Feature Acquisition using Monte Carlo Tree Search

Abstract

Feature acquisition algorithms address the problem of acquiring informative features while balancing the costs of acquisition to improve the learning performances of ML models. Previous approaches have focused on calculating the expected utility values of features to determine the acquisition sequences. Other approaches formulated the problem as a Markov Decision Process (MDP) and applied reinforcement learning based algorithms. We focus on 1) formulating the feature acquisition problem as a MDP and applying Monte Carlo Tree Search, 2) calculating the intermediary rewards for each acquisition step based on model improvements and acquisition costs and 3) simultaneously optimizing model improvement and acquisition costs with multi-objective Monte Carlo Tree Search. With Proximal Policy Optimization and Deep Q-Network algorithms as benchmark, we show the effectiveness of our proposed approach with experimental study.

1 Introduction

Many machine-learning algorithms work with the assumption that all features have been observed and available during training and testing times or the missing data are disregarded as unacquired. Feature acquisition, a process in which further relevant data are acquired at variable costs, addresses this assumption to more closely align with some real-world applications [Huang *et al.*, 2018]. For medical diagnostic tasks, from the basis of incomplete features, doctors sequentially obtain additional test results until they obtain sufficient information to make adequate diagnoses of the patients. Determining which features to acquire is dependent on the previous diagnostic observations and the sequence at which the features are obtained can vary from patient to patient. Although accurate diagnoses are more likely with additional features, acquiring them incurs variable costs and is balanced with the improvement in performance [Melville *et al.*, 2004].

Previous studies on the feature acquisition problem address the trade-off between acquisition costs and performance improvement and the sequential decision making process, and are categorized into non-reinforcement learning and reinforcement learning (RL) approaches. Non-RL approaches focus on selecting the most informative features to acquire based on their utility values. These methods, [Melville *et al.*, 2004], [desJardins *et al.*, 2010], and [Huang *et al.*, 2018], estimate the expected utility of a feature for improving the model performance and acquire the feature with maximum expected utility. Although these methods provide a framework for feature acquisition based on utility values, they focus on subsets of features to acquire at a time, do not consider acquisition costs, or treat the model performance and acquisition costs as an aggregated single objective. RL approaches, [Contardo *et al.*, 2016], [Shim *et al.*, 2018], and [Li and Oliva, 2021], formulate the feature acquisition problem as a Markov decision process (MDP), where the state is the set of currently acquired features and the action is the acquisition of the next feature, and learn the best feature acquisition policy. For each acquisition step, the acquisition cost is incurred and defined as the reward for the action. Prediction error is calculated when the episode ends or the agent decides to stop the acquisition process. Additionally, the additive constraint of the acquisition costs to the rewards also necessitates further fine-tuning of a regularization parameter.

Monte Carlo Tree Search, [Kocsis and Szepesvári, 2006], for feature acquisition has the advantage over other RL algorithms in the fact that the reward (prediction) is obtained only at the end of an episode. Our Monte Carlo Tree Search (MCTS) approach also considers intermediary rewards for each acquisition step. We model the reward for each feature acquisition action as the division of the classification prediction probability with the feature being acquired by the cumulative incurred acquisition costs. The cumulative incurred acquisition costs are normalized by the cost of all features.

We also propose the trade-off between acquisition costs and model performance as a multi-objective optimization (MO) problem. In MO-MCTS, we model the costs and classification prediction probabilities as two conflicting objectives to be optimized simultaneously. Previous studies have applied the RL algorithms on the additive scalar aggregation of the two objectives. The two policies may be incomparable and the Pareto optimal set of solutions need to be found [Wang and Sebag, 2012]. We modify the algorithm presented in [Wang and Sebag, 2012] to find the Pareto optimal solution for each feature acquisition step and incorporate it within MCTS.

In comparison to the Proximal Policy Optimization, [Schulman *et al.*, 2017], and Deep Q-Network, [Mnih *et al.*, 2015], algorithms, our Monte Carlo Tree Search approach shows performance improvements in all the data sets we considered, with the relative improvement in the range of 1.2% to 25.1%. The multi-objective Monte Carlo Tree Search implementation shows an advantage in tight budget situations, as it leads to more variable feature acquisition sequences and can thus satisfy different cost budgets and confidence thresholds.

Our main contributions in this work are as follows.

- We propose to apply Monte Carlo Tree Search (MCTS) for the first time to the feature acquisition problem.
- We apply multi-objective MCTS to optimize feature acquisition costs and classification prediction probabilities simultaneously.
- We show the advantages of our proposed approaches on three medical data sets and the MNIST data set in comparison to two reinforcement learning approaches, Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017] and Deep Q-Network (DQN) [Mnih *et al.*, 2015]).

Related works are reviewed in Section 2. Section 3 presents our approaches in detail. Experimental setup and results are presented in Section 4.

2 Related Works

2.1 Feature Acquisition

Previous non-RL approaches address the feature acquisition problem from the expected utility of an unacquired feature. [Melville *et al.*, 2004] quantifies an Uncertainty Score for a feature, which is defined as the absolute difference between the estimated class probabilities of the two most likely classes when trained with the feature. [desJardins *et al.*, 2010] calculates a Confidence Score for a subset of features based on an ensemble of classifiers. [Huang *et al.*, 2018] incorporates an iterative supervised matrix completion algorithm with the variance of a feature after the iterations as its utility. [Melville *et al.*, 2004] does not consider acquisition costs, but others incorporate them by first sorting the unacquired features by costs, [desJardins *et al.*, 2010], or constructing an objective function with the acquisition costs and applying gradient descent, [Huang *et al.*, 2018]. [Contardo *et al.*, 2016] applies PPO to the policy network. Similarly, [Shim *et al.*, 2018] also considers the DQN to model the feature acquisition policy. [Li and Oliva, 2021] uses a pretrained surrogate model to estimate both the state transitions and the prediction in a unified model in which the intermediate prediction errors based on information gain are also calculated. The classification errors and acquisition costs are additively aggregated into a single objective function in these RL approaches. With the exception of [Li and Oliva, 2021], prediction errors are also only calculated at the end of an episode.

2.2 Monte Carlo Tree Search

By applying the Upper Confidence Bounds (UCB) bandit algorithm, [Auer *et al.*, 2002], MCTS iteratively searches the state space while balancing the exploration of suboptimal actions and exploitation of optimal actions [Kocsis and Szepesvári, 2006]. AlphaGo and its variants also utilize a neural network in conjunction with MCTS. This network outputs a vector of move probabilities and a scalar value estimation from the position state s and is used as both policy and value networks. The network is then used to guide the simulations and is iteratively trained using the results from self-play [Silver *et al.*, 2017]. In our approach, we consider the default uniform random policy for the simulations and similarly consider iteratively training the acquisition policy based on the simulations.

2.3 Multi-objective Monte Carlo Tree Search

For multi-objective reinforcement learning problems, previous approaches have focused on optimization based on the total order of the solutions and aggregation of the vectorial objectives into a scalar objective function. Similar to the previous RL approaches, weighted summation of the different objectives has been a popular choice [Wang and Sebag, 2012]. For conflicting objectives, this strategy does not lead to an optimal policy, as there exists a set of optimal solutions ordered along the Pareto Front [Wang and Sebag, 2012]. [Wang and Sebag, 2012] proposes a hypervolume indicator based scalarization scheme, where the rewards maximizing the indicator belong to the Pareto Front [Fleischer, 2003]. [Painter *et al.*, 2020] provides a linear transformation scheme to achieve scalarization. In our approach, we closely follow the algorithm in [Wang and Sebag, 2012].

3 Feature Acquisition using Monte Carlo Tree Search

3.1 Problem Statement

Consider a predictive task with feature vector $X \in \mathbb{R}^d$ and class y. For $C \in \{1, \cdots, d\}$, we denote vector $X_C = (X_i)_{i \in C}$. Starting from an empty set of features, we perform a sequential feature acquisition process. We address the case where we obtain complete information with all the features acquired for their ground-truth values. The aim of the process is to obtain the sequences of feature acquisition steps that maximize the task performance while minimizing the acquisition costs.

We formulate the problem as a Markov decision process

$$\begin{aligned} s_t &= X_{O_t}, \\ a_t &\in A_t = \{1, \cdots, d\} \setminus O_t, \\ r_t &= \frac{P(\hat{y} | X_{O_t \cup \{a_t\}})}{\frac{\sum_{i=0}^{t} C_i}{C_{\text{total}}}}. \end{aligned}$$

We consider episodic solutions from the empty set of features ($t = 0$) to the complete set of features ($t = d$). At a given time, the agent is in state s_t and selects a feature to acquire (a_t) according to its policy. The agent then receives the reward r_t from the environment and transitions to the state $s_{t+1} = X_{O_t \cup \{a_t\}}$. The goal of the agent is to maximize the cumulative rewards.

State. The state at time t, s_t, is the X_{O_t}, the values of the already acquired feature subset $O_t \subseteq \{1, \cdots, d\}$.

Action. The action space at time t is the unacquired feature set A_t. The action at time t is then the acquisition step for a candidate feature with its value X_{a_t}.

Reward. The reward at all times of the episode is defined as the fraction of the classification prediction probability and the normalized incurred acquisition costs up to time t. The prediction is made with the feature vector consisting of the acquired feature subset $X_{O_t \cup \{a_t\}}$. The incurred acquisition costs $\sum_i C_i$ is normalized by the total cost C_{total} of all features.

3.2 Monte Carlo Tree Search for Feature Acquisition

We present the Upper Confidence Tree MCTS algorithm with our approach-specific implementation details. Starting from an empty feature state as the root node, MCTS explores and builds a search tree with N simulations. Each simulation consists of three phases [Świechowski *et al.*, 2021].

Selection. Starting from the root node, a feature is selected iteratively until arriving at a leaf node. The set A_{s_t} of admissible features in node/state s_t defines the child nodes of s_t. Feature selection according to the maximization of the Upper Confidence Bound, Auer [Auer *et al.*, 2002], reads

$$a_t^* = \underset{a_t \in A_{s_t}}{\arg\max}\ Q(s_t, a_t) + c\sqrt{\ln(n_{s_t})/n_{s_t,a_t}}, \qquad (1)$$

where $Q(s_t, a_t)$ is the average cumulative reward of feature a_t, n_{s_t} is the visit count of node s_t, and n_{s_t,a_t} is the number of times a_t has been selected in node s_t. The exploration and exploitation trade-off is controlled by the hyperparameter c, which is optimized as described in a next section.

Expansion. Once a leaf node has been selected, all the absent child nodes of the leaf node are added to the tree.

Simulation. Starting from the leaf node, a feature is selected uniformly at random until the terminal state is reached. Differently from previous studies in AlphaGo and its variants, we utilize the uniform random policy as our default simulation policy. As defined in the previous section, we compute the reward for each feature and calculate the cumulative reward.

Backpropagation. During backpropagation, $Q(s_t, a_t)$, n_{s_t,a_t}, and n_{s_t} are updated

$$\begin{aligned} r_{s_t,a_t} &= \sum_{t'=t}^{d} r_{t'}, \\ n_{s_t,a_t} &= n_{s_t,a_t} + 1, \\ n_{s_t} &= n_{s_t} + 1, \\ Q(s_t, a_t) &= \frac{r_{s_t,a_t}}{n_{s_t,a_t}}. \end{aligned}$$

After N simulations and updated statistics using backpropagation, the feature acquisition action is defined as

$$a_t^* = \underset{a_t \in A_{s_t}}{\arg\max}\ Q(s_t, a_t). \qquad (2)$$

The next state is then obtained according to the acquisition step and N further simulations are conducted with the next state as the new root node. This process continues until the terminal, complete feature state is reached.

We have two variants of the MCTS algorithm. In the **standalone** implementation, we conduct MCTS training by constructing a search tree for each sample in the training data set. The visited states and their Q values are then stored for the entire training data set. This stored set is then used to calculate the next feature probabilities for each visited state. The next feature probabilities are calculated with the cumulative Q values for each admissible feature. We then train a policy network with the visited states and their next feature probabilities.

In the **integrated** implementation, we embed a policy network in the training phase and periodically train the network during MCTS training. After initializing with random weights, the network is then used to guide the feature acquisition step. The network is periodically trained with visited states and their next feature probabilities. We also optimize the network train frequency.

The pseudocodes for our **integrated** implementation is shown in Algorithm 1. We highlight the problem specific details in embedding the policy network and its training on the visited states and their next feature probabilities.

3.3 Feature Acquisition using Multi-objective Monte Carlo Tree Search

In this section, we present the multi-objective-MCTS algorithm in [Wang and Sebag, 2012] with our modifications in the reward formulation and scalarization, and Pareto Front approximation.

Vectorial Rewards. We define the reward for all timesteps in an episode as the vector of negative normalized incurred acquisition costs and classification probability. During backpropagation, the rewards are updated component-wise as

$$\begin{aligned} r_c &= \sum_{t'=t}^{d} r_{t',c}, \\ r_p &= \sum_{t'=t}^{d} r_{t',p}, \end{aligned}$$

where $r_{t',c}$ and $r_{t',p}$ are the negative normalized incurred costs and classification probabilities, respectively.

Pareto Front Approximation. In [Wang and Sebag, 2012], an approximation to the Pareto Front is maintained during training, which we use in the UCB feature selection and feature acquisition policy. When new nodes are added during the expansion and simulation phases, the Pareto Front approximation is updated with the vectors of normalized incurred costs and classification probabilities of the added nodes. We then determine the non-dominated set and denote it as **P**. We use **P** as the estimated Pareto Front for the data set. The pseudocode with the modifed expansion and simulation is shown in Algorithm 2.

Reward Scalarization. We calculate the hypervolume indicator as the reward scalarization method

$$HV(r; z) = \mu\left(r; z\right),$$

which is defined as the Lebesgue measure with respect to a reference point z [Fleischer, 2003]. Vector z is set at $(-1.0, 0)$ so that it is dominated by every $r \in \mathbf{P} \cup \{r\}$. Then, the

Algorithm 1 Single-objective Monte Carlo Tree Search (Integrated)

Input: Iteration number I, initial policy network weights θ, policy network update frequency f
Output: MCTS trained policy network weights θ

Initialize policy network ϕ with θ
Initialize list L of visited nodes and their Q and visit counts N
$i \leftarrow 0$
for sample = 1,2,. . .,m **do**
    $i \leftarrow i + 1$
    Initialize state s_0
    Create root node v_0 with s_0
    $Q(v_0)$: reward of v_0
    $N(v_0)$: visit count of v_0
    $C(v_0)$: children of v_0
    $a(v_0)$: action of v_0
    while v_0 not terminal **do**
        MCTS(v_0,I)
        $a \leftarrow \phi_\theta(s_0)$
        $v_0 \leftarrow$ **makeChild**(v_0, a)
    end while
    Append $Q(v)$ and $N(v)$ for v in **MCTS** to L
    if f % i == 0 **then**
        S, $A \leftarrow$ **preprocess**(L)
        Train ϕ_θ on S and A
    end if
end for

function **preprocess**(L)
    Input: List L of visited nodes and their Q and N
    Output: Visited nodes S and their next action probabilities A
Make each node v in L to be distinct with addition for $Q(v)$ and $N(v)$ for duplicates
$A = \vec{0}$
$S = v$ in L
for v in L **do**
    for action in A **do**
        Find child nodes of v in L
        for node in child nodes **do**
            A(action) $+= Q$(node)$/N$(node)
        end for
    end for
end for
Normalize A with division by max(A)
return S, A

modified Upper Confidence Bounds selection is

$$Q(s_t, a_t) = \frac{HV(\mathbf{P} \cup \{r\}; z)}{n_{s_t,a_t}},$$
$$a_t^* = \underset{a_t \in A_{s_t}}{\arg\max}\; Q(s_t, a_t) + c\sqrt{\ln(n_{s_t})/n_{s_t,a_t}}.$$

For the acquisition policy, the next state is obtained with the selected acquisition feature and serves as the next root node. We also embed the policy network in the training phase in the **integrated** implementation.

4 Experiments

4.1 Data Sets and Benchmark Algorithms

We use four data sets. **(a) Heart Failure (HF)** [Chicco and Jurman, 2020]: This data set contains medical records of 299 patients who had heart failure with 13 clinical features and 2 classes (boolean for death event). **(b) Coronary Heart Disease (CHD)** [FHS, 2022]: The Framingham Heart Disease data set contains medical records of 4,238 patients with 16 risk factors for coronary heart disease as features and the ten year presence of CHD as the class. **(c) PhysioNet** [Goldberger *et al.*, 2000]: The data set from the PhysioNet/CinC Challenge 2012 consists of medical records of 4,000 ICU stay patients. The data set has 39 clinical features with 2 classes for the death event. **(d) MNIST** [Deng, 2012]: Each 4×4 block is considered as a feature with 70,000 samples, 49 features, and 10 classes.

For the three medical datasets, acquisition cost is set at 1 and 7 for categorical and continuous features, respectively. These costs are determined by the costs of the medical tests required and comparing them to a previous data set where the relative costs of similar tests were quantified [Cestnik *et al.*, 1988]. For the MNIST data set, we also define blocks of 4×4 pixels as features. For each block, the acquisition cost is defined as 16 with 1 for each pixel.

For the experiments, we create 4 splits and use 3 seeds for the total of 3 experimental runs. For each data set, the $80/20$ split is used for the training and test samples.

We use Proximal Policy Optimization (PPO) and Deep-Q Network (DQN) as the baseline algorithms to compare to our approaches. For PPO, we also incorporate two variants of the algorithm: PPO-PG and PPO-AC with the difference in network update frequencies to reflect vanilla policy gradient and actor-critic methods, respectively.

4.2 Setup and Evaluation Metrics

For evaluation of the feature acquisition algorithms, we plot the F1 scores against the incurred acquisition costs. We then calculate the areas under the curves (AUCs) of the resulting F1 curves and average them across the splits and seeds. We also report the highest test F1 AUC values of the 3 experimental runs of each algorithm. Since the obtained feature acquisition sequences do not contain all the cost points up to the full cost of all features, we also extrapolate the F1 scores at these points with the F1 scores of lower costs that are visited by the solution policy. Figure 1 shows a sample run of our experiments.

Algorithm 2 Multi-objective Monte Carlo Tree Search

<u>function</u> **expand**(v)
Input: Node v
Output: Child nodes of v, their initialized vectorial R's, and updated Pareto Front approximation
for all unacquired actions $a \in A(v)$ **do**
  $v' \leftarrow$ **makeChild**(v, a)
  Add v' to $C(v)$
  $a(v') \leftarrow a$
  $R(v')[0] \leftarrow$ **classificationProbability**(v')
  $R(v')[1] \leftarrow$ **findCost**(v')
  $P \leftarrow$ **findGlobalP**$(P, R(v'))$
end for

<u>function</u> **simulate**(v)
Input: Node v
Output: Cumulative vectorial reward of v from simulation to the terminal state
reward = []
while v not terminal **do**
  Choose $a \in A(v)$ uniform randomly
  $v \leftarrow$ **make child**(v, a)
  R(v)[0] $\leftarrow R(v)[0]+$**classificationProbability**(v)
  R(v)[1] $\leftarrow R(v)[1]+$**findCost**(v)
  $P \leftarrow$ **findGlobalP**$(P, R(v))$
  reward$[0] \leftarrow$ reward$[0]$ +**classificationProbability**(v)
  reward$[1] \leftarrow$ reward$[1]$ +**findCost**(v)
end while
return reward

All experiments are run on a server with Intel core i9-13900k and NVIDIA GeForce RTX 3080 graphics card.

We use the logistic regression and neural network classifiers for the calculation of the rewards during training and for the evaluation of the F1 scores. To this end, we utilize the following 4 classifier strategies. **Pretrain**: The pretrain strategy uses classifiers trained on complete feature vectors. **Random**: The classifiers are trained on random subsets of the features. **Retrain**: Starting with the pretrain strategy, classifiers are retrained on the augmented data set with the feature vectors of states visited during training of the algorithms. The frequency at which the classifiers are retrained is optimized by the resulting AUC of the train F1 curve. **Fit**: In the fit strategy, each subset of the feature set is used to train a single classifier. Each classifier is used for the same subset of features whose states are visited. This strategy is considered for the HF, CHD, and PhysioNet data sets where the numbers of features are low.

For categorical features, the unacquired features are set as its own categories and we one-hot encode such features. For continuous features, we initialize at -1 (all feature values in our data sets are non-negative). With the MNIST data set, all the unacquired features are set at 0; this value is used for the policy networks and classifiers. For other data sets, we also utilize hyperparameters to determine how the values of the unacquired continuous features are set with respect to the acquisition costs in calculating classification prediction probabilities and training the policy networks. Using 0 at 0 cost and varying the values at full acquisition cost from 0 to a large negative value (this hyperparameter is set at -100 in our experiments), we fit a quadratic, linear, or constant function with the value at full cost. The best strategy is determined by the resulting AUCs of the train F1 curves for each algorithm and classifier. We then use the identified function for setting the all yet to be acquired continuous features.

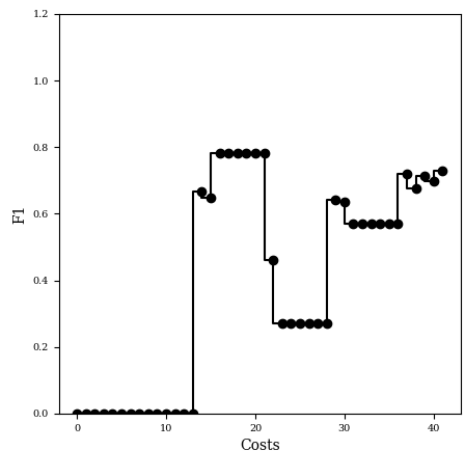

Figure 1: F1 score curve on the incurred acquisition costs.

Hyperparameters in the algorithms were optimized based on the resulting F1 AUCs. For PPO, the number of episodes, entropy and value coefficients and learning rates were optimized. The number of episodes, learning rates and ϵ-decay parameter were optimized in DQN. For MCTS, the number of simulations and UCB parameter were optimized. For the Retrain classifier strategy and the integrated implementations of MCTS, the retrain frequencies were also optimized.

4.3 Experimental Results

F1 AUC

The Monte Carlo Tree Search implementations show performance improvement from the benchmark algorithms for all data sets in Figure 2. Comparing the best performing MCTS implementation and the best performing benchmark algorithm, the relative improvements range from 1.2% to 25.1% and the logistic regression classifiers show higher improvement than the neural network classifiers with the exception of MNIST.

Heart Failure For the logistic regression classifier (LR), the SO-MCTS integrated implementation with the Pretrain strategy is the best performer with PPO-PG with the Fit strategy as the best benchmark. The SO-MCTS standalone implementation with the Pretrain strategy performs best and PPO-AC with the Random strategy is the best benchmark for the neural network classifier.

Coronary Heart Disease The SO-MCTS standalone implementation with the Retrain strategy is the best performer with PPO-PG with the Fit strategy as the best benchmark for LR. The MO-MCTS integrated implementation with the Random strategy performs best and PPO-PG with the Random strategy is the best benchmark for the neural network classifier.

	HF		CHD		PhysioNet		MNIST	
	LR Mean	LR Max	LR Mean	LR Max	LR Mean	LR Max	LR Mean	LR Max
SO-MCTS Standalone	52.7	70.7	**52.9**	53.9	51.9	62.0	56.4	61.4
SO-MCTS Integrated	**64.4**	67.1	51.6	53.9	**55.2**	61.0	**61.1**	64.2
MO-MCTS Integrated	59.5	65.9	49.6	53.3	46.3	52.2	57.2	58.9

	HF		CHD		PhysioNet		MNIST	
	NN Mean	NN Max	NN Mean	NN Max	NN Mean	NN Max	CNN Mean	CNN Max
SO-MCTS Standalone	**61.4**	70.0	59.8	60.2	52.2	59.1	62.9	72.4
SO-MCTS Integrated	**61.4**	71.5	59.0	62.0	**52.5**	55.3	**70.3**	77.0
MO-MCTS Integrated	60.0	65.9	**63.3**	63.7	52.2	53.6	**70.3**	72.0

Table 1: Summary tables of the MCTS implementations. Results are the percentages of the average F1 AUCs with respect to the highest possible F1 AUCs of total costs of full features. Mean are the average and max are the maximum individual experimental run. Maximum values from the implementations are in **bold**.

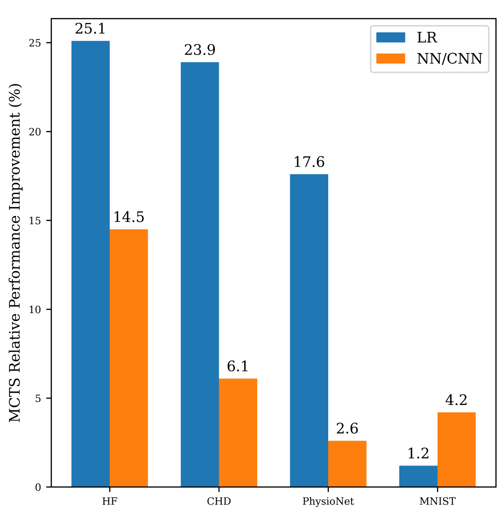

Figure 2: Relative differences between the best performing Monte Carlo Tree Search implementation and the benchmark algorithms (LR: logistic regression, NN/CNN: neural network/convolutional neural network).

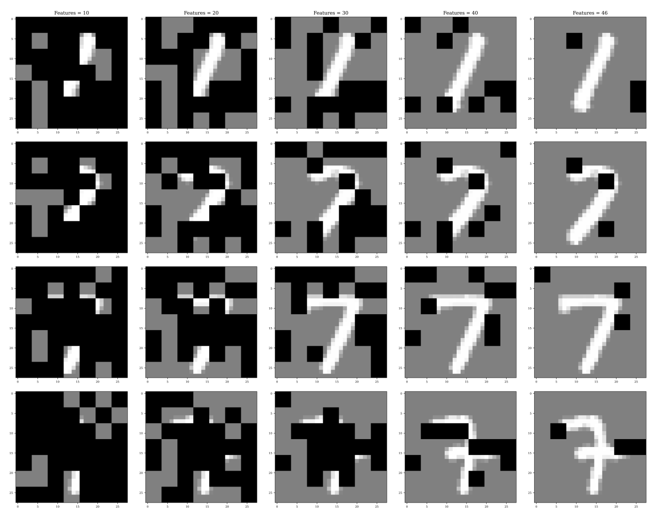

Figure 3: At the number of acquired features at 10, 20, 30, 40, 46, the unacquired features are plotted in black scale and the acquired features in gray scale. The top row shows an anticipated acquisition strategy of acquiring the informative pixels first before the background pixels. The second row exhibits a surprising acquisition strategy. The last two rows are the anticipated and surpring acquisition strategies where the cost of acquiring the features in the 16×16 pixel square in the middle is set to be 160.

PhysioNet For LR, the SO-MCTS integrated implementation with the Retrain strategy is the best performer with PPO-PG with the Random strategy as the best benchmark. For the neural network classifier, the SO-MCTS integrated implementation with the Random strategy performs best and PPO-PG with the Random strategy is the best benchmark.

MNIST The SO-MCTS integrated implementation with the Random strategy is the best performer with PPO-PG with the Random strategy as the best benchmark for LR. For the convolutional neural network classifier (CNN), the SO-MCTS integrated implementation with the Random strategy performs best and PPO-PG with the Random strategy is the best benchmark. For 10 randomly selected samples, we also visually analyze the resulting feature acquisition sequences at the numbers of acquired features of 10, 20, 30, 40, and 46 to determine that 70.0% of the samples are acquiring the informative digit pixels first before acquiring the background pixels. In Figure 3, the top row shows an anticipated acquisition strategy. Of the 70.0% samples exhibiting the anticipated behavior, at the number of acquired features points of 10 and 20, the informative pixels consist of 84.0% and 67.0% of the acquired pixels, respectively. The second row in Figure 3 exhibits a surprising acquisition strategy. We also set the cost of acquiring the features in the 16×16 pixel square in the middle to be 160 and visually compare to the case when the cost of acquiring each feature is 16. Of the randomly selected 10 samples, the higher cost experiment shows 25.0% of the samples acquir-

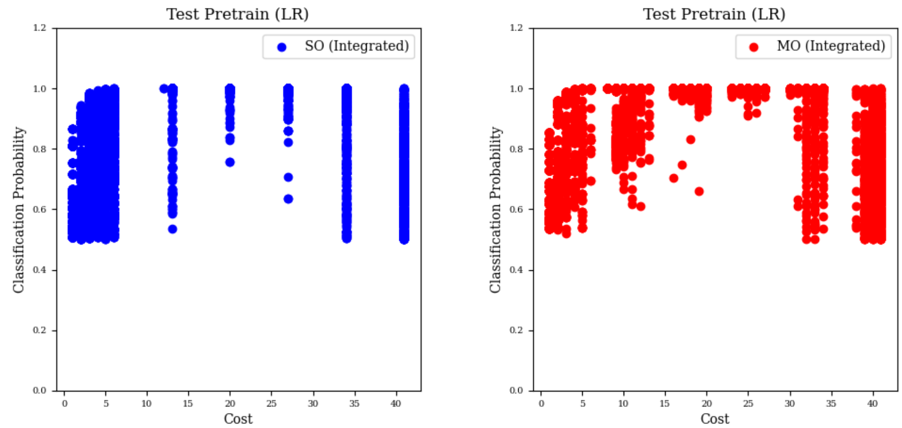

Figure 4: Solutions of the SO-MCTS and MO-MCTS integrated implementations for the Heart Failure data set.

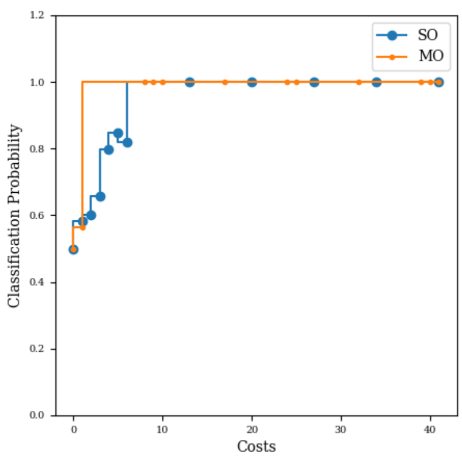

Figure 5: Sample feature acquisition sequences of the SO-MCTS and MO-MCTS integrated implementations for the Heart Failure data set.

ing the informative digit pixels before the background pixels. At the number of acquired features points of 10 and 20, the informative pixels in this case consists of 66.0% and 62.0%, respectively. The last two rows in Figure 3 show anticipated and surprising acquisition cases with higher cost. We note that the AUC with all equal cost is 0.556, but with higher cost it is 0.387 (when integrating AUCs, both maximum costs have been scaled to 1).

Comparison of the SO and MO MCTS Implementations Best performance results from our MCTS implementations are shown in Table 1. The results are shown as the percentages of the average F1 AUCs for each implementation with respect to the highest possible F1 AUCs of total costs of full features.

With the exception of the Coronary Heart Disease data set, the SO-MCTS integrated implementation has higher F1 AUCs than the MO-MCTS integrated implementation. We plot the solutions from the Heart Failure data set in the objective space in Figure 4. We see that **(1)** for lower costs, the SO-MCTS solutions are more frequent and **(2)** for higher costs, the SO-MCTS solutions are confined to cost regions that are separated by that of continuous features. This indicates that the SO-MCTS trained policy acquires the lower cost categorical features before the higher cost continuous features, whereas the MO-MCTS trained policy does not. For the Coronary Heart Disease with the random logistic regression classifier strategy, where the MO-MCTS integrated implementation has a higher F1 AUC, the solutions in the objective space are similar to the SO-MCTS integrated implementation with the policy acquiring the lower cost categorical features first before venturing to the higher continuous features.

In Figure 5 with sample acquisition trajectories from the Heart Failure data set, the SO-MCTS solution acquires the lower cost categorical features before acquiring the higher cost continuous features. For the MO-MCTS integrated implementation, the solution optimizes the classification probability and the acquisition cost simultaneously. We also observe that the MO-MCTS solution has more acquisition cost budgets (10) than the SO-MCTS solution (5) under which the classification confidence threshold of 1.0 can be reached. Since we considered the case of infinite budgets, where we obtained the ground-truth values for all the features, it is more advantageous to use the MO-MCTS implementation in tight budget situations. Since the MO-MCTS trained policy shows more diversity in the solution space, it provides more solutions matching variable budgets and confidence thresholds.

5 Conclusions

We studied the feature acquisition problem, where missing features in data are acquired for ground-truth values at variable costs. In comparison to the PPO and DQN algorithms, our MCTS implementations show performance improvements, with the relative improvement in the range of 1.2% to 25.1%. The multi-objective implementation shows an advantage over the single-objective implementation in budgeted situations, as it leads to more variable sequences and thus can satisfy different cost budgets and confidence thresholds.

References

[Auer *et al.*, 2002] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[Cestnik *et al.*, 1988] G. Cestnik, I. Konenenko, and I. Bratko. Assistant-86: A knowledge-elicitation tool for sophisticated users. *2nd European Working Session on Learning*, pages 31–54, 1988.

[Chicco and Jurman, 2020] D. Chicco and G. Jurman. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Informatics and Decision Making*, 20, 2020.

[Contardo *et al.*, 2016] G. Contardo, L. Denoyer, and T. Artieres. Sequential cost-sensitive feature acquisition. *Advances in Intelligent Data Analysis XV*, (284-294), 2016.

[Deng, 2012] L. Deng. The mnist database of handwritten digit images for maching learning research. *IEEE Signal Processing Magazine*, pages 141–142, 2012.

[desJardins *et al.*, 2010] M. desJardins, J. MacGlashan, and K. L. Wagstaff. Confidence-based feature acquisition to minimize training and test costs. *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 514–524, 2010.

[FHS, 2022] FHS. https://framinghamheartstudy.org/. 2022.

[Fleischer, 2003] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. *International Conference on Evolutionary Multi-Criterion Optimization*, (519-533), 2003.

[Goldberger *et al.*, 2000] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P.C. Ivanov, R. Mark, J.E. Mietus, G.B. Moody, C.K. Peng, and H.E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101:e215–e220, 2000.

[Huang *et al.*, 2018] S.J. Huang, M. Xu, M.K. Xie, M. Sugiyama, G. Niu, and S. Chen. Active feature acquisition with supervised matrix completion. *arXiv preprint arXiv:1802.05380*, 2018.

[Kocsis and Szepesvári, 2006] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *Proceedings of the 17th European Conference on Machine Learning*, pages 282–293, 2006.

[Li and Oliva, 2021] Y. Li and J. B. Oliva. Active feature acquisition with generative surrogate models. *Proceedings of the 38th International Conference on Machine Learning*, pages 6450–6459, 2021.

[Melville *et al.*, 2004] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. *Fourth IEEE International Conference on Data Mining*, pages 483–486, 2004.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, and A.A. Rusu. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[Painter *et al.*, 2020] M. Painter, B. Lacerda, and N. Hawes. Convex hull monte-carlo tree-search. *arXiv preprint arXiv:2003.04445*, 2020.

[Schulman *et al.*, 2017] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shim *et al.*, 2018] H. Shim, S.J. Hwang, and E. Yang. Joint active feature acquisition and classification with variable-size set encoding. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1375–1385, 2018.

[Silver *et al.*, 2017] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550, 2017.

[Świechowski *et al.*, 2021] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *arXiv preprint arXiv:2103.04931*, 2021.

[Wang and Sebag, 2012] W. Wang and M. Sebag. Multi-objective monte-carlo tree search. *JMLR: Workshop and Conference Proceedings*, 25:507–522, 2012.