STAT 570 Probabilistic Machine Learning
Assignment 1

Problem 1: Monty Hall

We will apply Bayesian reasoning to the Monty Hall problem.

There are N doors, labelled 1 to N. Behind each door, there is a goat, except one where there is a car. You select a door, say i-th door, hoping to find the car behind it. The show host then opens all of the doors except for the i-th door and another door, j. Let H_n represent hypothesis that car is behind door n and Y denote the unopened door.

Q1 (2 points). Derive expression for posterior $P(H_n|Y = j)$ for $n = 1, ..., N$.

Q2 (1 point). Implement `monty_hall_posterior` function in file `problem1.py`. Return a numpy array of length N such that the n-th element stores $P(H_{n-1}|Y = j)$.

Problem 2: Monte Carlo integration

Q1 (2 points). (Naïve Monte Carlo) Estimate the value of π. Implement `monte_carlo_pi` in `problem2.py`.

Q2 (2 points). (Quasi Monte Carlo) We will again estimate π but use a specialized random number generator called Sobol sequence. Implement `quasi_monte_carlo_pi` in `problem2.py`.

Q3 (1 point). Briefly explain the idea behind QMC. How does QMC achieve faster rate of convergence?

Problem 3: Rao-Blackwellization

Let X be a random variable with density given by $f_x(x)$. The goal is to estimate expectation $I = \mathbb{E}_X[g(X)]$. The naïve Monte Carlo estimator is given by,

$$\hat{I} = \frac{1}{N} \sum_i g(x_i),$$

where $x_i \sim f_x$ denote i.i.d samples of X.

Now, suppose we have an auxiliary variable Y such that the joint density $f_{x,y}(x, y) = f_y(y) f_{x|y}(x|y)$ and the marginalization with respect to Y yields $f_x(x) = \int f_{x,y}(x, y)dy$. Then, we can formulate an alternative estimator for I:

$$\hat{I}^* = \frac{1}{N} \sum_i \mathbb{E}_{x|y}[g(X)|y_i],$$

where $\mathbb{E}_{x|y}[g(X)|y] = \int g(x) f_{x|y}(x|y)dx$ and $y_i \sim f_y(y)$.

Q1 (2 points). Show that $\mathbb{E}_Y[\hat{I}^*] = I$ (unbiased) and $var(\hat{I}^*) \leq var(\hat{I})$ (variance reduction).

If conditional expectation $\mathbb{E}_{x|y}[g(X)|y]$ can be computed in closed form, the Rao-Blackwellization yields variance reduction by replacing the problem of sampling from f_x to that of sampling from f_y.

Let (X, Y) follow a bivariate Gaussian distribution with parameters μ and Σ.

Q2 (1 points). (Tail probability estimation) Implement function `gaussian_tail` in `problem3.py` to estimate $P(X > x_0)$ using naive Monte Carlo.

Q3 (4 points). (Rao-Blackwellized estimator) Implement `gaussian_tail_rb` in `problem3.py` to estimate $P(X > x_0)$ using Rao-Blackwellization.

Q4 (3 points). Generate a plot to compare the variance of the estimators from Q2 and Q3 for $1e5$ Monte Carlo samples. Clarify the distinction between variance and the convergence rate.

Problem 4: Optimization

Let $Y_{ij} \sim \text{Multinomial}(\pi_i)$ where π_i is given by the softmax regression,

$$\pi_{ij} = \frac{\exp(x_i^T \beta_j)}{\sum_{j'} \exp(x_i^T \beta_{j'})}.$$

- Define a multinomial regression model using PyTorch's `torch.nn`.Module.
- Use autograd to compute gradients for optimization.
- Optimize the model parameters.

Q1 (3 points). Currently, `multinomial_regression.py` implements the log likelihood function. We will incorporate Normal prior on β_j and optimize the posterior.

1. First, implement `log_prior` in `problem4.py`.
2. Write a new function `log_posterior` in `problem4.py` that combines log likelihood and log prior.
3. Finally, define and implement `optimize` function, taking in number of iterations as argument and optimizes the parameters of Multinomial regression model. Refer to `MultinomialRegression.ipynb`.

Q2 (2 points). Implement `std_err` function in `problem4.py`. Use autograd functionality of PyTorch to obtain the Hessian and hence, the standard error estimates for each β_j's. Report the results using a figure and state which variables were selected as significant.