Brief Report (Part 1) on Dynamic Causal Bayesian Optimization and Optimal Intervention

Name: Lau, Ying Yee Ava

A. Literature Review

Causal inference is a critical aspect of understanding the cause-and-effect relationships between variables, distinguishing itself from mere correlation analysis. This pursuit is paramount in fields such as epidemiology, economics, and artificial intelligence, where the elucidation of causal mechanisms can significantly enhance decision-making processes. However, the challenge lies in accurately learning these causal relationships and determining optimal interventions within a causal framework. This challenge necessitates advanced optimization techniques capable of navigating complex causal structures.

Bayesian Optimization (BO) has emerged as a powerful tool for addressing global optimization problems characterized by expensive, unknown objective functions. By employing surrogate models, typically Gaussian Processes (GPs), BO provides a probabilistic framework that models the latent objective function, facilitating the exploration-exploitation trade-off. Incorporating causal inference into BO frameworks extends their utility by enabling the optimization of interventions within causal models. This integration allows BO to not only seek optimal solutions but also understand the causal pathways leading to these solutions.

Adaptive Bayesian Optimization (ABO) extends the traditional BO framework by capturing the time dependency of the objective function. However, ABO does not consider the causal structure among inputs nor their temporal evolution. It treats inputs as fixed, thus breaking causal dependencies and limiting its ability to model dynamic causal relationships.

Causal Bayesian Optimization (CBO) focuses on optimizing interventions within a causal graph, specifically in Directed Acyclic Graphs (DAGs). CBO models intervention functions using GPs, integrating causal inference by considering the causal relationships among variables. This approach allows for interventions on a subset of variables, exploiting the propagation of effects in the causal graph to achieve cost-effective and optimal solutions. However, CBO does not account for temporal evolution, which limits its ability to find time-indexed optimal actions.

Dynamic Causal Bayesian Optimization (DCBO) improves upon existing methods (CBO, ABO, BO) by accounting for both the causal relationships among input variables and the causality between inputs and outputs, which may evolve over time. This capability prevents suboptimal interventions and allows for interventions on continuous variables where the target variable has a non-stationary interventional distribution.

In examining temporal causal relationships across different decision-making frameworks, including Multi-Armed Bandits (MABs) and Reinforcement Learning (RL), distinct approaches are observed. Causal MABs focus on identifying the appropriate set of interventions that align with rewards, without specifying the intervention value. Conversely, RL tackles dynamic environments using Markov Decision Processes (MDPs), which require well-defined state dynamics. DCBO distinguishes itself from these approaches by not depending on an MDP framework; it instead focuses on time-indexed actions without incorporating state transitions.

B. Why this choice

Propose your choice of methods and the reason for the choice

- Machine learning techniques have evolved from traditional statistical methods, meaning they often identify associations between objects rather than causation. There is a growing demand for AI systems to possess a more logical comprehension of the world, which would make their predictions more explainable and, therefore, more trustworthy. As a result, I anticipate that causal inference will become a widely discussed and popular topic in the future.
- Causal inference is a challenging subject and is entirely new to me. I would like to take this interview as an opportunity, as working on a project is one of the quickest ways to learn about a completely new topic.

C. Questions

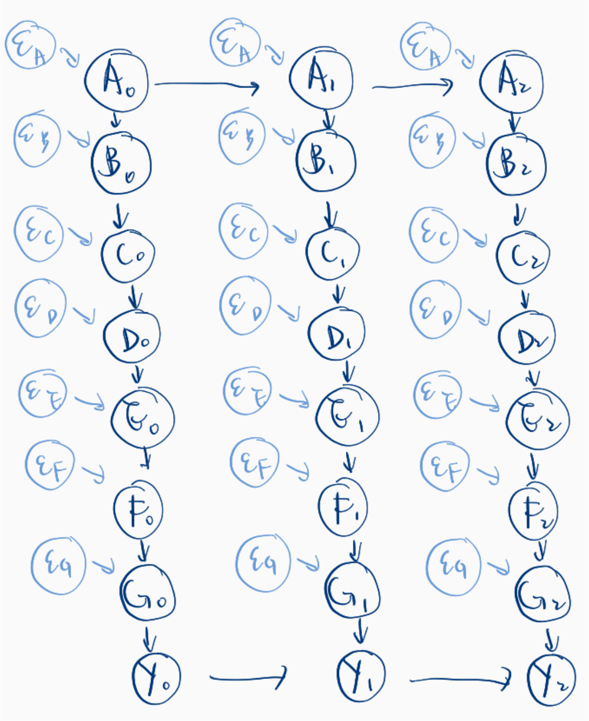
C1. Q1

In the paper, the authors considered very small and simple graphs, which might not be the case in practice.

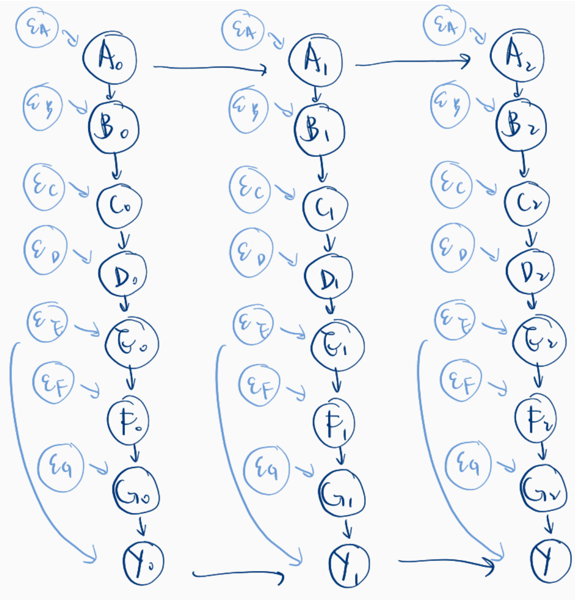
C1.a. Can you give an example of a causal graph with 15 nodes at each time step – 7 non-manipulable, 7 manipulable, and 1 target variable, how would you get the exploration set (a key input to the algorithm)?

1. The exploration set depends on the causal graph’s structure. If I need to determine the exploration set manually, using fundamental knowledge of structural causal graphs and do-calculus, it would vary based on the graph’s structure. In this context, node Y is the target variable. Dark blue nodes represent manipulable variables, while light blue nodes are non-manipulable. For instance, the common cause ε_E is non-manipulable.

    - In the example below, to investigate the causal effect of B or its descendants on Y, we might consider intervening on A. However, given that some variables cannot be manipulated, intervening on B could be advantageous for examining the causal effects of its descendants on the target variable, particularly if the non-manipulable variable has a substantial influence. Therefore, the exploration set is {{A},{B},{C},{D},{E},{F},{G}}.



- In the next example, if we aim to study the causal effect of E on Y, we cannot intervene on ε_E as it is only observable. From do-calculus knowledge, we can perform a frontdoor adjustment by intervening on E, F, and G. For the causal effect of B,C or D on Y, we can intervene on {A, E, F, G}. Other non-manipulable variables might introduce noise to their children's nodes, so intervention could be necessary. One potential exploration set is {{E,F,G},{F,G}, {G},{A,E,F,G}, {B,E,F,G},{C,E,F,G},{D,E,F,G}}.



2. We can also rely on the variable of interest and domain knowledge. For example, in the DCBO paper there is a ODE experiment with 7 variables. They did not select the exploration set through the graph structure. Since they wanted to control the effect of mortality concentration in the chemostat, they chose to intervene on Nitrogen concentration, Predator juvenile concentration, and Predator adult concentration.

3. Consider conducting a literature review on methods for identifying optimal exploration sets in causal graphs with high dimensionality. It is possible that there are state-of-the-art algorithms available that can effectively address this challenge.

C1.b. Would you write a program for this purpose?

Yes, it is worthwhile to develop a program. Relying solely on domain knowledge or using small graphs is not always feasible. Manually finding the exploration set can be time-consuming and prone to errors, such as overlooking or over-identifying backdoor paths.

C1.c. Is it enough to have simply the causal diagram to get the exploration set? What additional specifications do you need ?

I believe if we have a clear domain of interest or a treatment for measurement and the model is small enough, it might be possible to determine the exploration set. However, considering all temporal and causal dependencies from an exploratory perspective in medium-sized graphs, there are numerous combinations of backdoor paths to account for. In such cases, it is essential to incorporate domain knowledge regarding the subject of the graph.

C2. Replicate their synthetic experiment results in Tensorflow probability.

https://github.com/neildhir/DCBO. Hint: Check how they used the GPy package and think about how we can use TFP to replicate it.

- Gaussian Process Regression They mainly used the GPy package for RBF Kernels and Gaussian Process Regression.

  ```
  def fit_gp(x, y, lengthscale=1.0, variance=1.0, noise_var=1.0, ard=False,
  n_restart=10, seed: int = 0,) -> Callable:
      np.random.seed(seed)
      kernel = RBF(x.shape[1], ARD=ard, lengthscale=lengthscale, variance=variance)
      model = GPRegression(X=x, Y=y, kernel=kernel, noise_var=noise_var)
      model.optimize_restarts(n_restart, verbose=False, robust=True)
      return model
  ```

  In `TFP`, there is a built-in GP module that can be used for this purpose. The `RBF` kernel is equivalent to the `ExponentiatedQuadratic` kernel in TFP. Unlike `GPy`, the optimization process in `TFP` is configurable.

  ```
    ...
    kernel = psd_kernels.ExponentiatedQuadratic(amplitude, length_scale,
  feature_ndims=feature_ndims)
    ...
    self.gp = tfd.GaussianProcess(
        kernel=kernel,
        index_points=x,
        observation_noise_variance=observation_noise_variance)

    optimizer = tf.optimizers.Adam(learning_rate=.05, beta_1=.5, beta_2=.99)

    @tf.function
    def optimize():
        with tf.GradientTape() as tape:
            loss = -self.gp.log_prob(y,)
        grads = tape.gradient(loss, self.gp.trainable_variables)
        optimizer.apply_gradients(zip(grads, self.gp.trainable_variables))
        return loss

    for i in range(n_restart):
            neg_log_likelihood_ = optimize()
  ```

  If we would like to sample from the GP using a different set of points, we can wrap the GP in a `GaussianProcessRegressionModel` and sample from it.

  ```
  def predict(self, x):
        # import ipdb; ipdb.set_trace()
        return tfd.GaussianProcessRegressionModel(
            kernel=self.gp.kernel,
            index_points=x,
            observation_index_points=self.X,
            observations=self.Y,
            observation_noise_variance=self.gp.observation_noise_variance
        ).sample()
  ```

- Kernel Density Estimation In the official repository, they used the `scipy` package for Kernel Density Estimation.

  ```
  from sklearn.neighbors import KernelDensity
  ```

  However, I believe that there is no direct equivalent in `TFP`.

- I am still working on the replication of the DCBO algorithm.

C3. Q3 (Part 1)

C3.a. What are the key ideas of this paper? What are the ideas that excite you the most? Why do you find them interesting and critical?

The paper presents several key ideas in the theorems and propositions, which are crucial for the DCBO algorithm:

1. Dynamic Causal Global Optimization (DCGO): This extends Bayesian Optimization by incorporating temporal causal dynamics among variables. At each time step t, interventions build upon those from the preceding time step $t - 1$. Interventions are not performed on time steps earlier than $t - 2$. The optimization goal is to identify the optimal intervention set and levels that minimize the expectation of the target variable.

2. Assumption 1:
   - The causal structure remains invariant within each time slice.
   - The parents of variables are divided into two groups: those parents having the same timestamp as target node (emission) and those which are one timestamp behind (transition).The SEMs are estimated using GP Regression. The GP functions for the two parent groups can be added together to reflect their causal effect on the variable.
   - No unobserved confounders.

3. Theorem 1 (Time Operator): This theorem states that the transition function of the target variable can directly utilize the optimal intervention sets found in the previous timestamps.

4. Corollary 1: It highlights the need to update the values of the children of intervened nodes in the causal graph during interventions.

5. Proposition 3.1: If the causal structure remains invariant, the exploration set can remain consistent across time steps.

The most intriguing concept to me is the time operator. In temporal systems, the number of causal variable relations grow over time. The time operator allows the intervention at t to directly on conditioned on the optimal intervention set at $t - 1$. This is crucial as the exploration set does not expand over time. This simplifies the optimization problem and reduces computational complexity.

C3.b. Do you think the acquisition function is correct? Is there any typo? If yes, can you explain? If not, can you derive the correct one?

The definition presented in the paper for the acquisition function is $\mathrm{EI}_{s,t}(\mathbf{x}) = \mathbb{E}_{p_{(y_{s,t})}}\left[\max\left(y_{s,t} - y_t^*, 0\right)\right] / \operatorname{cost}\left(\mathbf{X}_{s,t}, \mathbf{x}_{x,t}\right)$. While the overall concept of the acquisition function is sound, there are two minor concerns:

1. The paper does not explicitly define $p\left(y_{s,t}\right)$. I think it should reference to $p\left(y_{s,t} \mid f_{s,t}, \sigma^2\right)$ mentioned in the "Likelihood" subsection, since the mean function and the variance is provided to the bayesian model by the causal prior.
2. As noted in a later paragraph of the "Acquisition Function" subsection, the parameter ξ is employed to adjust the system's level of exploration, which corresponds to the `jitter` parameter of the `class CausalExpectedImprovement` in the code.

Thus, a more precise formulation of the acquisition function would be: $\mathrm{EI}_{s,t}(\mathbf{x}; \xi) = \mathbb{E}_{p(y_{s,t} \mid f_{s,t}, \sigma^2)}\left[\max\left(y_{s,t} - (y_t^* + \xi), 0\right)\right] / \operatorname{cost}\left(\mathbf{X}_{s,t}, \mathbf{x}_{x,t}\right)$.

C3.c. Do you find any errors or questionable issues?

- I think there is a potential issue in the code. In `dag_utils/adjacency_matrix_utils.py`, the function `get_emit_and_trans_adjacency_mats` generates emission and transition matrices from adjacency matrices. When repeated edges exist in the graph, the resulting matrix elements can exceed 1. However, this situation is not addressed in the `fit_arcs` function within `sem_utils/sem_estimate.py`; the program may mistakenly interpret repeated edges as fork nodes. Since causal graphs should not contain repeated edges, it is essential to enforce this constraint in the code.

C3.d. Are there any parts that are not clear in the paper?

- In definition 4, the terms Y_t^{PT} and $\mathbf{X}_{s,t}^{\mathrm{PY}}$ appear, but their interpretations are ambiguous. I interpret "PT" as referring to the parent of the target variable, while "PY" seems to denote the parent of variable Y. Thereore, I think "PT" and "PY" carry the same meaning.
- The paper does not provide a definition for $\mathcal{D}^I$ iat the beginning, but it is introduced in the experiments section of CBO. There, it is defined as $\mathcal{D}^I = \{(\mathbf{x}_s^i, \mathbb{E}[Y|\,\mathrm{do}(\mathbf{X}_s^i = \mathbf{x}_s^i]))\}_{i=1,s=1}^{P,\,|\mathbf{ES}|}$.
- In Corollary 1, the term "PW" is not defined; I assume it refers to the parents of W.
- Also in Corollary 1, although the function $C(\cdot)$ is mentioned in the main text, its definition as a function that returns child nodes is not explicitly stated. I inferred this meaning from Appendix B, where it is noted that $\{f_W(\cdot)\}_{W\in\mathbf{W}}$ is recursive.

C3.e. Can you please include derivations that are not clear in their paper?

Still looking into the proof at this stage.

C3.f. Is there any important part of their code that is not mentioned in the paper?

- The intervention grid for generating the intervened samples has a limit in size. In `utils/sequential_intervention_functions.py`, if the number of nodes in the `exploration_set` is less than or equal to 2, the value of `size_intervention_grid` is unrestricted. However, if the number of nodes exceeds 2, `size_intervention_grid` is capped at 10 when the input size is set greater than or equal to 100.
- The concept of transition and emission matrices is introduced in Assumption 1, but its term is not explicitly introduced in the paper.
- Different BO models are utilized at different time steps. When initializing the BO model, there is an option to transfer variances from the previous time step to the current one. Alternatively, the user can choose to use a default initialization value of 1.0 for the variance.
- In the "Prior Surrogate Model" section, it is mentioned that kernels are created for source nodes, while "other nodes" are represented as GPs. In the code, "other node" is further divided into fork, chain and collider, in which there are differences in handling the GP regression model. For example, the GP function for colliders (n-to-1) is $f : \mathbb{R}^n \to \mathbb{R}$, while for forks (1-to-n), it is n initializations of $f : \mathbb{R} \to \mathbb{R}$.

C3.g. Can you explain the paper based on details you have learnt from their code and your result replication experience?

The following is the detailed explanation of the DCBO algorithm based on the code and the replication experience:

1. Observation Dataset and SEMs: Observation dataset samples are generated from the true / pre-defined SEMs.
2. Causal Priors Generation:
   - Analyze the SCM by separating its adjacency matrix into transition and emission matrices.

    - Generate two sets of priors for transition and emission, fitting KDEs for source nodes, and GP Regression for colliders and other nodes.
    - These priors form an estimated SEM, providing static and dynamic SEMs that encapsulate relationships in the SCMs.
    - Create two copies of SEMs for mean and variance, which serve as causal priors.

3. Bayesian Optimization Setup:
    - Before entering the optimization loop, pass the mean and variance functions of causal priors into the acquisition function to calculate Expected Improvement (EI).
    - For every exploration set, create an intervention input data utilizing the acquisition function. The output data is generated by sampling with the true SEMs. This output dat can be sampled multiple times for the same input data, taking into account varying noise levels.
    - For each exploration set, create a BO model at each time step using the mean and variance functions of causal priors.
        - Define a GP Regression model with a mean function from the prior mean and a causal RBF kernel using the prior variance function, allowing hyperparameters to be inferred from the previous time step.

4. Bayesian Optimization Loop:
    - With at least one sample in the intervention dataset, repeat the BO loop for several iterations.
    - For each exploration set, calculate EI using the acquisition function, with mean and variance predicted by the BO model.
    - The acquisition function samples the next intervention input, and the true SEM provides the intervention value (output).
    - Update the BO model with the intervention dataset
    - Update the current best intervention level and value.

5. Outcome: Return the optimal intervention sets and levels for each time step.

C3.h. Can you think of a possible application of such techniques for a bank?

This method may be useful for banks that want to understand the impact of regulatory changes on outcomes like bank stability, risk-taking behavior, or market liquidity. For example, a bank can compare the changes in outcomes over time between a treatment group (e.g., banks affected by a new regulation) and a control group (e.g., banks not affected by the regulation). This method helps learn the causal effect of the regulation by finding optimal interventions on trends that would occur regardless of the regulatory change.

D. Testing plans

D1. Unit Testing

1. Unit Testing of Individual Components

    - Objective: Verify the correctness of individual components and functions in isolation. Ensure that each member function of an object is tested thoroughly.

    - Example Individual Components to Test:
        - Graphs, SEMs, Acquisition Functions, Causal Priors, Gaussian Process (GP) Regression Functions.

    - Testing the above classes in different scenarios:
        - Graphs with different connection types (many-to-1, 1-to-many, many-to-many) and node configurations (source nodes, varying time steps, and node counts).
        - Node and edges information handling in complex graphs

    - High dimensional exploration sets
    - Cover all possible edge cases and invalid inputs for each component.

2. Unit Testing of Main Component
    - Objective: Validate the interaction between components and the overall functionality of the system.
    - Example Main Components to Test:
        - Main flow of Bayesian Optimization
        - DCBO Algorithm
    - Testing content:
        - Setting different configurations and hyperparameters. Ensure that the system works correctly with all varying settings.
        - Evaluate performance and correctness with different datasets and parameter settings.

D2. Consistency Testing with Fix Seeds

- Objective: Ensure reproducibility and consistency between the official repository and your replication.
- Approach:
    - Fix random seeds for all stochastic processes in both the official and replicated codebases.
    - Compare outputs of GP models. Establish a tolerance level for discrepancies due to inherent stochasticity.
    - Use provided synthetic datasets to compare results and evaluation metrics, ensuring similar outcomes.

D3. Code Readability and Maintainability Testing

- Objective: Assess and improve the readability and maintainability of the code. (Essential for future development and collaboration in corporate environments)
- Code Analysis Tools:
    - `pyre`: Perform Python type checking to ensure type safety and clarity.
    - `radon`: Analyze code complexity and maintainability metrics, identifying areas for improvement.

D4. Documentation

- Maintain detailed records of test cases, expected outcomes, and actual results.