ODIN: Optimal Discovery of High-value INformation Using Model-based Deep Reinforcement Learning

Sara Zannone [1] **José Miguel Hernández-Lobato** [2 3] **Cheng Zhang** [3] **Konstantina Palla** [3]

Abstract

We consider the problem of active feature selection where we dynamically choose the set of features that acquires the highest predictive performance relative to a task. We propose a model-based deep reinforcement learning framework for Optimal Discovery of high-value INformation (ODIN) in which the agent either chooses to ask for a new feature or to stop and predict. Utilizing the ability of the partial variational autoencoder (Ma et al., 2018) the framework models the conditional distribution of the features allowing for data efficiency. We introduce a novel cost function that is sensitive to both cost and order of feature acquisition. ODIN handles missing data naturally and ensures the globally optimal solution for most efficient feature acquisition while preserving data efficiency. We show improved performance on both synthetic and real-life datasets.

1. INTRODUCTION

Feature selection manifests as a need in many real world settings. Often the goal is to discover features that are most relevant to a particular outcome, e.g. prediction (Jovic et al., 2014; Guyon & Elisseeff, 2003). Consider, for instance, the case of a medical doctor trying to diagnose their patient. Ideally, the doctor would have access to information provided by all the available tests. However, this is impractical and associated with a considerable cost (e.g. insurance costs, hospital overhead etc.) In this scenario, active feature selection is assigned with the task of sequentially choosing the subset of features (lab tests, exams, etc.) that ensures accuracy while keeping the cost low.

Traditional feature selection is not efficient in this setting. For example, different patients may require different tests, and the outcome of a first exam could influence the choice of the following medical investigations. We want to answer the question: "which information shall we acquire next in order to improve our prediction while keeping data acquisition cost low? " Automating this process is challenging, as we need to learn an optimal decision making policy for each data point/patient dynamically, assessing the information provided by the test results observed so far.

In this work, we propose ODIN, a model-based reinforcement learning framework for Optimal Discovery of high-value INformation framework. ODIN casts the dynamic feature selection task into an optimization problem using reinforcement learning (RL) (Sutton & Barto, 1998). Our main contributions are:

- An novel model based RL formulation for sequential feature selection.
- A novel state model that alleviates the problem of small sample size while naturally handling missingness.
- A reward function that enables the feature acquisition to be order sensitive while the RL agent aims for the globally optimal solution.

Our framework is general, allowing its application to many different settings where the information (features) can be of any form. We demonstrate the performance of our framework on various settings and apply it in real-world datasets.

2. Proposed formulation

We formulate the sequential feature selection following the notation used in (Ma et al., 2018). At each step, we determine the best feature x_i to select from the set of unobserved features x_U, given the observed set x_O.

We look at the feature selection task as a RL problem. We want to choose feature after feature (dynamically) in a fashion that optimizes a reward, which gives high prediction accuracy but with low cost. Considering the feature acquisitions as actions, at each time step t, the agent finds itself in the state $\mathbf{s}_t$, defined as the set of features observed so far $\mathbf{x}_{O_t}$. The agent has then to select the next feature (action) $a_t = i$, where $i \in U$ and U is the set of indices of the un-

[1]Imperial College, London, UK [2]Department of Engineering, University of Cambridge, Cambridge, UK [3]Microsoft Research Cambridge, Cambridge, UK. Correspondence to: Sara Zannone <sara.zannone@gmail.com >, Konstantina Palla <Konstantina.Palla@microsoft.com>.

Real-world Sequential Decision Making workshop at ICML 2019,

observed features. One time step later, the state changes to $\mathbf{s}_{t+1} = \mathbf{x}_{O_{t+1}} = \mathbf{x}_{O_t} \cup x_i$ and the agent receives a reward r_{t+1} for its action.

We further inspire the design of the framework from the challenges we want to address. *Scarcity*, *missingness* in data and the *optimal feature acquisition sequence* are our drives. In many real world applications, we are dealing with data that are scarce, hard or costly to find. For instance, in many healthcare applications, finding enough training examples is a challenge, e.g. think of rare diseases case. The small dataset size often has an impact on the performance of the machine learning approaches. Moreover, one of the key problems often faced in data is *missing values*; using the doctor diagnosis process as an example again, at any point they only observe a small subset of the patient exams (measurements) and yet they have to reason about possible values for the remaining. We thus need a generative model as a component of the RL framework that can simulate the data mechanism but also impute the missing values given a variable subset of those. Further, we acknowledge the importance of acquiring the features in a specific sequence found in many settings. For instance, a doctor's decision for the next test depends on the result of the ones already undertaken. There is a importance in the feature sequence dependence. Thus, we introduce a novel sequence dependent reward function.

The introduction of these components to the RL framework constitutes our proposed methodology for the feature selection task. We now describe these two components along with the policy network.

The generative model G models the distribution $\mathcal{P}^{x_i}_{\mathbf{x}_t\mathbf{x}_{t+1}} = P[\mathbf{x}_{t+1}|\mathbf{x}_t, x_i]$, i.e. the conditional distribution of the feature x_i decided to be chosen next given the current observed features $\mathbf{x}_t$. The generative model must be able to work with missing data, something that few existing approaches for generative modeling can do. We use the Partial Variational Autoencoder PVAE (Ma et al., 2018) that learns distributions even when missingness is present. Learning these distributions, the generative model augments the framework with the ability to simulate additional training data and thus alleviates the problem of small sample size.

The reward network R returns the immediate reward associated with the choice of the feature at each step, i.e. $\mathcal{R}^{x_i}_{\mathbf{x}_t} = f(\mathbf{x}_t, x_i)$. The choice of the function is crucial as it determines how the agent "ought" to behave. We design the reward to account for both the merit of choosing the feature x_i to extend the feature sequence at step t but also for the cost associated with acquiring an additional feature. Given the observed features $\mathbf{x}_{t+1} = [\mathbf{x}_t, x_i]$, we define the loss function $\mathcal{L}(f(\mathbf{x}_{t+1}), y_{true})$ that quantifies the accuracy of the prediction given by the features observed at time $t+1$. We also introduce the acquisition cost value c_i. At every time-step t, we have

$$R_t = \mathcal{L}(f(\mathbf{x}_{t+1}, y_{true}) - n_t \cdot c_i, \tag{1}$$

where n_t is the total number of features already observed at step t. The function $f(.)$ can be approximated by any classification or regression algorithm. Similar to the generative network, the classifier/regressor should be able to handle inputs of variable length. For this reason, we use the PointNet deepNet architecture by (Qi et al., 2016) also used as the encoder part of the PVAE in (Ma et al., 2018).

Assigning key importance to the order which the features are acquired, we introduce the reward function in Equation 1. It allows for a feature-order sensitive and cost effective acquisition; the first term quantifies the merit of choosing the feature at time $t+1$, while the cost term strategically forces for order in the acquisition by keeping the the feature collection cost low. In this case, ordering the features acquired can get the most useful information first, while providing predictions on partial information and minimizing the cost of the data collection. To the best of our knowledge, we are the first ones to use such a reward function in a RL framework for feature selection.

The policy network The policy defines the agent's way of behaving at a given time by mapping states to actions to be taken in those states. In our framework, the policy is stochastic, i.e. models a distribution over actions (feature selections) and draws an action according to this distribution. The policy network is implemented as a deep neural network following again the PointNet architecture so that states of variable length are stochastically mapped to actions (features). The parameters of the network during policy learning are optimized using the Proximal Policy Optimization algorithm (PPO, Schulman et al., 2017).

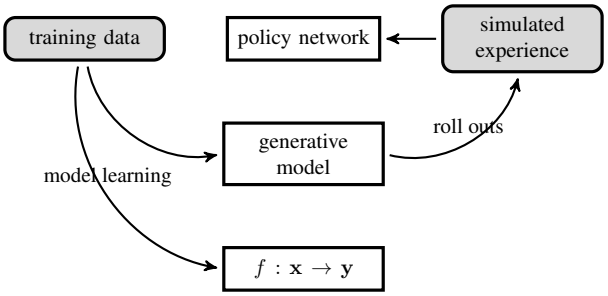

Figure 1. Diagram of the proposed RL framework for dynamic feature acquisition.

The cartoon diagram in Figure 1 and the description in Algorithm 1 provide an overview of how the different components are combined. During training, both the generative model and the reward network are pretrained on the training

Algorithm 1 Proposed algorithmic framework

Require: Partially observed training data $\mathbf{D}_{tr} = \{\mathbf{x}_{tr}, \mathbf{y}_{tr}\}$; Fully unobserved dataset $\mathbf{D}_{tst} = \{\mathbf{x}_{tst}, \mathbf{y}_{tst}\}$
1: **Train Partial VAE and classifier** on $\mathbf{D}_{tr}$.
2: **Roll out** $\mathbf{D}_{roll} = \{\mathbf{x}_{roll}, \mathbf{y}_{roll}\}$; $\mathbf{x}_{roll} \sim \mathcal{N}(g(\mathbf{z}; \phi), \sigma^2 * I)$ and $\mathbf{y}_{roll} \sim f_\theta(\mathbf{x}_{roll})$.
3: **Learn the policy** on $\mathbf{D}_{roll}$ using PPO.
4: **Sequentially acquire feature value** x^i to estimate y for each test point in $\mathbf{D}_{tst}$.
for each test instance **do**
  $\mathbf{x}_O \leftarrow \emptyset$
  repeat
    Choose to observe feature i according to policy
    $\mathbf{x}_O \leftarrow x^i \cup \mathbf{x}_O$
  until agent decides to stop or all features are observed
end for

dataset. The framework moves on with simulating experience by sampling roll-outs from the generative model and add them to the original training dataset resulting in an augmented one. The agent is then trained on the augmented dataset. During test time, for each instance, the agent infers the feature set sequentially using the learnt policy; after the policy network recommends to observe a particular feature, that feature is observed and the state is extended to include the new feature. Then, we call the policy network again to repeat the whole process.

3. Related Work

There is a plethora of works in the literature related to the dynamic feature selection (Melville et al., 2004; Saar-Tsechansky et al., 2009). However, these work actively selected training set instead of applying it at test time.

We propose a general framework for efficient sequential discovery of information in a *variable-wise* fashion using a reinforcement learning architecture which can be used in test time. Only few works have addressed this problem before. The closely related work include Ma et al. (2018); Shim et al. (2018); Janisch et al. (2017). Ma et al. (2018) uses Bayesian experimental design sequentially for optimal decision making at each step. It is a greedy method and does not ensure the global optimality of the decision sequence.

Another closely related work is the work by (Shim et al., 2018; Janisch et al., 2017) which deploy a RL agent to sequentially decide on the feature to choose. It also allows for a cost-sensitive feature acquisition but not order sensitive. Their method does not provide an optimal solution at each time-step of the sequence. On the contrary, we learn a policy under a reward function designed to be sensitive to the ordering of variables chosen. Such a property is crucial in decision making settings when the decision to act next depends on the previous actions. Moreover, our framework introduces roll outs from a generative model and thus accounts for improved data efficiency.

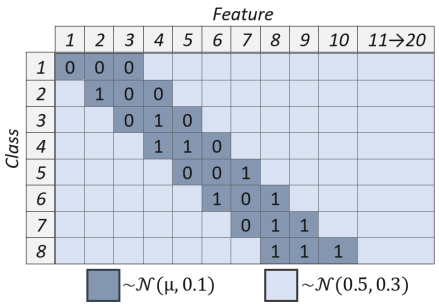

Figure 2. The Cube dataset. Data points are 20-dimensional vectors and belong to one of 8 possible classes. All features follow a Normal distribution, but only three of them are informative of the class.

4. Experiments

We evaluate the proposed framework under various settings. Our major focus is on evaluating the performance in data efficiency tasks and tasks where the sequence of the features acquired so far affects the performance. ODIN shows clear improvement comparing to greedy method such as Ma et al. (2018), as well as tradtional RL method where the order for the decision sequence is not considered (Shim et al., 2018).

4.1. Synthetic Dataset

We first show the performance of the framework on a synthetic toy dataset to demonstrate ODIN in a controlled environment. The dataset is inspired by the CUBE dataset (Rückstieß et al., 2013; Shim et al., 2018). It consists of data points that belong to one of eight possible classes. Each data point (instance) is composed of 20 features, but only 10 of them carry information useful for classification. Additionally, a different subset of features is informative for different classes. Figure 2 shows this dataset.

The task we are addressing here is classification. We considered three experiments focusing on the performance of the approach on different data sizes (data efficiency), percentage of missing features and use of order sensitive or not reward function. The first two aim to investigate the contribution of the generative model component, while the last one explores the reward function. For the last one, we considered the reward function as given in Equation 1, while we chose a reward with removed ordering sensitivity, i.e.

$$R = \begin{cases} -c, & t \neq T \\ \log P(c_{true}|f_{1:t}), & t = T \end{cases} \tag{2}$$

The acquisition cost is $c = 0.05$ unless otherwise specified.

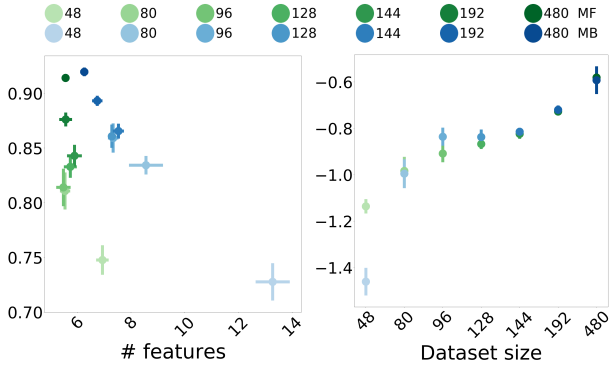

Figure 2. (a) Accuracy *Figure 2.* (b) Average return

Figure 3. Comparison of the model based and model free implementation of the proposed architecture with different dataset sizes.

Data efficiency We compare the performance of our agent when trained: i) straight on the original training dataset $\mathbf{x}_{tr}$ without the use of samples from the generative model (*model-free*) and ii) on the augmented training dataset after sampling roll-outs from the generative model (*model-based*). We consider a number of training datasets with different sizes. Both approaches use the same number of test data points after the policy is learnt.

Figure 3(a) shows the accuracy and the average number of chosen features acquired by each agent, while Figure 3(b) shows the corresponding average return as a function of the training dataset size. We see that the PVAE can successfully model the state and make ODIN select the useful feature as desired without requiring massive amount of training data. The generative model is particularly advantageous for medium-sized datasets (96 and 128 datapoints in this example). As the size of the original training dataset increases, the performance difference between the two frameworks shrinks. Last, in the regime of the 48 points, while the accuracy achieved by the two frameworks is comparable, the model-based does worse in the average return and acquires a larger number of features on average indicating that the PVAE cannot learn very efficiently in this small regime.

Missing data In this experiment we analyse the contribution of the generative model component by introducing different proportions of missingness in the training dataset.

Figure 4(a) presents the accuracy achieved by the model-based framework for different proportions of missingness and the associated average number of returned features in the test points. The agent's performance appears quite robust to the introduction of increasing missingness with the achieved accuracy remaining high.

Feature-ordering sensitive reward We now explore how the use of the proposed reward (Equation 1) affects the performance of the model-based agent. We modify the

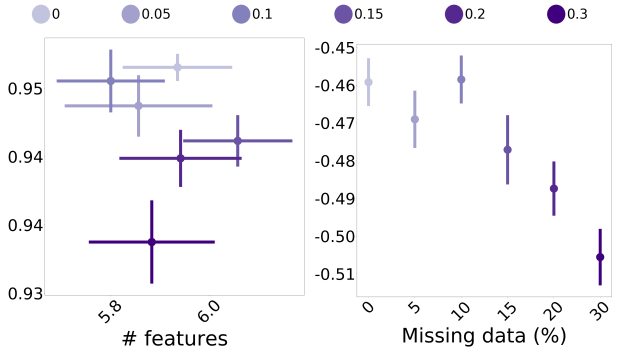

Figure 3. (a) Accuracy *Figure 3.* (b) Average return

Figure 4. Comparison of the model based implementation of the proposed architecture for different proportions of missingness in the Cube dataset.

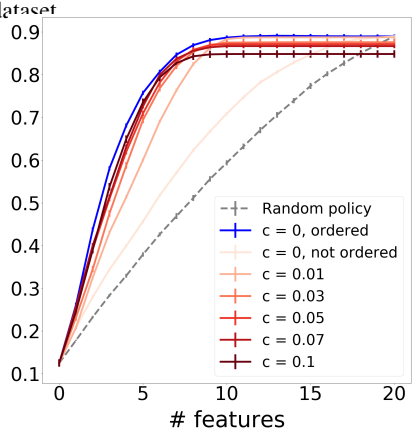

Figure 5. Comparison of ODIN reward and zero acquisition cost and with reward used in (Shim et al., 2018) with different choices of acquisition cost values.

Cube dataset so that each one of the informative features has a different level of signal-to-noise ratio. Since the features have different degrees of informative power, the order of acquisition becomes relevant.

Figure 5 shows the average predicted probability of the true class as achieved by the different framework settings. Our proposed order-sensitive reward performs best. The plot achieved by the agent with the order-sensitive reward and zero cost appears as an upper threshold to the performance achieved by the agents with no order sensitivity and different acquisition cost value choices. The agent with no order-sensitivy in its reward function, starts with a low achieved probability of the true class but by increasing the cost value, the performance increases as well. Although no order sensitive reward is used, the component of the cost presses the agent to choose fewer features. This leads the agent to select the more informative features, thus implicitly imposing ordering. However, even for high cost values, it never surpasses the performance of the agent with explicit order-sensitive reward function and zero cost value.

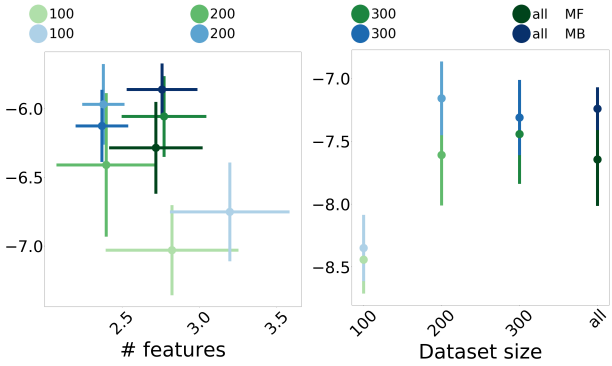

Figure 5. (a) Absolute error *Figure 5.* (b) Average return

Figure 6. Performance plots of the model-based and model-free frameworks on the Boston dataset.The plots display (a) absolute error and (b) return value (negative absolute error).

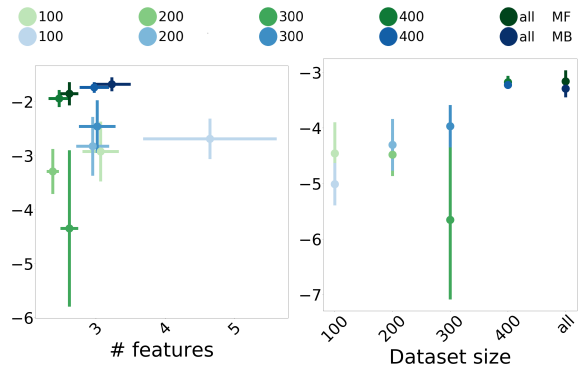

Figure 6. (a) Absolute error *Figure 6.* (b) Average return

Figure 7. Performance plots of the model-based and model-free frameworks on the Energy dataset.

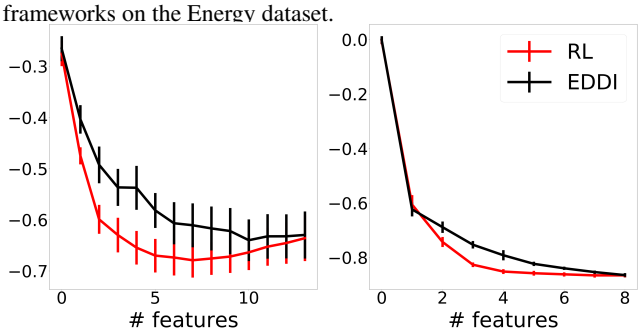

Figure 7. (a) Boston *Figure 7.* (b) Energy

Figure 8. Information curves of dynamic variable selection on the two UCI datasets. The plots display negative test log likelihood (y axis, the lower the better) during the course of dynamic feature acquisition (x-axis) for the two frameworks.

4.2. UCI datasets

We next apply our framework to regression tasks using two datasets from the UCI repository (Dheeru & Karra Taniskidou, 2017), i.e. Boston housing and energy efficiency dataset. We investigate the data efficiency property of our framework and to challenge the RL framework approach against the greedy approach.

Data efficiency We follow a similar experimental setting as in the case of the CUBE dataset and compare the performance of the proposed model-based framework and the model-free counterpart (see subsection 4.1). Once again, we use the Partial VAE as the generative model component. We use a PointNet as the regressor $f(.)$ in our framework. As before, we consider a number of training datasets of different sizes. We focus on the effect of the model component and choose the no order-sensitive reward:

$$R = \begin{cases} -c & , t \neq T \\ -|y_{true} - \hat{y}| & , t = T \end{cases} \tag{3}$$

where cost c set to 0.005.

Figures 6(a) and 7(a) show the accuracy for the two datasets and the average number of acquired features, while Figure 6(b) and Figure 7(b) show the corresponding average returned reward value on the test points for the different training dataset splits. In these applications, the proposed model-based framework outperforms the model-free approach with a significant advantage on the small training size regimes. The error bars are computed using 5 runs.

Global comparison with EDDI Lastly, we compare our framework to a greedy feature selection approach. For this, we chose EDDI (Ma et al., 2018). Both frameworks use Partial VAE to learn the generative distribution of the data. In order to make a fair comparison with EDDI, we train the same PVAE on the training set including the label. We also use the same PVAE as regressor $f(.)$ in our framework. For EDDI, the sequence of features is then chosen following a greedy method based on mutual information (Ma et al., 2018). In our framework, we train a model-based agent to maximise the average log likelihood, as predicted by the PVAE, at every timestep. It is worth noting that, in order to make the comparison with EDDI fair, our classifier is effectively given by the PVAE itself. For fair comparison, we also remove the STOP action and force the agent to always acquire all features. ODIN finds a globally superior solution comparing to EDDI.

5. Conclusion

In this paper, we present ODIN, a novel and efficient architecture for dynamic active feature selection using the Reinforcement Learning framework. The proposed architecture allows for the use of a generative model component that augments the training dataset and handles missing data. We also empower the feature acquisition of the agent with a novel order-sensitive reward function. The framework has demonstrated its effectiveness on feature acquisition tasks

including real-world applications.

A. Architecture and training details

We describe the architecture of the three components, i.e. the PVAE, the regressor/classification (reward) network and the policy network.

The implementation has the same basic architecture for all components. The initial layers are a PointNet network as in (Ma et al., 2018) that maps an unordered set to an embedding of dimensionality K = 20. The PointNet then feeds into an encoder network of dimensionality K-500-200 with ReLU activations. The last layer changes depending on the component; for the reward network, it has either a sigmoid or a linear activation function, for classification and regression respectively. For the policy network, the last layer has two outputs: i) a sigmoid activation function that gives the probability of taking each one of the available actions (given the input state) and ii) a linear layer that outputs the value of the input state. For the PVAE, the last layer outputs the parameters of a diagonal Gaussian latent variable. The decoder part of the PVAE is a network of dimensionality 50-100 for UCI and 500-200 for CUBE. The last layer of the decoder estimates the mean of a gaussian for UCI, and mean and variance for CUBE.

The training of the framework consists of two steps: First, we train the PVAE and the classifier/regressor. Both components are trained on the original data. Second, we train the policy network on the roll outs.

We train the PVAE using the variable of interest, i.e. class or dependent variable (target variable), as part of the feature vector. During training the PVAE and the PointNet implementations of classifier/regressor, we introduce missingess at random in the training dataset. We use a uniform distribution $\mathcal{U}(0, 1)$ to sample a missing rate parameter and choose features randomly as unobserved.

The roll outs are sampled from the PVAE by randomly dropping features but with an increasing missing rate $\in (0, 1)$.

B. Experiments

Here we provide more details on experiments conducted in the main paper.

B.1. Synthetic dataset experiments

Data efficiency We compare the performance of the model free and model based frameworks on different splits (sizes) of synthetic dataset sizes. The sizes shown in Figure 3, i.e. 48, 80, 96, 128, 144, 192 and 480, refer to the training set size. The validation sets were taken to be always half the size of the training set, i.e. 24, 40, 48, 64, 72, 96 and 240. The test set was composed of 1000 data points and was the same for all runs. After training the PVAE, we sampled (roll-outs) 10K data points. Each result reported in Figure 3 is an average over 5 runs.

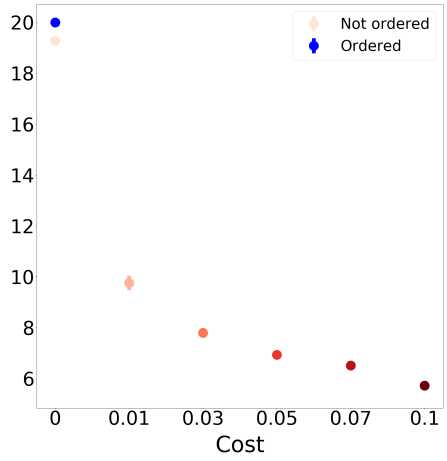

Figure 9. Average number of features chosen as a function of cost for ordered (blue) and unordered (red) reward functions in the Boston.

Missing data We used a dataset of N = 4K datapoints from Cube and split into 2000/1000/1000 for train/validation/test. Numbers plotted in Figure 4 are an average over 5 runs.

Reward Here, the dataset was slightly modified such that informative features had different signal to noise ratio. The three informative features of each class (2) were set to noise levels $\sigma_{low} = 0.05$, $\sigma_{medium} = 0.3$ and $\sigma_{high} = 0.5$. Here we generated N = 1000 CUBE data points making sure we keep class balance. Again, we split in 500/250/250 for train/validation/test. Figure 5 plots the averaged (over the test points) predicted probability of the true class averaged (error bars plotted) over 5 runs. As we introduce cost terms in the reward functions, the different settings, i.e. different costs, force the agents to stop after having acquired different number of features. Each value at each feature step index in Figure 10 is the probability of the true class as averaged over all the test points that have at least that many features acquired by the agent. In Figure 5 in the main paper, the test points with less features than the current feature index are filled in with the probability of the true class returned in the last acquired feature.

Figure 9 reports the average (over the test points) number of features found by the different agents in the related figure 5 in the main paper.

B.2. UCI experiments

We ran experiments on 2 UCI datasets (Dheeru & Karra Taniskidou, 2017); Boston housing and energy ef-

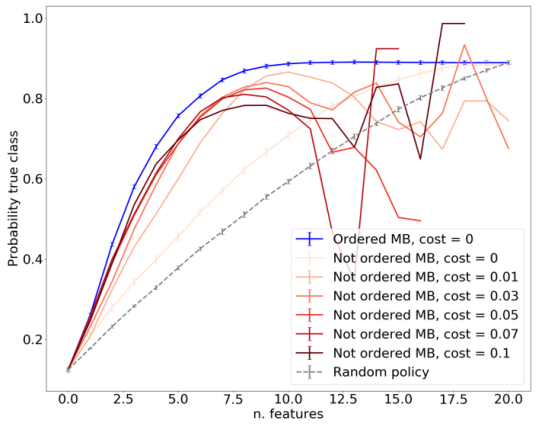

Figure 10. Comparison of ODIN reward and zero acquisition cost and with reward used in (Shim et al., 2018) with different choices of acquisition cost values and without filling in. Reported here is the predicted probability of the true class.

ficiency. The task here is regression. The size of the two datasets and their complete feature set is 506/13 and 768/8 respectively. All experiments are ran for 5 repetitions and results are reported as the average.

Data efficiency As regressor we use a PointNet deepNet that handles inputs of varialbe length. For both datasets and for all splits, the train/validation/split percentage is 80/10/10.

Global comparison with EDDI To make the comparison with EDDI fairer, we use as regressor (provides the reward) the trained Partial VAE itself. We also remove the STOP action and force the agent to always acquire all features.

The reward returned is an average log likelihood, as predicted by the Partial VAE, at every timestep. The average log likelihood is calculated using 100 samples of $\mathbf{x}_\phi \sim p(\mathbf{x}_\phi|\mathbf{x}_O)$ through $p(\mathbf{x}_\phi|\mathbf{x}_O) \approx \frac{1}{M}\sum_{m=1}^{M} p(\mathbf{x}_\phi|\mathbf{z}_m)$, where $\mathbf{z}_m \sim q(\mathbf{z}|\mathbf{x}_O)$ (Ma et al., 2018). The variables of interest $\mathbf{x}_\phi$ are chosen to be the target variables of each UCI dataset in the experiment. During test time and for the plotting of the information curves in Figure 8, the negative test log likelihood of the target variable is also estimated using 100 samples.

References

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435.

Janisch, J., Pevný, T., and Lisý, V. Classification with costly features using deep reinforcement learning. *CoRR*, abs/1711.07364, 2017. URL http://arxiv.org/abs/1711.07364.

Jovic, A., Brki, K., and Bogunovic, N. An overview of free software tools for general data mining. pp. 1112–1117, 05 2014. ISBN 978-953-233-077-9. doi: 10.1109/MIPRO.2014.6859735.

Ma, C., Tschiatschek, S., Palla, K., Hernández-Lobato, J. M., Nowozin, S., and Zhang, C. EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. *ArXiv e-prints*, September 2018.

Melville, P., Saar-Tsechansky, M., Provost, F., and Mooney, R. Active feature-value acquisition for classifier induction. In Rastogi, R., Morik, K., Bramer, M., and Wu, X. (eds.), *Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004*, Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004, pp. 483–486, 12 2004.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL http://arxiv.org/abs/1612.00593.

Rückstieß, T., Osendorfer, C., and van der Smagt, P. Minimizing data consumption with sequential online feature selection. *Int. J. Machine Learning Cybernetics*, 4:235–243, 2013.

Saar-Tsechansky, M., Melville, P., and Provost, F. Active feature-value acquisition. *Management Science*, 55(4): 664–684, 2009.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Shim, H., Hwang, S. J., and Yang, E. Joint active feature acquisition and classification with variable-size set encoding. In *Advances in Neural Information Processing Systems*, 2018.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.