ProxyTune: Hyperparameter tuning through iteratively refined proxies

Agrin Hilmkil [*1] **Wenbo Gong** [*1] **Nick Pawlowski** [1] **Cheng Zhang** [1]

Abstract

Tuning the hyperparameters of machine learning algorithms against a target metric is an essential way of ensuring good performance on tasks. However, in areas such as causal machine learning the target metric may not be accessible due to the lack of ground truths. In this work, we compare two existing approaches and propose an extension, which iteratively refines proxies towards the dataset, called *ProxyTune*. This allows constructing previously unavailable metrics through proxies, which enables the existing hyperparameter tuning methods. We focus on the causal discovery, where the ground truth graph is unavailable. Our preliminary results on synthetic data show the ineffectiveness of existing approaches and the advantages of the iterative refinement.

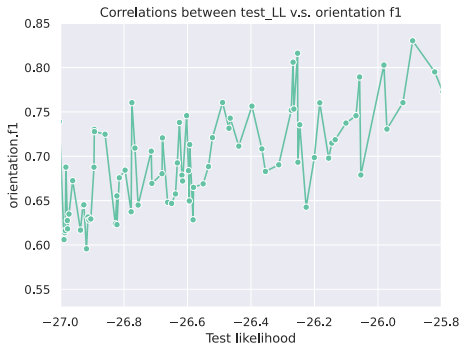

Figure 1. Correlation plot between test LL and F1 for a single dataset. These plots are generated by randomly selecting 2000 from the total 10000 runs and sort them in a descending order based on test LL. While the PM is correlated with the target metric (F1), it is very noisy and can lead to selecting a suboptimal model.

1. Introduction

Hyperparameter Tuning (HT) in causal machine learning (Rubin, 2005; Parikh et al., 2022; Peters et al., 2017; Zheng et al., 2018) is hard due to the lack of ground-truth target quantities, such as the true structures or average treatment effects. Therefore, it is impossible to calculate the target metric for different hyperparameters complicating the comparison of their respective performances, rendering the use of classical HT methods such as Bayesian Optimization (BO) (Frazier, 2018) as incompatible choice.

Despite the above difficulty, a common approach is to use an alternative available metric that indirectly reflects the quality of the target metric, called proxy metrics. HT is then applied to maximize the proxy metric. Depending on the tasks, different proxy metrics are preferred. Under the scope of this investigation, we limit ourselves to causal discovery, where we want to select model hyperparameters for better graph metrics, specifically the F1 score of edge orientation (F1) (Geffner et al., 2022; Annadani et al., 2024). One common proxy metric for such task (Zheng et al., 2018; Geffner et al., 2022; Lachapelle et al., 2019) is Log Likelihood (LL) of the test dataset, which indicates how well the model fits the data. In this paper, we will use ProxyMetric (PM) to refer to the HT method that optimizes the proxy metric. Unfortunately, from our empirical investigation (Figure 1 and Section 4.2), LL does not serve as a perfect indicator for F1. Alternative to PM, Parikh et al. (2022) recently proposed a causal validation method, called *Credence*, where they leveraged a deep generative model to create a synthetic dataset, called proxy dataset, such that its corresponding target metric can be evaluated. Then, this target metric of proxy dataset can be used as the proxy metric for model validation or HT.

However, *Credence* generates a single synthetic dataset, the quality of which can also be affected by the hyperparameters of the generative model. To mitigate this, we propose an iterative refinement strategy, called *ProxyTune*, compared to the 1-step *Credence*. *ProxyTune* uses previously selected hyperparameters as the initialization of the next step, and adds an acceptance stage to avoid deterioration of generation quality. In this preliminary investigation, we compare the above three approaches on the causal discovery task with synthetic dataset. Due to the generality of these frameworks, they can be applied to other target metrics as well, with the only requirements being: (1) one can sample a dataset from the generative model with access to the target metric of the proxy dataset; and (2) one can evaluate the LL of the model given a data set. For this paper, we choose DECI (Geffner et al., 2022) as our candidate generative model.

We summarize our empirical findings as the following:

- The proxy metric LL alone does not provide a useful signal for graph metrics like F1, and *Credence* typically selects worse hyperparameters compared to PM.
- *ProxyTune* demonstrates improvements over the previous two approaches, indicating the potential advantages of iterative refinements.
- LL is a good indicator for F1 together with *ProxyTune*, despite its ineffectiveness if applied alone.

2. Related Work

HT serves an important step towards improving the performance, especially when one cares about the metrics that cannot be directly evaluated in practice, such as most tasks related to causal modeling and structure learning. Machlanski et al. (2023) run extensive empirical trials using semi-synthetic datasets and draws the conclusion that careful HT and proxy metric selection have a larger effect than choosing any particular model. Furthermore, they show that the available proxy metrics do not robustly reflect the true target metric across datasets. To resolve the above issue, there are some existing research designed specifically for a given problem. Alaa & Van Der Schaar (2019) develop an evaluation method that uses an additional *plug-in* estimator to approximate a specific target metric called conditional average treatment effect (CATE). Saito & Yasui (2020) similarly use a plug-in estimator to rank the causal inference models. Biza et al. (2020) on the other hand, propose scoring structure learning methods through the average predictive performance of regressors on each variable given its Markov blanket. We note that this is similar to the predictive performance metric like LL. The work most similar to *ProxyTune* is *Credence* (Parikh et al., 2022), which evaluates the performance of causal inference methods using synthetic data generated from a separate deep generative model fitted to the target dataset. However, its original formulation is designed specifically for causal inference tasks without iterative refinement. On the other hand, *ProxyTune*, by design, can be applied to both many tasks through an iterative refinement manner without external generative models.

3. ProxyTune

3.1. Problem setup

The goal of causal discovery is to infer the underlying causal structures between variables from the observational dataset $\mathcal{D}$. Specifically, we investigated different approaches of HT in terms of finding good hyperparameters η and its corresponding model $\mathcal{M}$ to maximize the target graph metric μ, such as F1.

3.2. Candidate 1: ProxyMetric

The PM procedure can be described as: (1) for an initialized hyperparameter $\widetilde{\eta}_0$, we train a model $\widetilde{\mathcal{M}}_0$ on $\mathcal{D}$; (2) propose another $\widetilde{\eta}_i$. We repeat the above steps, until a total budget N is consumed. Then, we evaluate all $\{\widetilde{\mathcal{M}}_i\}_{i=0}^{N}$ on the proxy metric (e.g. test LL) and pick the best one. We use random sampling to propose the new candidate hyperparameter to avoid the effects from other factors, but it can easily be generalized to use BO.

3.3. Candidate 2: Credence

Instead of directly optimizing the proxy metric, *Credence* incorporates a generative model to produce a proxy dataset with access to its target metric, which mimics the true data $\mathcal{D}$. Specifically, the procedures are: (1) apply method PM to pick the best model π^0 based on test LL with half the budget $\frac{N}{2}$; (2) generate a proxy dataset $\widetilde{\mathcal{D}}^1$ from π^0, note that now we have the access to the graph underlying $\widetilde{\mathcal{D}}^1$; (3) perform a hyperparameter search with budget $\frac{N}{2}$ but under proxy data $\widetilde{\mathcal{D}}^1$ and target metric μ of its graph.

3.4. Candidate 3: ProxyTune

Our proposed method *ProxyTune*, can be seen as a generalization of *Credence* with iterative refinement. The intuition is that when the proxy dataset is close to the ground truth, the hyperparameters selected by maximizing the target metric of proxy are transferable to the ground truth data.

ProxyTune is illustrated in Algorithm 1. It starts from initial hyperparameters η^0 from a distribution H and a trained model π^0 with ground truth data $\mathcal{D}$. For each *ProxyTune* iteration $t = 1, \ldots, T$, we will first create a proxy dataset $\widetilde{\mathcal{D}}^t$ using the previously accepted model π^{t-1}. Then, a search procedure will be performed to find a set of K candidate hyperparameters $\{\widetilde{\eta}_i^t\}_{i=1}^{K}$ based on ranking the target metric on proxy data $\widetilde{\mathcal{D}}^t$ (refer to Section 3.4.3). Then, it is followed by an acceptance test to select one hyperparameter among $\{\widetilde{\eta}_i^t\}_{i=1}^{K}$ and its corresponding trained model π^t with true data $\mathcal{D}$ (see Section 3.4.2). If no one is selected, it will return to the previously accepted model $\pi^t \leftarrow \pi^{t-1}$. After finishing all iterations, the final model selected among all the accepted ones will be chosen based on the test likelihood l of true test data $\mathcal{D}_{\text{test}}$. Since the proxy data are generated from a proxy, previously unavailable metrics (e.g. F1) can be computed and provide useful signals for search hyperparameters.

Algorithm 1 ProxyTune

function INIT_HPARAMS($\mathcal{D}, \mathcal{D}_{\text{test}}, C,$ H) ▷ Initialize hyperparameters with dataset $\mathcal{D}$, $\mathcal{D}_{\text{test}}$, total computation budget C and hyperparameter distribution H. See Section 3.4.1
function FIT(η, $\mathcal{D}$) ▷ Fit a model with hyperparameters η on dataset $\mathcal{D}$.
function TEST(π, $\mathcal{D}_{\text{test}}$) ▷ Evaluate the test likelihood with proxy π
function ACCEPT($\mathcal{D}$, $\mathcal{D}_{\text{test}}$, $\{\widetilde{\eta}_i^t\}_{i=1}^K$, π^{t-1}) ▷ Acceptance procedure, see Section 3.4.2.
function SAMPLE(π) ▷ Sample dataset from proxy π, see Section 3.4.4
function SCORE($\mathcal{D}$, π) ▷ Score the proxy π on dataset $\mathcal{D}$, e.g. F1.
function SEARCH($\widetilde{\mathcal{D}}$, K, H, C) ▷ Tuning steps with SCORE, K candidates and hyperparameter distribution H under computation budget C. See Section 3.4.3

procedure *ProxyTune*($\mathcal{D}$, $\mathcal{D}_{\text{test}}$, T, H, $C \times (T+1)$) ▷ Target train and test dataset $\mathcal{D}$, $\mathcal{D}_{\text{test}}$, number of *ProxyTune* iterations T, hyperparameter distribution H and total budget $(T+1) \times C$.
  $\eta^0 \leftarrow$ INIT_HPARAMS($\mathcal{D}$, $\mathcal{D}_{\text{test}}$, C, H) ▷ Initialize hyperparameters
  $\pi^0 \leftarrow$ FIT(η^0, $\mathcal{D}$) ▷ Fit the initial model

  for $t \leftarrow 1, \cdots, T$ **do**
    $\widetilde{\mathcal{D}}^t \leftarrow$ SAMPLE(π^{t-1}) ▷ Sample proxy dataset from previously accepted proxy
    $\{\widetilde{\eta}_i^t\}_{i=1}^K \leftarrow$ SEARCH($\widetilde{\mathcal{D}}^t$, K, H, C) ▷ Search for candidate hyperparameter sets.
    if t=1 **then** ▷ Always accept at first iteration
      $\pi^t \leftarrow$ FIT($\mathcal{D}, \widetilde{\eta}_1^t$)
      $\ell_t \leftarrow$ TEST($\hat{\pi}_i, \mathcal{D}_{\text{test}}$) ▷ Evaluate the test LL
    else
      $\pi^t, \ell^t \leftarrow$ ACCEPT($\mathcal{D}$, $\mathcal{D}_{\text{test}}$, $\{\widetilde{\eta}_i^t\}_{i=1}^K$, π^{t-1}) ▷ Accept one and return test likelihood
  return π^i where $i = \arg\max_j \ell^j$ ▷ Final model selection based on maximizing test likelihood

3.4.1. INITIALIZATION

Algorithm 2 Initialize hyperparameters

procedure INIT_HPARAMS($\mathcal{D}$, $\mathcal{D}_{\text{test}}$, C, H)
  for $i = 1, \ldots, C$ **do**
    $\widetilde{\eta}_i \sim H$ ▷ Sample a hyperparameter from H
    $\hat{\pi}_i \leftarrow$ FIT($\mathcal{D}$, $\widetilde{\eta}_i$)
    $\ell_i \leftarrow$ TEST($\hat{\pi}_i, \mathcal{D}_{\text{test}}$) ▷ Evaluate the test likelihood
  return $\hat{\pi}_j$ with $j = \arg\max_j \ell_j$ ▷ Return the model with maximized likelihood

The purpose of the hyperparameter intialization, denoted INIT_HPARAMS in Algorithm 1 is to provide a good starting point for the following *ProxyTune* iteration. There are several possible choices. For example, one can manually set the hyperparameters or start with a hyperparameter search using a manual proxy model when a prior model or knowledge exists for the problem. In this paper, however, we perform the initialization by HT using a proxy metric, the test LL. The detailed agorithm is shown in Algorithm 2.

Algorithm 3 Acceptance step

procedure ACCEPT($\mathcal{D}$, $\widetilde{\mathcal{D}}$, $\{\widetilde{\eta}_i\}_{i=1}^K$, π^{t-1})
  for $i = 1, \ldots, K$ **do**
    $\hat{\pi}_i \leftarrow$ FIT($\mathcal{D}, \widetilde{\eta}_i$) ▷ Model training with true data and candidate hyperparameters
    $\ell_i \leftarrow$ TEST($\mathcal{D}_{\text{test}}$, $\hat{\pi}_i$) ▷ Evaluate the test LL
  Sort $\{\hat{\pi}_i\}_{i=1}^K$ in descending order by $\{\ell_i\}_{i=1}^K$
  for $i = 1, \ldots, K$ **do**
    if Accept $\hat{\pi}_i$ with probability Equation (1) **then**
      return $\hat{\pi}_i$ and ℓ_i
  return π^{t-1} and corresponding test likelihood.

3.4.2. ACCEPTANCE TESTING

Since the key assumption of *ProxyTune* is based on the similarity between proxy data and ground-truth data, there is no guarantee to that all of candidate hyperparameters $\{\widetilde{\eta}_i\}_{i=1}^K$ will result in models with reasonable generation quality. Thus, we propose an additional acceptance step to select suitable hyperparameters η^t and their corresponding model trained with $\mathcal{D}$ for generating the next proxy dataset $\widetilde{\mathcal{D}}^{t+1}$.

For the acceptance step, we first train each model with the

true dataset $\mathcal{D}$ for the top K candidate hyperparameters $\{\widetilde{\eta}_i^t\}_{i=1}^K$ (assuming they have been ranked by target metric of the proxy), which results in K candidate proxies $\{\hat{\pi}_i^t\}_{i=1}^K$. For each candidate model, we compute $p(\mathcal{D}_{\text{test}} \mid \hat{\pi}_i^t)$ as the generation quality measure since LL is a good indicator for model fitting. Other generation quality measures can also be easily adapted to *ProxyTune*. Then, those proxies will be sorted in a descending order based on LL and the corresponding acceptance probability will be computed in this order. We propose the following for acceptance probability:

$$s_i = \min\left(\left(\frac{p(\mathcal{D}_{\text{test}} \mid \hat{\pi}_i^t)}{p(\mathcal{D}_{\text{test}} \mid \pi^{t-1})}\right)^{\frac{1}{\beta}}, 1\right), \tag{1}$$

where s_i is the acceptance probability of $\hat{\pi}_i^t$, $0 < \beta \leq 1$ is the temperature that controls the sensitivity to the likelihood ratio; π^{t-1} is the previously accepted proxy. When $\beta \to 0$, the acceptance is greedy and s_i is closed to 0 or 1, implying that the candidate is only accepted with higher LL than π^{t-1}. The detailed algorithm is in Algorithm 3.

Acceptance During the acceptance, we will compute s_i and decide if the candidate is accepted or not one at a time. If no candidate is accepted after top K candidates, there are two possible choices: (1) **single-batch** (SB): we stop the acceptance and $\pi^t = \pi^{t-1}$, $\eta^t = \eta^{t-1}$; (2) **loop-batch** (LB): we move onto the next batch of candidates $\{\widetilde{\eta}_i^t\}_{i=K+1}^{2K}$ and repeat the above until one is accepted. If still no candidate is selected, we set $\pi^t = \pi^{t-1}$, $\eta^t = \eta^{t-1}$. Note that the computational cost of LB is stochastic and cannot be precisely controlled. Algorithm 3 illustrates the detailed acceptance test with SB.

SB is computationally cheaper compared to LB, but has the possibility of missing potentially good candidates. Especially when the proxy data is poorly generated, the top candidates may have poor $p(\mathcal{D}_{\text{test}} \mid \hat{\pi}_i^t)$, and SB will not accept any new model in the following steps. On the other hand, LB will loop over all the candidates to pick a model with better LL, which has a better chance to escape from this sub-optimality. Thus, we will use LB as the default choice.

3.4.3. SEARCH

The tuning step is equivalent to HT on the proxy, where the access to the data generating process allows tuning on previously inaccessible metrics. In this work we mainly rely on random search to avoid the effectiveness of other factors for better ablation and allows a greater degree of parallelization. But other choices, like BO, can be adapted in practice.

Specifically, based on a fixed computational budget C, we will randomly sample C hyperparameters $\{\widetilde{\eta}_i\}_{i=1}^C$ from some fixed distribution H. For each hyperparameter, we fit a candidate $\hat{\pi}_i = \text{FIT}(\widetilde{\eta}_i, \widetilde{\mathcal{D}})$. Then, we can SCORE each $\hat{\pi}_i$ and rank them in a descending order, indicating the quality of each candidate. Algorithm 4 describes the detailed algorithm.

Algorithm 4 Search the candidate hyperparameters $\widetilde{\eta}$

procedure SEARCH($\widetilde{\mathcal{D}}$, K, H, C)
  for $i = 1 \ldots, C$ **do**
    $\widetilde{\eta}_i \sim H$
    $\hat{\pi}_i \leftarrow \text{FIT}(\widetilde{\mathcal{D}}, \widetilde{\eta}_i)$ ▷ Model training with proxy data and candidate hyperparameters
    $\mu_i \leftarrow \text{SCORE}(\widetilde{\mathcal{D}}, \hat{\pi}_i)$ ▷ Score the trained model with target metric
  $\{\widetilde{\eta}_i\}_{i=1}^K \leftarrow$ pick top K candidate hyperparameters based on μ_i
  return $\{\widetilde{\eta}_i\}_{i=1}^K$

3.4.4. *ProxyTune* FOR CAUSAL DISCOVERY

Here we describe the details of how *ProxyTune* can be used to select hyperparameters for Causal Discovery. We limit our considerations to generative models that learn the full structural equation model (SEM), such as DECI (Geffner et al., 2022).

With a model π, one can generate a proxy dataset $\widetilde{\mathcal{D}}^t$, allowing the access to the corresponding causal graph. In the following, we will take additive noise model (ANM) as an example and describe how the proxy data can be generated.

Let $\mathbf{x} = (x_1, \ldots, x_D)$ be a collection of random variables. SEMs (Pearl, 2009) describe the causal relationships between the individual variables x_i. Given a causal graph G on nodes $\{1, \ldots, D\}$, the SEM for x_i is $x_i = F_i\left(\mathbf{x}_{\text{pa}(i;G)}, z_i\right)$, where z_i is an exogenous noise variable that is independent of all other variables, $\text{pa}(i; G)$ is the set of parents of node i in G, and F_i specifies the functional relationship. ANMs (Hoyer et al., 2008; Peters et al., 2014; Geffner et al., 2022) are defined as:

$$\begin{aligned} F_i\left(\mathbf{x}_{\text{pa}(i;G)}, z_i\right) &= f_i\left(\mathbf{x}_{\text{pa}(i;G)}\right) + z_i \\ \text{or} \quad \mathbf{x} &= f_G(\mathbf{x}) + \mathbf{z} \quad \text{in vector form.} \end{aligned} \tag{2}$$

By fixing the noise vector $\mathbf{z}$, the vector form describes a fixed point equation. Therefore, one can sample a data point by (1) sampling a noise vector $\mathbf{z}$; and (2) solving the fixed point equation by iteratively plugging the $\mathbf{x}$ back into $f_G(\cdot) + \mathbf{z}$. Since the causal graph G is a DAG, this update equation will converge to a stable fixed point at most D steps. We can repeat the above steps to generate the entire proxy dataset $\widetilde{\mathcal{D}}$.

Hyperparameter	Description	Distribution
λ_s	Sparsity penalty coefficient controlling graph sparsity	$U[0, 1000]$
λ_{fr}	Learning rate for the functional relationships	$U[10^{-5}, 10^{-1}]$
λ_{noise}	Learning rate for the noise distributions	$U[10^{-8}, 10^{-3}]$
$\lambda_{\text{adjacency}}$	Learning rate for the variational graph distribution	$U[10^{-5}, 10^{-1}]$

Table 1. Search space H for *ProxyTune*. Here, U represents a uniform distribution over the specified ranges. For detailed implication of each hyperparameter, please refer to (Geffner et al., 2022).

4. Experiment

In this section, we investigate the effectiveness of the three proposed HT methods. Specifically, we want to study two questions: (1) Is LL a good proxy metric for causal discovery? (2) Does iterative refinement (i.e. *ProxyTune*) provide improvements under a fixed computation budget N? We refer to Section 4.2 and Section 4.3, respectively. In summary, we show that (1) test LL is a reasonable but not perfect proxy metric for causal discovery, since it can select a model with poor F1 score; (2) a reasonable number of refinement steps will improve F1. Longer or shorter runs will harm performances.

4.1. Experiment Setup

Synthetic data Generation For ground truth dataset generation, we choose a commonly used benchmark generation mechanism for causal discovery (Zheng et al., 2018; Geffner et al., 2022; Lachapelle et al., 2019; Zheng et al., 2020). First, we sample a ground-truth directed acyclic graph (DAG) G from Erdős-Rényi (ER) graph distribution with 16 nodes and 64 expected edges. Then, we sample the structural equations $x_i = f_i\left(\mathbf{x}_{\text{pa}(i;G)}\right) + z_i$ where $\mathbf{x}_{\text{pa}(i;G)}$ are the parents of node x_i given graph G; $z_i \sim \mathcal{N}(0, 1)$ and f_i is a multi-layer perceptron (MLP) with randomly sampled weights. We sample 5000 data points for both $\mathcal{D}$ and $\mathcal{D}_{\text{test}}$. We sample 6 datasets with different seeds.

ER graph generation Specifically, the edge probabilities are independent to each other and equal to $\frac{V}{(D,2)}$, where $(D, 2)$ is the the number of combinations of 2 variables among D variables; and V is the number of expected edges. Then, we sample an adjacency matrix $\boldsymbol{G}$ with the above edge probabilities, where $G_{i,j} = 1$ represents the edge $i \to j$. Since $\boldsymbol{G}$ does not guarantee a DAG, we post-process it by (1) randomly permutating the nodes (i.e. equivalent to the permutation of the corresponding rows and columns); and (2) take the lower triangular part of $\boldsymbol{G}$ to ensure DAGness.

Sampling the generation mechanism Based on the sampled graph $\boldsymbol{G}$, one can easily infer the topological order of variables. For simplicity, we assume $d = 1, \ldots, D$ have been arranged in this order. Then, we generate the dataset based on the following equation:

$$F_i\left(\mathbf{x}_{\text{pa}(i;G)}, z_i\right) = f_i\left(\mathbf{x}_{\text{pa}(i;G)}\right) + z_i \tag{3}$$

where $z_i \sim N(0, I)$ and f_i is a two-layer neural network with 100 hidden randomly sampled units and sigmoid activation. The weights for each layer are obtained by (1) sampling according to a uniform distribution between 0.5 and 2; (2) randomly negating half of the weights. We repeat the above procedure for each f_i to obtain the generation mechanism.

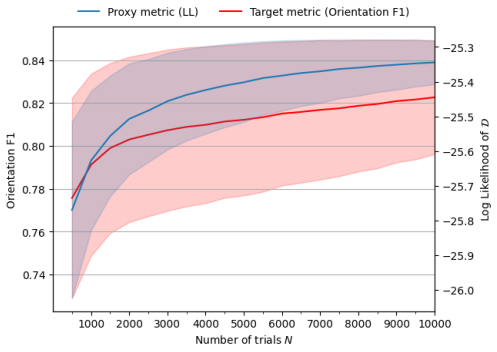

Figure 2. Bootstrapped estimates of the proxy and target metrics for HT with 500 to 10 000 trials. Each line shows the expected value of the winning trial as selected by the PM, while the filled areas show the standard deviation.

Model and target metrics We adopt DECI (Geffner et al., 2022) as our candidate model[1]. Since there are many choices for hyperparameters, we limit ourselves to the important parameters that affect the final performance in this preliminary study. Table 1 specifies the choice of hyperparameters and their corresponding search space. Each candidate hyperparameter is randomly sampled from the uniform distribution (U) over the specified ranges.

4.2. Insufficiency of proxy metric

First, we investigate the effectiveness of LL as the proxy metric for causal discovery. Figure 2 plots the bootstrap performance of PM with different budgets on a single dataset.

[1]https://github.com/microsoft/causica

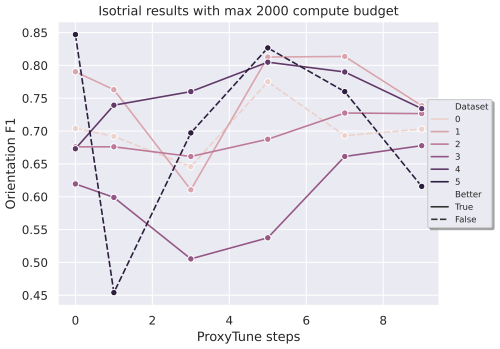

Figure 3. Each point represents the F1 of the model selected by *ProxyTune* with corresponding total tuning steps. Step 0 indicates the PM performance and step 1 represents the Credence (Parikh et al., 2022). We select step 7 as the best proxy tune steps number of *ProxyTune*. The solid line represents *ProxyTune* at step 7 is the best among all methods, and dashed line indicates the worse performance than method (1).

For each budget, we bootstrap the performance by selecting among 10000 training runs on $\mathcal{D}$. We repeat the bootstrapping 1000 times to estimate the performance mean and variance. As expected both LL and F1 increase monotonically w.r.t. the trial budget. However, we observed a large variation of F1 even with $N = 10000$ budgets. Specifically, performance with budget $N = 3000$ overlaps significantly with $N = 10000$. This indicates that despite LL showing overall positive correlations, HT with it does not guarantee the best hyperparameters and may result in a poor model. To demonstrate this more clearly, we plot the trend of F1 w.r.t. test LL for this dataset in Figure 1. We can see that large oscillations are present, which explains the large variance in Figure 2.

4.3. Isotrial experiments

Next, we investigate the effectiveness of the three methods under a fixed total compute budget. We call these *isotrial* experiments. The aim of the experiment is to demonstrate the advantages of iterative refinement under a fixed budget.

First, we fix a total budget N and total ProxyTune step T. We will then compute the per-step budget $\lfloor \frac{N}{T} \rfloor$ for HT on the proxy dataset and for the acceptance step. One thing worth noting is that the actual per-step consumption of *ProxyTune* is stochastic due to the LB setup. Therefore, for a fair comparison, we set $N = 1000$ for *ProxyTune* with $T > 0$ (i.e. *Credence* and *ProxyTune*) and $N = 2000$ for PM. This is because the worst-case per-step cost is $2\lfloor \frac{N}{T} \rfloor$ for *ProxyTune* (i.e. half for HT and half for acceptance). Therefore, we need to halve the total budget so that the worst-case cost is equivalent to PM ($N = 2000$). But the actual consumption of *ProxyTune* is usually much cheaper. We use a temperature of $\beta = 0.001$ for greedy acceptance.

Then we run *ProxyTune* with a total of T ProxyTune steps and select the final winner based on test LL as Algorithm 1. In the end, we extract the corresponding F1, and plot a single point in Figure 3, sitting at the position with T *ProxyTune* steps. We repeat the above process for $T = 0, 1, 3, 5, 7, 9$ with different datasets. The best *ProxyTune* step number is chosen as $T = 7$, which achieves a great compromise between distributing budgets to longer iterative refinements or thorough per-step HT search.

From Figure 3, we can clearly observe the effectiveness of iterative refinement. Firstly, we observe a trend such that after step $T = 3$, running longer iterative refinements seems to improve the F1 until step $T = 7$. This suggests a compromise between distributing the budget to longer iterative refinements or more thorough per-step proxy HT. With longer steps T, less budget is consumed for the per-step HT, resulting in poor hyperparameter candidates. On the other hand, if one uses fewer iterative steps, an excellent hyperparameter for the proxy dataset can be selected. But the quality of the generation may be poor due to the lack of iterative refinements. We can see that *ProxyTune* achieves the best performance among three methods in 4 out of 6 datasets, suggesting that *ProxyTune* is a promising method for HT. In contrast, *Credence* underperforms the other two methods for most datasets.

We also plot the trend of LL and F1 of a single *ProxyTune* run with $T = 9$ in Figure 4. We can see that *ProxyTune* provides a regularization effect which increases the correlation between LL and F1Specifically, the model corresponds to highest test LL typically has the best F1. This suggests LL is a good selection metric together with *ProxyTune*, but should not be used alone.

5. Conclusion

In this work, we examined three HT methods, *ProxyMetric*, *Credence* and *ProxyTune*, on their effectiveness for causal discovery tasks. In particular, we propose *ProxyTune*, which extends *Credence* with iterative refinements. We experimentally demonstrate the difficulty in relying on proxy metrics such as the LL in causal discovery. Finally, we show preliminary results on how *ProxyTune* performs on synthetic datasets, providing a promising signal on its effectiveness. Despite this, some difficulties remain. In particular, future work is needed to demonstrate its superiority in the case of comprehensive synthetic and real datasets with diverse tasks as well as expand the application of *ProxyTune* to further tasks without access to groundtruth metrics.

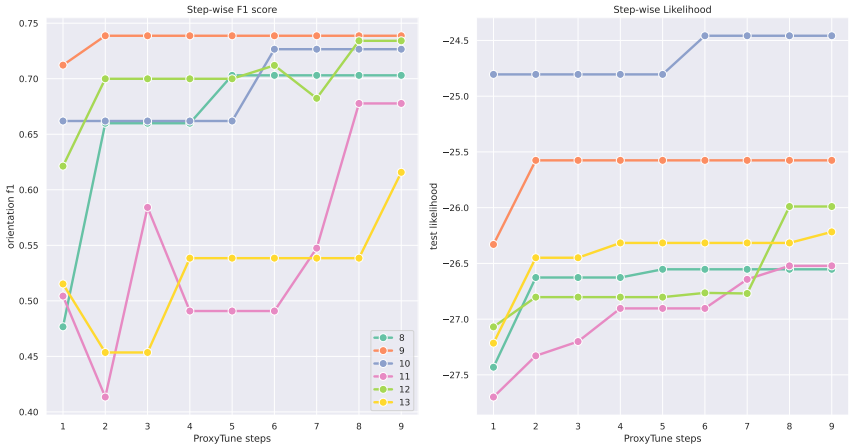

Figure 4. The step-wise plot of *ProxyTune* across different datasets. Note that these plots are different from the isotrial plots (Figure 3). Here, each line is generated by examining the step-wise trend with $T = 9$. (**Left**) It plots the step-wise F1 score. (**Right**) It plots the corresponding step-wise test likelihood.

References

Alaa, A. and Van Der Schaar, M. Validating causal inference models via influence functions. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 191–201. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/alaa19a.html.

Annadani, Y., Pawlowski, N., Jennings, J., Bauer, S., Zhang, C., and Gong, W. Bayesdag: Gradient-based posterior inference for causal discovery. *Advances in Neural Information Processing Systems*, 36, 2024.

Biza, K., Tsamardinos, I., and Triantafillou, S. Tuning causal discovery algorithms. In Jaeger, M. and Nielsen, T. D. (eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pp. 17–28. PMLR, 23–25 Sep 2020. URL https://proceedings.mlr.press/v138/biza20a.html.

Frazier, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Geffner, T., Antoran, J., Foster, A., Gong, W., Ma, C., Kiciman, E., Sharma, A., Lamb, A., Kukla, M., Pawlowski, N., et al. Deep end-to-end causal inference. *arXiv preprint arXiv:2202.02195*, 2022.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21, 2008.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

Machlanski, D., Samothrakis, S., and Clarke, P. Hyperparameter tuning and model evaluation in causal effect estimation. *arXiv preprint arXiv:2303.01412*, 2023.

Parikh, H., Varjao, C., Xu, L., and Tchetgen, E. T. Validating causal inference methods. In *International Conference on Machine Learning*, pp. 17346–17358. PMLR, 2022.

Pearl, J. *Causality*. Cambridge university press, 2009.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15(58):2009–2053, 2014. URL http://jmlr.org/papers/v15/peters14a.html.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Rubin, D. B. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Saito, Y. and Yasui, S. Counterfactual cross-validation: Stable model selection procedure for causal inference models. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8398–8407. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/saito20a.html.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. Learning sparse nonparametric dags. In *International Conference on Artificial Intelligence and Statistics*, pp. 3414–3425. Pmlr, 2020.