CS329H Autumn 2024 Assignment 2: Model-based Preference Optimization

Due: Monday, November 11th 11:59PM PST

There are 170 points total worth of questions on this assignment. **However, we will only grade out of 85 points.** As a result, grades of over 100% are possible and will be viewed as extra credit. You are free to pick the questions you would like to do. Point values of questions indicate relative difficulty.

All submissions should be made on GradeScope. A separate announcement on Ed will be made regarding when the GradeScope opens with the code autograder. **You may work in groups of up to** 4 **(including yourself).** As per the honor code, please indicate the SUNet IDs (e.g., `kenanhas`) of your groupmates below:

Collaborator SUNet IDs:

__________ __________ __________

As part of working in a group, you may discuss the questions with one another but **must submit your own written solutions and own code.** That is, **you may not use the same written or coding submission as any of your peers.** If you have any questions, please ask on Ed under the tag "Assignment 2." **Do not publicly share your written or coding solutions anywhere online (e.g., GitHub).** In addition, do not reveal your solutions on Ed either. If you are unsure whether your Ed post is too revealing, please post it privately.

Question 1: Uncertainty Quantification in Preference Learning (40 points)

In this question, we will explore Bayesian approaches to logistic regression in the context of preference learning using the Bradley-Terry model. We will compare different models and inference methods, including parametric linear models estimated using Metropolis-Hastings, parametric neural network models estimated using Hamiltonian Monte Carlo, and non-parametric models with Gaussian Processes. Finally, we will assess the uncertainty quantification in these models using the Expected Calibration Error (ECE).

Assume we have a dataset of pairwise preferences $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ represents the feature difference between two items (i.e., $x_i = e_1^{(i)} - e_2^{(i)}$ for embeddings $e_1^{(i)}$ and $e_2^{(i)}$), and $y_i \in \{0, 1\}$ indicates the preference ($y_i = 1$ if item 1 is preferred over item 2 in the i-th pair).

The likelihood of observing y_i given x_i and model parameters θ is given by the logistic function:

$$P(y_i = 1 | x_i, \theta) = \sigma(x_i^\top \theta) = \frac{1}{1 + e^{-x_i^\top \theta}}.$$

We will adopt a Bayesian approach by placing priors on the model parameters and using Markov Chain Monte Carlo (MCMC) methods to estimate the posterior distributions.

(a) **Uncertainty Quantification and Expected Calibration Error (11 points)**

(i) **(Written, 2 point)**. Spend some time reading `https://tinyurl.com/m77mk9c`. Explain what the Expected Calibration Error (ECE) measures and why it is important for assessing uncertainty quantification in probabilistic models.

(ii) **(Coding, 6 points)**. In `uncertainty_quantification/ece.py`, implement the ECE using the formula

$$\text{ECE} = \sum_{k=1}^{K} \frac{n_k}{N} \left| \text{acc}(B_k) - \text{conf}(B_k) \right|,$$

where n_k is the number of samples in bin B_k, N is the total number of samples, $\text{acc}(B_k)$ is the accuracy in bin B_k, and $\text{conf}(B_k)$ is the average confidence in bin B_k.

(iii) **(Written, 3 point)**. After doing parts (b), (c), and (d), compare the ECE scores and reliability diagrams of the 3 models. Which model(s) provide the best uncertainty quantification? Discuss possible reasons for the observed differences.

(b) **Parametric Linear Model Estimated Using Metropolis-Hastings (11 points)**

(i) **(Written, 3 points)**. Assume a prior on θ such that $\theta \sim \mathcal{N}(0, \sigma^2 I)$, where σ^2 is the variance and I is the identity matrix. Derive the expression for the posterior distribution $P(\theta|\mathcal{D})$ up to a normalization constant.

(ii) **(Coding, 6 points)**. Implement the Metropolis-Hastings algorithm to sample from the posterior distribution of θ in `uncertainty_quantification/metropolis.py`.

(iii) **(Written, 2 points)**. Discuss how you chose the proposal variance τ^2 and the number of iterations T and $T_{\text{burn-in}}$. How did these choices affect the convergence and mixing of your MCMC chain?

(c) **Parametric Neural Network Model Estimated Using Hamiltonian Monte Carlo (11 points)**

(i) **(Written, 2 points)**. Explain why Hamiltonian Monte Carlo (HMC) is suitable for sampling from the posterior distribution of neural network parameters compared to Metropolis-Hastings.

(ii) **(Coding, 7 points)**. Implement HMC to sample from the posterior distribution of the parameters θ of a neural network $f(x;\theta)$ used for preference prediction in `uncertainty_quantification/hmc_nn.py`. This will require a GPU and take around 5 minutes on it!

(iii) **(Written, 2 points)**. Briefly describe the performance of the HMC and Metropolis-Hastings models and provide the accuracy numbers.

(d) **Non-Parametric Model with Gaussian Process (GP) (7 points)**

(i) **(Written, 2 point)**. Describe how a Gaussian Process can be used for preference learning in this context (i.e., describe how the latent function is used for classification).

(ii) **(Coding, 2 points)**. Run the GP classification for preference learning code in `uncertainty_quantification/gaussian_process.py` and provide the accuracy numbers. This can only be run on a CPU and may take around 10 minutes to complete.

(iii) **(Written, 3 point)**. Discuss the computational complexity of the GP model compared to the parametric models. What are the advantages and disadvantages of using a GP in this setting?

Question 2: Active Learning for Preference Learning (40 points)

In this question, you will explore active learning strategies for preference learning using a linear model. We will use expected information gain as the acquisition function to select the most informative queries, where each query is a pair of items. Assume that we model the preferences using a simple linear model. Given feature vectors x_1 and x_2 corresponding to two items, the probability that x_1 is preferred over x_2 is modeled using a logistic regression model, i.e.,

$$P(x_1 \succ x_2|\theta) = \sigma(\theta^\top (x_1 - x_2)),$$

where $\theta \in \mathbb{R}^d$ is the model parameter vector, and $\sigma(z)$ is the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$. The goal is to sequentially select pairs of items to maximize the information gained about θ through preference queries.

(a) **Expected Information Gain (15 points)**

(i) **Derive the Expected Information Gain (Written, 3 points).** Suppose that after observing a preference between two items x_1 and x_2, the posterior distribution over θ is updated. The information gain from this observation is the reduction in uncertainty about θ measured using the Kullback-Leibler (KL) divergence between the prior and posterior distributions. Given the current posterior distribution $P(\theta|\mathcal{D})$ and a possible observation $y \in \{0, 1\}$ (where $y = 1$ if x_1 is preferred over x_2, and $y = 0$ otherwise), the expected information gain is:

$$\begin{aligned}\mathbb{E}[\mathrm{IG}(x_1, x_2)] = &P(y = 1|x_1, x_2, \theta) D_{\mathrm{KL}} \left(P(\theta|y = 1, \mathcal{D}) \parallel P(\theta|\mathcal{D})\right) \\ &+P(y = 0|x_1, x_2, \theta) D_{\mathrm{KL}} \left(P(\theta|y = 0, \mathcal{D}) \parallel P(\theta|\mathcal{D})\right)\end{aligned}$$

Derive this expression for the expected information gain of selecting the pair (x_1, x_2) for a preference query. Start by explaining how the KL divergence measures the information gain, and break down the expectation over the possible outcomes of the query.

(ii) **Simplifying the KL Divergence (Written, 4 points).** Assuming the prior and posterior distributions over θ are Gaussian (i.e., $P(\theta) \sim \mathcal{N}(\mu, \Sigma)$ and $P(\theta|\mathcal{D}) \sim \mathcal{N}(\mu', \Sigma')$), show that the KL divergence between the Gaussian posterior and prior simplifies to:

$$D_{\mathrm{KL}} \left(\mathcal{N}(\mu', \Sigma') \parallel \mathcal{N}(\mu, \Sigma)\right) = \frac{1}{2} \left(\mathrm{tr}(\Sigma^{-1}\Sigma') + (\mu' - \mu)^\top \Sigma^{-1} (\mu' - \mu) - d + \log \left(\frac{\det(\Sigma)}{\det(\Sigma')}\right)\right).$$

(iii) **Approximate Information Gain for a Linear Model (Written, 4 points).** In the case of a linear model with Gaussian priors on θ, assume that the posterior distribution $P(\theta|\mathcal{D}) \sim \mathcal{N}(\mu, \Sigma)$ is updated using Bayes' rule after each observation. The likelihood of observing a preference y is logistic, which does not conjugate with the Gaussian prior. However, for the purposes of this question, assume that after each query, the posterior mean μ' and covariance Σ' can be updated using an approximation method such as Laplace's approximation.

Using these assumptions, compute the expected information gain for a specific query (x_1, x_2) in closed form. You may express the information gain in terms of the updated mean μ' and covariance Σ' after observing the preference outcome.

(iv) **Laplace Approximation for Posterior (Written, 4 points).** The Laplace approximation for the posterior is given by

$$\begin{aligned}\mu' &= \arg\min_{\theta} -\log P(\theta|\mathcal{D}) \\ \Sigma'^{-1} &= \nabla_\theta \nabla_\theta - \log P(\theta|\mathcal{D})|_{\theta=\mu'}\end{aligned}$$

In our scenario with the Bradley-Terry model for likelihood, simplify $-\log P(\theta|\mathcal{D})$ and its Hessian ignoring the normalization constant.

(b) **Active Learning Algorithm (25 points)** In this section, you will implement an active learning algorithm for selecting the most informative queries using the expected information gain criterion.

(i) **(Coding, 4 points).** Implement `kl_divergence_gaussians` in `active_learning/main.py`.

(ii) **(Coding, 4 points).** Following your derived Laplace approximation, implement `negative_log_posterior`.

(iii) **(Coding, 4 points).** Implement `compute_hessian` that is used to obtain the inverse of the covariance matrix.

(iv) **(Coding, 3 points).** Implement `expected_information_gain`.

(v) **(Coding, 4 points).** Finally, implement `active_learning`.

(vi) **(Coding + Written, 6 points).** Plot the L^2 norm of the covariance matrix for each loop of the active learning loop. Additionally, on the same plot, implement a random baseline and plot its L^2 covariance matrix norm. The random baseline should randomly select a point in the dataset and not use any acquisition function. Interpret your plot and use it to compare the two methods.

Question 3: Linear Performance Metric Elicitation (30 points)

1. **(Written, 10 points).** For background on the problem setting, read `https://tinyurl.com/3b92sufm`. Suppose we have a linear performance metric given by

$$p(C) = 1 - \alpha(FP) - \beta(FN)$$

where C is a confusion matrix and FP, FN denote false positive and false negative rates. We wish to find the optimal classifier w.r.t. p. That is,

$$\phi^* = \arg\max_{\phi \in \Phi} p(C(\phi))$$

where Φ is the space of all probabilistic binary classifiers from $X \to [0, 1]$. Note that these classifiers return probabilities corresponding to the label 1. Show that ϕ^* is in fact deterministic and given by

$$\phi(x) = \begin{cases} 1 & \text{if } p(y|x) > f(\alpha, \beta) \\ 0 & \text{otherwise.} \end{cases}$$

for a threshold function f that you must find. (Hint: For a classifier ϕ, $FP = P(\phi = 1, y = 0)$ and $FN = P(\phi = 0, y = 1)$. Marginalize these joint probabilities over x and simplify.)

2. **(Written + Coding, 5 points).** Implement `classifier_metrics` in `lpme/main.py`. After doing so, run `plot_confusion_region` and attach the plot. What do you notice about the region of possible confusion matrices?

3. **(Coding, 15 points).** Implement `search_theta` in order to elicit the metric used by the oracle (which is parametrized by θ). Play around with the oracle's theta and run `start_search` to see how close you can approximate it!

Question 4: D-optimal Design with Logistic Model (30 points)

In this question, we explore D-optimal designs in the context of the Bradley-Terry model. The Bradley-Terry model is a logistic regression model used for paired comparison data. Given two items x_1 and x_2, the probability that item x_1 is preferred over x_2 is modeled as:

$$P(x_1 \succ x_2|\theta) = \frac{e^{\theta^\top x_1}}{e^{\theta^\top x_1} + e^{\theta^\top x_2}} = \frac{1}{1 + e^{\theta^\top (x_2 - x_1)}}$$

where $\theta \in \mathbb{R}^d$ represents the unknown model parameters, and $x_1, x_2 \in \mathbb{R}^d$ are the feature vectors associated with the two items. D-optimal design aims to maximize the determinant of the Fisher information matrix, thus minimizing the volume of the confidence ellipsoid for the estimated parameters. In this exercise, you will analyze D-optimal designs for this model.

(a) **Fisher Information Matrix for the Bradley-Terry Model (12 points)**

(i) **(Written, 6 points).** Derive the Fisher information matrix for the Bradley-Terry model at a design point (x_1, x_2). Show that the Fisher information matrix at a design point is:

$$I(x_1, x_2, \theta) = w(x_1, x_2, \theta)(x_1 - x_2)(x_1 - x_2)^\top,$$

where $w(x_1, x_2, \theta)$ is a weight function given by:

$$w(x_1, x_2, \theta) = \frac{e^{\theta^\top x_1} e^{\theta^\top x_2}}{\left(e^{\theta^\top x_1} + e^{\theta^\top x_2}\right)^2} = \sigma'(\theta^\top (x_1 - x_2)).$$

σ' is the derivative of the sigmoid function.

(ii) **(Coding, 6 points).** Implement `fisher_matrix` in `d_optimal/main.py` based on the derived expression.

(b) **D-optimal Design Criterion (18 points)**

(i) **(Coding, 11 points).** In the context of the Bradley-Terry model, a D-optimal design maximizes the determinant of the Fisher information matrix. Suppose we have a set of candidate items $\{x_1, \dots, x_n\}$, and we can choose N comparisons to make. Formally, the D-optimal design maximizes:

$$\det\left(\sum_{i=1}^{N} w(x_{i1}, x_{i2}, \theta)(x_{i1} - x_{i2})(x_{i1} - x_{i2})^\top\right),$$

where (x_{i1}, x_{i2}) denotes a pair of compared items in the design. Implement a greedy algorithm to approximate the D-optimal design. Given a set of n items and their feature vectors $\{x_1, \dots, x_n\}$, your task is to iteratively select the pair of items (x_{i1}, x_{i2}) that maximizes the determinant of the Fisher information matrix. Please implement `greedy_fisher`. Note that the setup in the code assumes we have a dataset of all possible differences between pairs of items as opposed to directly selecting the pairs.

(ii) **(Written + Coding, 7 points).** Notice that `posterior_inv_cov` uses a Laplace approximation for the posterior centered around the ground truth weights after labeling the chosen points. However, it turns out this approximation doesn't actually depend on the labels when taking the Hessian. Please run the file `d_optimal/main.py` and attach a plot of the norm of the covariance matrix of the posterior. What difference do you observe between greedy and random sampling? What is the win rate of greedy?

Question 5: Nonparametric Metric Elicitation (30 points)

In this question, we explore the problem of performance metric elicitation using a Gaussian Process (GP) to map the elements of the confusion matrix, specifically false positives (FP) and false negatives (FN), to an unknown performance metric. The goal is to learn a non-linear function that maps FP and FN to the metric, using relative preferences from pairwise classifier comparisons. We will use elliptical slice sampling for posterior inference.

(a) **Gaussian Process for Metric Elicitation (10 points)**

(i) **(Written, 2 points).** Assume that the performance metric $\phi(C)$ is a non-linear function of the confusion matrix C. For simplicity, assume that ϕ depends only on FP and FN, i.e.,

$$\phi(\mathrm{FP}, \mathrm{FN}) \sim \mathcal{GP}(0, k((\mathrm{FP}, \mathrm{FN}), (\mathrm{FP}', \mathrm{FN}'))),$$

where k is the covariance kernel function of the Gaussian Process. Explain why using a GP allows for flexible modeling of the metric ϕ as a non-linear function of FP and FN. What are the advantages of using a GP over a linear model in this context?

(ii) **(Written, 2 points).** Suppose we observe pairwise comparisons between classifiers, where a user provides feedback on which classifier they prefer based on the unknown metric ϕ. Given two classifiers with confusion matrices $C_1 = (\mathrm{FP}_1, \mathrm{FN}_1)$ and $C_2 = (\mathrm{FP}_2, \mathrm{FN}_2)$, the user indicates their relative preference. Let the observed preference be modeled by Bradley-Terry as:

$$\Pr(C_1 \succ C_2) = \sigma(\phi(\mathrm{FP}_1, \mathrm{FN}_1) - \phi(\mathrm{FP}_2, \mathrm{FN}_2)).$$

where we view ϕ as the reward function. How does this likelihood affect the posterior inference in the GP? Where does it introduce additional complexity?

(iii) **(Written + Coding, 6 points).** Given a set of observed pairwise comparisons, derive the posterior distribution over the latent function values ϕ given a set of confusion matrices preferences using Bayes' rule. Express the posterior distribution in terms of the GP prior and the pairwise likelihood function. You do not need to include the normalization constant. Implement the likelihood function in `loglik_from_preferences`.

(b) **Elliptical Slice Sampling for Posterior Inference (20 points)**

(i) **(Written, 3 points).** Read `https://proceedings.mlr.press/v9/murray10a/murray10a.pdf`. Elliptical slice sampling is a sampling method used to generate samples from the posterior distribution of a Gaussian Process. Explain the key idea behind elliptical slice sampling and why it is well-suited for sampling from the GP posterior in this context.

(ii) **(Coding, 10 points).** Implement elliptical slice sampling in `npme/elliptical_sampler.py` by following Figure 2 in the paper.

(iii) **(Written, 3 points).** Run the algorithm on a synthetic preference dataset of confusion matrices with pairwise preferences. The synthetic data will be constructed using the metric

$$\phi_{\text{true}}(\text{FP}, \text{FN}) = \log(1 + \text{FP}) + \log(1 + \text{FN}),$$

which captures the idea that the human oracle perceives both false positives and false negatives in a way that flattens out as these values increase (i.e., marginal increases in FP and FN have diminishing effects on the performance metric). Explain the psychological motivation behind this non-linear function. Why might a logarithmic form be appropriate for modeling human perception of classification errors?

Run the file `npme/main.py` and attach the plot of ϕ_{true} vs your elicited metric. What do you notice in the plot?

(iv) **(Written + Coding, 4 points).** Once the GP has been trained and posterior samples of the function $\phi(\text{FP}, \text{FN})$ have been obtained, how can we evaluate the quality of the elicited metric? Propose a method to evaluate how well the elicited metric ϕ aligns with the user's true preferences and implement it in `evaluate_elicited_metric` taking into the plot you saw in part (iii).