points]** Implement the derived variational EM algorithm for HMM-LDA using Python (≥ 3.5). Train the model on the provided Wikipedia corpus (details in `readme.txt`). Set the number of topics $K = 10$.

9. **[3 points]** Visualize each of the learned 10 topics by printing its top 10 words with highest probabilities, both in your script and write-up.

    - Tune hyper-parameters for correct convergence. Print ELBO values after each iteration in stdout during training. Report the iteration of convergence. Print ELBO after convergence in stdout.
    - Visualize the learned K topics after convergence in both stdout and your write-up.

4 QWOP [20 pts]

Description: In 2010, the QWOP game (`https://www.foddy.net/Athletics.html`) gained popularity for its challenging gameplay. Your task is to write a computer program to control the QWOP avatar, simulating its thigh and knee angles to maximize the distance it runs in the 100-meter event at the Olympic Games.

The QWOP physics engine has been implemented in Python and you can find it in the course web site(QWOP.py). The function takes a list of 40 floating-point numbers (corresponding to 20 instructions for the angle of the thighs and 20 for the angle of the knees) as input, producing the final x-position of the avatar's head. Your goal is to find an input vector that maximizes this output.

For this problem:

1. Ensure you can call the Python `sim(plan)` function with a plan of 40 floating-point numbers in the range [-1, 1]. Take advantage of the visualization to understand the avatar's movements.

2. (15 points) Optimize QWOP. Maximize the distance d your avatar travels. Design your own methods; avoid using online optimization code. Experiment with MCMCs, variants of gradient descent, etc.

3. (5 points) Describe the techniques you tried, their success, and report the best distance achieved. Provide the length 40 plan that yields this distance.

Deliverables: For part (b), submit your code. In part (c), discuss the best distance achieved, and provide the length 40 input plan for this distance.

Instructions for how to write your report and submission:

Hand in your solutions (including figures/plots) to all the questions in this document as one PDF file. Bibliography, simulation code, and additional material can be added as an appendix. The main pages should contain all important derivations and results, as well as design choices made in the implementation.
For each question only submit a single python script named `Qi.py` i= 1,2,3,4. Also create a `requirements.txt` where you put all your needed python requirements. Assuming the dataset file is placed on the same directory path with your script(don't need to add datasets to the zip file), make sure your script is executable (without bugs) under a clean python ($\geq$ 3.5) virtual environment with the following two commands:

```
pip install -r requirements.txt
python Qi.py
```