Researching Methods in Active Feature Acquisition

Omar Hilal Shaban
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599
ohshaban@cs.unc.edu

Abstract

In traditional machine learning settings, models are evaluated on data that are assumed to always be available in full. As such, all features of a particular instance are required in order to make a prediction using such models. If any of the features are not available, they are regarded as missing data that would either have to be imputed or would render that observation unusable with that particular model. Active Feature Acquisition (AFA) considers models that can operate on input with missing data in order to (1) make a prediction, and (2) query the environment for unobserved features that will improve said prediction. In this paper, we explore various methods surrounding AFA in order to glean some insight on the performance of different methods and how they can be improved upon.

1 Introduction

Typical machine learning approaches involve learning a distribution for an output y given some input x: $p(y|x)$ where $x \in \mathbb{R}^d$ and represents the complete set of features being trained on.

While this method has proven useful in many settings and applications, it is a rather incongruous presentation of real-world decision systems in that predictions can only be made when a complete set of features for a particular observation is present during inference. In addition, these methods do not provide any means for the models that are trained to request additional information (features) at evaluation time in order to improve the quality of its prediction. In contrast, real-world domains typically allow for human agents to reason about data with missing features as well as about which information or steps would be most productive to take next.

Take the case of a doctor diagnosing a patient as an analogy. The doctor begins with some preliminary set of features about their patient. These features could include information about the patient's background, their symptoms, temperature, etc... but would naturally be missing details regarding biological test results that would provide a more comprehensive picture of any medical conditions. In this situation, it is within a doctor's agency to provide a preliminary diagnosis after a basic patient examination. However, a doctor may also decide to order a particular test (e.g. blood test) based on known information in order to attain additional information that would confirm or otherwise narrow down their suspicion of the patient's underlying condition. Active Feature Acquisition is the field of research that aims to address this dissonance between typical ML and real-world intelligence systems.

2 Background

2.1 Active Feature Acquisition

The basic idea behind Active Feature Acquisition is to develop models that can make inferences from observations with an incomplete set of features, and query the environment for whichever unobserved feature(s) would be most valuable to acquire based on what is already known of a

particular instance. Thus, different instances with different states of observed features can yield a different selection of features to be acquired (see Li and J. Oliva 2021). It is this very property, i.e. dynamic acquisition for a given instance at a given time step, that makes these methods different from traditional feature selection methods (Cai et al. 2018). Hence, this category of methods is called **Active** Feature Acquisition. Moreover, real-time feature acquisition and information retrieval often incurs some cost on the system (e.g. time, energy, money, etc...). These costs are often built into AFA models in order to learn policies that can factor in some cost-benefit analysis when it comes to improving prediction performance as in related works by Ling et al. 2004 and Nan et al. 2014.

2.2 AFA for Unsupervised Tasks

In addition to constructing Active Feature Acquisition models for supervised tasks, the problem setting can be generalized to work for unsupervised ML tasks as well. In such settings, the output being inferred would simply be the values of the unobserved features and thus we are learning the distribution $p(x_u|x_o)$ where x_o are the observed features and x_u are the unobserved. Much of the work in this paper will deal with experiments in the unsupervised setting.

2.3 AFA as Markov Decision Processes

In their paper, Shim, Hwang, and Yang 2018, propose modeling Active Feature Acquisition problems after Markov Decision Processes (MDPs), where the state is defined by the set of all currently observed features. The action space is the set of all unobserved features that could be acquired in the next step. In addition, the action space includes a termination action to trigger making a final prediction instead of any further acquisitions.

By defining a reward function that factors in both the prediction performance and acquisition cost, this MDP can then be solved in a reinforcement learning setting with an agent and environment. At each step, the RL agent acquires the value of an unobserved feature from the environment and pays for the cost of acquiring it. Eventually, the agent decides to take the termination action and make a final prediction using the features that it had acquired thus far.

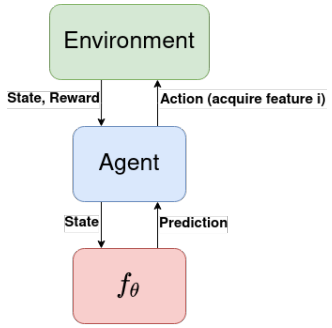


Figure 1: Illustration of the RL framework for AFA

2.4 AFA with Generative Surrogate Models

Li and J. Oliva 2021 introduce a novel approach to the AFA MDP forumation. They leverage the use of generative surrogate models that provide side information as well as intermediate rewards to the agent in order to guide its decision. The side information consists of output prediction and likelihood, imputations of unobserved features with their uncertainties, and estimated utilities of future acquisitions. The intermediate rewards are derived from the information gain of having acquired a particular feature.

The generative models used in GSMRL capture all the conditional dependencies between features $p(x_j|x_o)$, where x_j is an arbitrary feature. These arbitrary conditional distributions are then employed to inform the state transitions in the RL problem. Their method, Generative Surrogate Models for Reinforcement Learning (GSMRL), ultimately combines both model-free and model-based approaches into a single RL framework.

2.5 AFA with Black-Box Optimization (BBO)

In standard RL techniques, the goal is learn a policy for the agent's decisions such that the expected value of some reward function is maximized over time. This is typically done by computing function gradients and making parameter updates to the policy network in the direction that maximizes the expected reward.

Alternatively, the reward function can be treated as a black-box function such that we do not have access to the actual function and thus cannot calculate its true gradient.

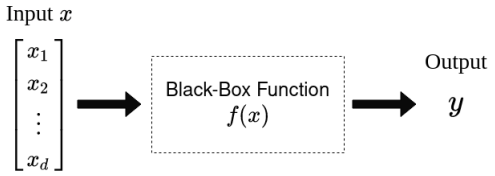


Figure 2: Point-wise evaluation in BBO

Here, we can sample the function at many points 2 and approximate the gradient using the mean and standard deviation if gradient-based RL is the desired approach. Evolutionary Strategies (see Salimans et al. 2017) is one such work in which a stochastic gradient estimate is used to update the parameters of a policy network. However, approximating gradients this way does not always produce reliable results.

Moore and J. Oliva 2022 propose a novel technique for Black-Box Optimization problems. Their method, Hypothesis-test Driven Coordinate Ascent (HDCA), combines coordinate ascent with hypothesis testing in order to learn policies in stochastic environments. HDCA perturbs a subset (block) of parameters and rolls out a number of these perturbed policies numerous times in any single iteration. The rewards from these rollouts are aggregated for each policy and the one that produces the best result with statistical significance is chosen to replace the current policy.

3 Contributions

In my research, I took a look at different AFA approaches and expanded on them in order to study whether performance is affected and can be improved. The contributions in this work are as follows:

- Generalized the scope of generative surrogate models for GSMRL techniques such that different models of arbitrary conditional distributions can be used by the agent in the AFA MDP.
- Contributed to the Active Feature Acquisition repository by formulating a modular implementation of surrogate models that follows object-oriented guidelines and design patterns.
- Proposed and implemented the use of pre-trained policy networks as the initial state for Black-Box Optimization techniques such as HDCA and ES.

4 Methods

4.1 Interchanging Surrogate Models for GSMRL

As noted earlier, GSMRL leverages information about conditional dependencies between features to inform the selection of which feature to acquire. The arbitrary conditional distribution used in Li and J. Oliva 2021, page 4, are arbitrary conditioning flow models (ACFlow) from Li, Akbar, and J. Oliva 2020. My goal was to re-run some of the same experiments that are in the paper but with a slight tweak to the method they used. In particular, we explored how the choice of surrogate model affects the evaluation performance of the learned policy. Thus, I swapped out the ACFlow code implementation of arbitrary conditionals with a different method for conditional density estimation. Namely, Arbitrary Conditional Distributions with Energy or ACE, (see Strauss and J. B. Oliva 2021).

The main challenges in making this change were two-fold:

4.1.1 Interfacing with surrogate models

The different surrogate models do not interface with the GSMRL framework in the same way as they have a varying set of functions and code structure.

$\Rightarrow$ This was resolved by creating a surrogate model wrapper class that could be used to encompass any desired implementation of a surrogate model. This solution served as the basis for what would later become a more refined abstract surrogate model class used in the Active Feature Acquisition code base.

4.1.2 Different library versions in the same environment

GSMRL is implemented in TensorFlow version 1 whereas ACE (and potentially future surrogate models) is relies on TensorFlow version 2. We quickly learned that the Python run-time environment does not allow for two versions of the same package to exist in the same environment. Nonetheless, TensorFlow 2 provides a backwards-compatible sub-package for accessing TensorFlow 1:

```
import tensorflow.compat.v1 as tf
```

However, we quickly learned that it is virtually impossible to use separate instances of both version 1 and version 2 in the same environment simultaneously, which is exactly what would need to happen in order for GSMRL to interface with the ACE model every time the agent requests side information from the surrogate model.

$\Rightarrow$ The solution was isolating the surrogate model in its own run-time process separate from that of the GSMRL training or evaluation scripts. This compromise introduced the use of `Ray`, a Python package for building distributed and parallel applications.

Ray allowed us to spin up a remote instance of the surrogate model in a separate process that loads TensorFlow version 2 all the while the main process has the backwards-compatible version 1 running. The two processes are still able to interact and transfer data between one another though function calls.

4.2 Pre-training Policy Networks before Performing Black-Box Optimization

The second Active Feature Acquisition method that we explored is formulating the AFA problem as a Black-Box Optimization problem that starts at a pre-trained policy network. The aim was to understand the advantage (if any) of pre-training a network beforehand and then optimizing its parameters using one of the BBO techniques discussed earlier such as Evolutionary Strategies or Hypothesis-test Driven Coordinate Ascent.

Irrespective of the optimization technique of choice, the policy network is pre-trained on data obtained as follows:

Algorithm 1 Generating input/output pairs for supervised learning

Input: $x, y \leftarrow$ data instance
$b \in [0, 1]^d \leftarrow$ randomly generated mask
$x_o \leftarrow$ subset of features derived from mask b
$m \in \mathbb{R}^{2d} \leftarrow [x_o, b]$
$f_\theta \leftarrow$ trained classification model on masked input
for every feature i **do**
  $p(y|x_o) \leftarrow f_\theta(x_o)$ ▷ probability distribution of output given observed features x_o
  $p(y|x_o, x_i) \leftarrow f_\theta(x_o, x_i)$ ▷ probability distribution of output given observed features plus x_i
  $I_i \leftarrow H(p(y|x_o)) - H(p(y|x_o, x_i))$ ▷ Calculating information gain
end for
$I_{d+1} \leftarrow$ threshold value for termination ▷ when greater than info gain of all other features
Output: $x_o, I \in \mathbb{R}^{d+1}$

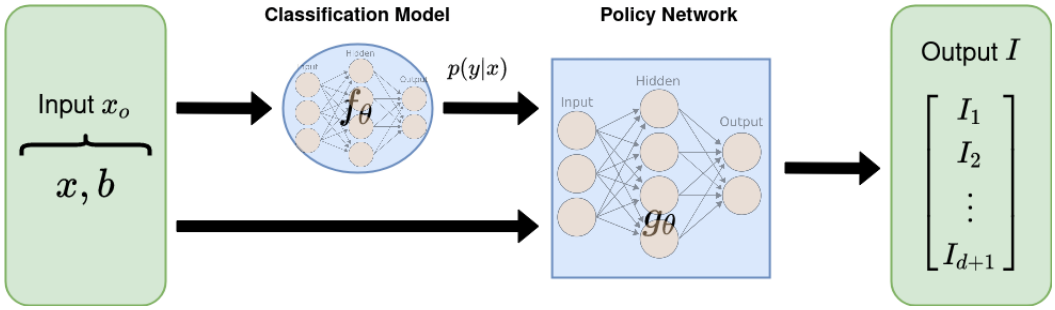


Figure 3: Pre-training policy network flow diagram

The code outlined in Algorithm 1 generates the inputs and outputs used to train a policy network, $g_\psi(x_o) \rightarrow I$, to estimate the information gain that is provided by each of the features given a particular state during evaluation. The true information gain is calculated using the following equation when training the policy network:

$$\mathbf{I} = H(p(y|x_o)) - H(p(y|x_o, x_i))$$

where **I** is the information gain for feature i, H is the entropy function, x_o are the observed features, and x_i is the missing feature in question, and $p(y|x)$ are the probability distributions of the output obtained from the pre-trained classification model.

The motivation behind predicting the information gain of each feature is that they provide a principled approach to selecting the next feature to acquire for a given instance and at a given time step. The agent can choose to take the action of acquiring the feature that is predicted to give the highest information gain among all features, thus essentially mimicking a greedy RL strategy. The termination threshold represents the value of information gain that must be exceeded by any one feature in order for the agent to acquire that feature instead of terminating and making a final prediction.

Furthermore, we extend this idea to taking the softmax of the policy network outputs and using those as the probabilities of randomly selecting the next action to take. By doing this, we can enable the agent to explore different trajectories stochastically during inference. Thus, the probability of selecting action i is:

$$p_i = \frac{exp(I_i/\tau)}{\sum_{j=1}^{d+1} exp(I_j/\tau)}$$

The temperature value, τ, passed into the softmax equation can be regarded as a hyper-parameter that is used to control the balance between agent exploration and exploitation.

The following pseudo-code describes the agent's strategy at inference:

Algorithm 2 Evaluating learned policy

Input: Pre-trained classifier and policy parameters θ, ψ
Hyperparameters: Softmax temperature τ
for every instance x in evaluation set **do**
  $b \leftarrow (0 \dots 0) \in \mathbb{R}^d$
  while true **do**
    $m \in \mathbb{R}^{2d} \leftarrow [x_o, b]$
    $I \leftarrow g_\psi(m)$
    $p \leftarrow softmax(I/\tau)$
    $a \leftarrow$ choose action based on distribution p
    if a is termination action **then**
      $pred \leftarrow f_\theta(m)$; **break**
    else
      $x_a \leftarrow$ acquire feature a; $b_a \leftarrow 1$
    end if
  end while
end for

5 Experiments

5.1 Running GSMRL with ACE model on UCI Gas Dataset

After implementing working code for the remote ACE surrogate model, we wanted to test the performance of GSMRL on an unsupervised task, also known as Active Instance Recognition (AIR). AIR is the extension of AFA in which the outputs being predicted are the values of the unobserved features (i.e. imputation of missing features).

First, we used the gas dataset, from the UCI Machine Learning Repository, modified for unsupervised learning settings. We train an ACE model on this data and then train and evaluate a Proximal Policy Optimzation (PPO) agent (Schulman et al. 2017) using the ACE model as the surrogate in the GSMRL framework. We report the performance of our model as the RMSE of the imputed unobserved features with their true value.

In addition, we train and evaluate the agent with multiple acquisition costs in order to measure performance for a range of different `average number of features acquired`. The RMSE values (shown in Figure 4) were then compared with those reported in the GSMRL paper for the same dataset (see Li and J. Oliva 2021, page 8).

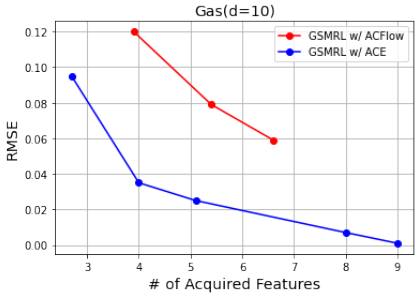


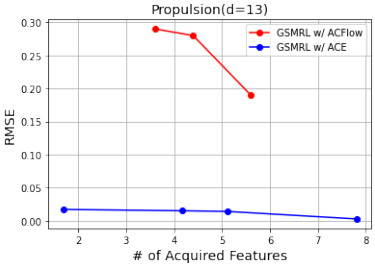
Figure 4: RMSE for Gas dataset using different surrogate models

As you can see, GSMRL with ACE outperforms its ACFlow counterpart with lower RMSE on the imputed values all throughout. These findings are in accordance with our expectation that GSMRL agent performance should improve when relying on a better performing surrogate model for its side information.

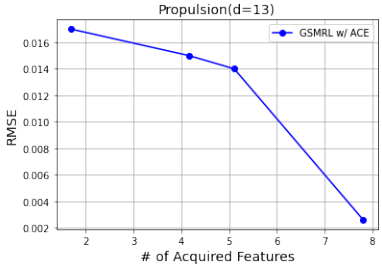
Note: these results were also reproduced later on by training a PPO agent with an ACE surrogate model using the new and improved feature acquisition code base.

5.2 Running GSMRL with ACE model on UCI Propulsion Dataset

Next, we repeat the same experiment for the propulsion dataset and also compare the different RMSE values, corresponding to training with different acquisition costs, with those reported in the GSMRL paper. The results (in Figures 5a and 5b) reaffirm the performance we saw on the gas dataset.



(a) Comparing performance using ACE with using ACFlow as surrogate model for GSMRL



(b) RMSE variation with varying number of features

Figure 5: RMSE for Propulsion Dataset

5.3 Pre-training Policy Network on CUBE Dataset

For the second AFA method explored in this work, I implemented the policy pre-training framework described in section 4.2 in Python and ran it to train a classifier and policy network for the 20-dimensional CUBE-0.3 dataset. CUBE-σ is a synthetic dataset with p-dimensional real-valued vectors and 8 classes. Only the first 10 dimensions contain the relevant information needed to classify a sample (see Shim, Hwang, and Yang 2018, pages 6-7 for more information).

The evaluation performance of the pre-trained policy network is reported in Table 1. Moreover, a short sample trajectory of the actions taken by the agent using our policy network during evaluation of a particular instance is visualized below:

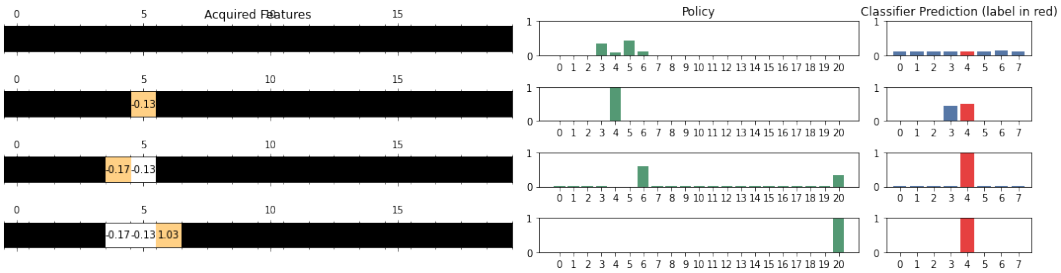


Figure 6: 4-step trajectory of agent acquisitions and prediction probabilities

Note that the red bar in the prediction histogram represents the true label for the given instance, and the yellow-colored boxes represent the agent's feature selection at any given step.

5.4 Running BBO using Pre-trained Policy Network for CUBE Dataset

After pre-training a policy network that achieved acceptable performance as is, we wanted to see if passing the learned policy through a Black-Box Optimization technique like HDCA or ES could improve its performance. As with general RL methods, the goal is to optimize a policy network's parameters to maximize some reward function (see 2.3). The reward function takes into account the cost for each acquisition as well as a cross-entropy loss for the predictions. The cost hyper-parameter is particularly relevant here as it can be used to control and optimize the number of features the agent selects on average by tweaking policy parameters that would change the termination action threshold.

With this reward function, we ran this optimization experiment using HDCA as well as ES with our pre-trained policy on the CUBE dataset. Results for both the gradient-free and gradient-based techniques are included in Table 1.

Table 1: Evaluation performance of using pre-trained policy network

Method	Accuracy	Avg. # of Features
No optimization	0.82	5.26
HDCA (10 iterations)	0.83	5.39
ES (50 iterations)	0.84	7.71

The results show that for both optimization methods, the policy was tweaked to select a higher average number of features for a 1-2% improvement in prediction accuracy. This improvement is not very substantial but acts as a proof-of-concept for optimizing AFA policies starting with a pre-trained initial set of parameters. Especially considering both algorithms were not run for many iterations compared to what is usually needed for network optimization problems. Future work could involve running HDCA or ES for more iterations as well as trying different configurations of hyper-parameters to achieve optimal results.

6 Conclusion

Active Feature Acquisition opens to door to new possibilities in the field of ML and intelligent decision systems. With new methods and techniques being developed to further advance this area of research, it is also useful to consolidate the tools and methods that are currently available and improve their performance, usability, and applicability to different disciplines.

References

Cai, Jie et al. (2018). "Feature selection in machine learning: A new perspective". In: *Neurocomputing* 300, pp. 70–79. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.11.077. URL: https://www.sciencedirect.com/science/article/pii/S0925231218302911.

Li, Yang, Shoaib Akbar, and Junier Oliva (13–18 Jul 2020). "ACFlow: Flow Models for Arbitrary Conditional Likelihoods". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5831–5841. URL: https://proceedings.mlr.press/v119/li20a.html.

Li, Yang and Junier Oliva (18–24 Jul 2021). "Active Feature Acquisition with Generative Surrogate Models". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 6450–6459. URL: https://proceedings.mlr.press/v139/li21p.html.

Ling, Charles X. et al. (2004). "Decision Trees with Minimal Costs". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, p. 69. ISBN: 1581138385. DOI: 10.1145/1015330.1015369. URL: https://doi.org/10.1145/1015330.1015369.

Moore, John Kenton and Junier Oliva (2022). *Hypothesis Driven Coordinate Ascent for Reinforcement Learning*. URL: https://openreview.net/forum?id=uoBAKAFkVKx.

Nan, Feng et al. (2014). "Fast margin-based cost-sensitive classification". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2952–2956. DOI: 10.1109/ICASSP.2014.6854141.

Salimans, Tim et al. (Mar. 2017). "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In.

Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347. arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

Shim, Hajin, Sung Ju Hwang, and Eunho Yang (2018). "Joint Active Feature Acquisition and Classification with Variable-Size Set Encoding". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2018/file/e5841df2166dd424a57127423d276bbe-Paper.pdf.

Strauss, Ryan and Junier B Oliva (2021). "Arbitrary Conditional Distributions with Energy". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., pp. 752–763. URL: https://proceedings.neurips.cc/paper/2021/file/06c284d3f757b15c02f47f3ff06dc275-Paper.pdf.