A. Policy Invariance under Intermediate Rewards

Assume the original Markov Decision Process (MDP) without the intermediate rewards is defined as $M = (S, A, T, \gamma, R)$, where S and A are state and action spaces, T is the state transition probabilities, γ is the discount factor, and R is the rewards. When we introduce the intermediate rewards R_m, the MDP is modified to $M' = (S, A, T, \gamma, R')$, where $R' = R + R_m$. The following theory provides a sufficient and necessary condition for the modified MDP M' to achieve the same optimal policy as the original MDP M.

Theorem 1. *The modified MDP $M' = (S, A, T, \gamma, R + F)$ with any shaping reward function F is guaranteed to be consistent with the optimal policy of the original MDP $M = (S, A, T, \gamma, R)$ if the shaping function F have the following form*

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s), \tag{A.1}$$

where $\Phi : S \to \mathbb{R}$ is a potential function evaluated on states. For infinite-state case (i.e., the state space is an infinite set) the potential function is additionally required to be bounded.

Proof. Please refer to Ng et al. (1999) for detailed proof. □

From the above theorem, we can see our intermediate rewards in (3) is a potential based shaping function and the potential function is $\Phi(s) = -H(y \mid s)$. For classification task where $y \in \mathcal{Y} = \{1, 2, \ldots, K\}$ is a discrete variable, the entropy is naturally bounded, i.e., $0 \leq H(y \mid s) \leq log|\mathcal{Y}|$, where $|\mathcal{Y}|$ is the cardinality of the label space. For regression task where $y \in \mathbb{R}$, the entropy is bounded by $0 \leq H(y \mid s) \leq H(y \mid \emptyset)$. The upper bound $H(y \mid \emptyset)$ is determined by the given surrogate model. Similarly, for the intermediate rewards in (10), the potential function $\Phi(s) = \frac{\log p(x_u|s)}{|u|}$ is also bounded for a given surrogate model.

B. Experiments

B.1. Classification

For classification tasks, we conduct experiments on MNIST and two UCI datasets. We downsample the MNIST images to 16×16 to reduce the total number of features in order to accommodate baselines such as (Ma et al., 2018), which had trouble scaling. Features are normalized into the range $[0, 1]$.

The surrogate model for classification task estimate arbitrary conditional distributions that are conditioned on the target variable y. For MNIST, we stack conditional coupling transformations and a conditional Gaussian likelihood module. For UCI dataset, we use an autoregressive likelihood module. To train the surrogate model, we randomly select two non-overlapping subsets u and o and optimize the arbitrary conditional log likelihood

$$\begin{aligned}\log p(y, x_u \mid x_o) &= \log p(x_u \mid x_o) + \log P(y \mid x_u, x_o) \\ &= \log p(x_u \mid x_o) + \log \frac{p(x_u, x_o \mid y)P(y)}{\sum_{y'} p(x_u, x_o \mid y')P(y')}.\end{aligned} \tag{B.2}$$

The agent is implemented as a PPO policy. Given the current state x_o and the auxiliary information from the surrogate model, we extract a set embedding using set transformer (Lee et al., 2019). The inputs are first transformed to sets by concatenating with the one-hot encoding of their indexes. The set embedding is beneficial to deal with arbitrary dimensionality of the inputs. The policy network then takes the set embedding as inputs and outputs the next action. The critic network takes the same set embedding as inputs and output an estimate of the state values. To help the agent extract meaningful representations from its inputs, we let the prediction model f_θ take the same set embedding as input. The policy network, the critic network and the prediction function are all implemented as fully connected layers.

We run the baseline model JAFA (Shim et al., 2018) using their public code. We cross-validate the optimal architecture by modifying the number of layers and the size of each layer for both the agent and the classifier.

We adapt EDDI (Ma et al., 2018) to perform classification task by modifying the decoder to output Categorical distribution for y and Gaussian distribution for x. EDDI learns the distribution $p(y, x_o)$ by utilizing a VAE based model. The acquisition metric for EDDI is

$$\begin{aligned}\mathcal{U}_i =& \mathbb{E}_{x_i \sim p(x_i|x_o)} D_{\mathrm{KL}}[p(z \mid x_i, x_o) \| p(z|x_o)] \\ &- \mathbb{E}_{y, x_i \sim p(y, x_i|x_o)} D_{\mathrm{KL}}[p(z \mid y, x_i, x_o) \| p(z \mid y, x_o)],\end{aligned} \tag{B.3}$$

which is estimated using the proposal distribution. Then, a greedy policy that acquires the feature with maximum utility is employed. We similarly cross-validate the architecture for each dataset.

We also compare to a greedy policy using the surrogate model where the utility is calculated by (8). At each acquisition step, the one with maximum utility is selected.

B.2. Regression

For regression task, the target variable y is concatenated into the features x and the surrogate model learns the distribution $p(y, x_u \mid x_o)$. The agent is similarly implemented as the PPO policy with a set transformer based feature extractor. Baseline models include JAFA and EDDI, where the architecture is selected by cross validation. We also build a

Algorithm 2 Active Feature Acquisition with GSMRL

1. load pretrained surrogate model M, agent *agent* and prediction model $f_\theta(\cdot)$
2. instantiate an environment with data D: *env* = Environment(D)
3. x_o, o = *env*.reset(); // $o = \emptyset$
4. done = False; reward = 0

while *not done* **do**

aux = M.query(x_o, o); // query M for auxiliary information
// aux contains the prediction $\hat{y} \sim p(y \mid x_o)$ and output likelihoods,
// the imputed values $\hat{x}_u \sim p(x_u \mid x_o)$ and their uncertainties,
// and estimated utilities $\mathcal{U}_i$ for each $i \in u$ ((4)).
action = *agent*.act(x_o, o, aux); // act based on the state and auxiliary info
r_m = M.reward(x_o, o, action); // calculate intermediate rewards with the surrogate model
x_o, o, done, r = *env*.step(action); // take a step based on the action
// if action indicates termination: done=True, r=$-\mathcal{L}(\hat{y}(x_o), y)$
// else: done=False, r=$-\alpha\mathcal{C}(action)$, $o = o \cup action$
reward = reward + r + r_m; // accumulate rewards

end

prediction = *agent*.predict(x_o, o, aux); // make a final prediction
// using either M.predict(x_o, o, aux) or f_θ(x_o, o, aux) based on validation

greedy policy using our surrogate model by estimating the utility following (6). For GSMRL and JAFA, the reward for a prediction $\hat{y}$ is calculated as the negative MSE $-\|\hat{y} - y\|_2^2$.

B.3. Medical Diagnosis

We evaluate our model on Physionet challenge 2012 dataset (Goldberger et al., 2000). We first preprocess the dataset by removing some non-relevant features (such as patient ID) and eliminating the instances with very high missing rate (larger than 80%). The features are then normalized to the range of [0,1]. The model and baselines are mostly the same as the classification experiments for UCI dataset. Since the classses are heavily imbalanced, we use weighted cross entropy as loss and reward. To evaluate the performance for data with missing entries, We first impute those missing features with our GSM model. For EDDI, the missing entries are similarly imputed by the VAE model. JAFA does not have a generative component, thus we simply replace the missing features with zeros. JAFA reports the AUC score for this dataset, but AUC is known inappropriate for imbalanced classification (Brabec & Machlica, 2018). We instead report the F1 scores for this experiment.

B.4. Time Series

Acquiring features for time series data requires the agent to integrate chronological constraints into the action space. For RL based approach, we manually set the probabilities of invalid action to zeros. For greedy approach, inspired by Thompson sampling (Thompson, 1933; Russo et al., 2017), we employ a prior distribution to encode our chronological constraint. Specifically, we set the prior as a Dirichlet distribution that is biased towards the selection of earlier time steps:

$$\pi(\rho) = \text{Dir}\left[\alpha(T - (\max(o) + 1)), \ldots, \alpha(T - (T - 1))\right](\rho), \tag{B.4}$$

where α is a hyperparameter, T is the total time steps, $\max(o)$ represents the latest time step already acquired, and ρ is a distribution for acquisition over the remaining future time steps. However, we still desire that the acquired features are informative for target y. Hence, we update the prior to a posterior using time steps V that are drawn according to how informative they are:

$$p(V_n = t) \propto \exp(I(x_t; y \mid x_o)), \quad t \in \{\max(o) + 1, \ldots, T - 1\},\ n \in \{1, \ldots, N\}, \tag{B.5}$$

where N is the number of samples. Due to conjugacy, the posterior is also a Dirichlet distribution

$$p(\rho \mid V) = \text{Dir}\left[\alpha(T - (\max(o) + 1)) + \sum_{n=1}^{N} \mathbb{I}\{V_n = \max(o) + 1\}, \ldots\right](\rho). \tag{B.6}$$

Samples from posterior represent the probabilities of choosing each candidate, which now prefer both earlier time steps and informative features. We draw a sample from posterior and select the most likely time step at each acquisition step.

B.5. Unsupervised

To perform active feature acquisition on unsupervised tasks, a.k.a, active instance recognition, we modify the reward for

prediction as the negative MSE of the unobserved features, i.e., $-\|\hat{x}_u - x_u\|_2^2$, where $\hat{x}_u$ is the imputed values of the unobserved features.

The JAFA is adapted to this task by changing the classifier to an auto-encoder like model, where the observed features x_o are encoded to predict the unobserved features x_u.

For EDDI, by plugging $y = x$ into (B.3), we have the acquisition metric for this setting as

$$\mathcal{U}_i = \mathbb{E}_{x_i \sim p(x_i|x_o)} D_{\mathrm{KL}}[p(z \mid x_i, x_o) \| p(z|x_o)], \quad \text{(B.7)}$$

since the second KL term in (B.3) equals to zero.

To build a greedy policy using our surrogate model, we estimate the utility using (9). Monte Carlo estimation is utilized to estimate the entropy.

C. Hyperparameters

We search the hyperparameters for both our GSMRL and baselines using cross-validation. The range of the hyperparameters is listed in Table C.1.

Table C.1. Hyperparameters for GSMRL and baselines.

GSMRL	set transformer	$\{32, 64\} \times \{1, 2\}$
	set embedding size	$\{32, 64\}$
	policy network	$\{32, 64\} \times \{2, 3\}$
	critic network	$\{32, 64\} \times \{2, 3\}$
	prediction network	$\{64, 128\} \times \{2, 3\}$
	advantage λ	0.95
	discount factor γ	0.99
	PPO clip range	$[0.8, 1.2]$
	entropy coefficient	0.0
JAFA	set embedding size	$\{16, 32, 64, 128\}$
	Q network	$\{16, 32, 64, 128\} \times \{2, 3, 4, 5\}$
	prediction network	$\{16, 32, 64, 128\} \times \{2, 3, 4, 5\}$
EDDI	set embedding size	$\{10, 20, 50, 100\}$
	encoder	$\{32, 64, 128, 256\} \times \{3, 4, 5, 6\}$
	latent code	$\{10, 20, 50, 100\}$
	decoder	$\{32, 64, 128, 256\} \times \{3, 4, 5, 6\}$

D. Additional Results

Due to the space limit, we only show one example for the acquisition process in the main text. Figure D.1 and D.3 show some additional examples for AFA and AIR tasks respectively. In Fig. D.2 and D.4, we present several examples of the acquisition process from the greedy policy. Note that the predictions for both the greedy and the non-greedy policy are from the same pretrained arbitrary conditioning model, therefore the only difference is the acquired features. Comparing the greedy and the non-greedy policy suggests that the non-greedy policy eliminates the prediction uncertainty much faster than the greedy one.

In Fig. 4 and 5, we present the acquired features from our GSMRL for several testing examples. To better understand the overall distribution of the acquired features across all the testing instances, we plot the frequencies of each feature being acquired in Fig. D.5 and D.6 for both AFA and AIR on MNIST respectively. A higher value of the frequency means the corresponding feature is acquired for more testing instances. Specifically, the frequency for a feature equals to one means the corresponding feature is a common feature acquired for all testing instances. The frequency loosely represents the importance of each feature, which could help with model interpretation and reasoning about decision making. We will explore this direction in future works.

In Fig. D.7, we analyse the sensitivity of our model to random initialization by running our model three times independently with different random seeds. We report the mean and standard deviation for both the number of acquisitions and the task performance. Baseline performance are presented for reference. We can see that our model is robust to random initialization and performs consistently better than baselines.

Small vs. Large Action Space For the sake of comparison, we employ a downsampled version of MNIST in the experiment section. Here, we show that our GSMRL model can be easily scaled up to a large action space. We conduct experiments using the original MNIST of size 28×28. We observe that JAFA has difficulty in scaling to this large action space, the agent either acquires no feature or acquires all features. The greedy approaches are also hard to scale, since at each acquisition step, the greedy policy will need to compute the utilities for every unobserved features, which incurs a total $O(d^2)$ complexity. In contrast, our GSMRL only has $O(d)$ complexity. Furthermore, with the help of the surrogate model, our GSMRL is pretty stable during training and converges to the optimal policy quickly. Fig. D.8 shows the accuracy with a certain percent of features acquired. The task is definitely harder for large action space as can be seen from the drop in performance when the agent acquires the same percentage of features for both small and large action space, but our GSMRL still achieves high accuracy by only acquiring a small portion of features.

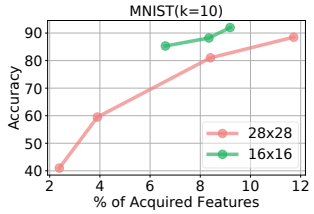


Figure D.8. Acquisition with large action space.

Reward Evaluation Since we are dealing with a dynamic acquisition scenario, different algorithms or the same algorithm with different hyperparameters, such as α, could lead to different acquisitions, which renders the direct comparison difficult. In the experiment section, we compare different algorithms by plotting the performance curve w.r.t.

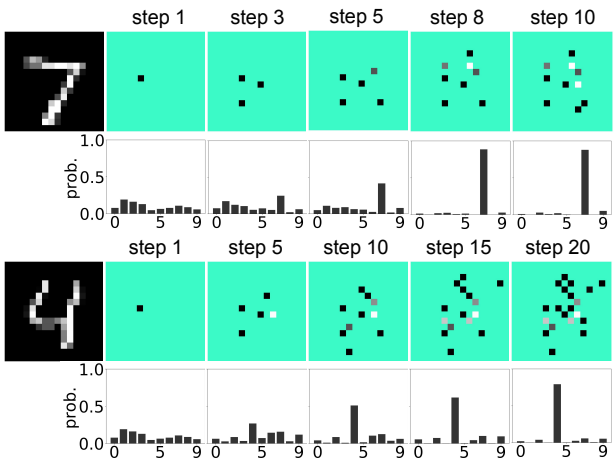


Figure D.1. Examples of the acquisition process for AFA task from GSMRL.

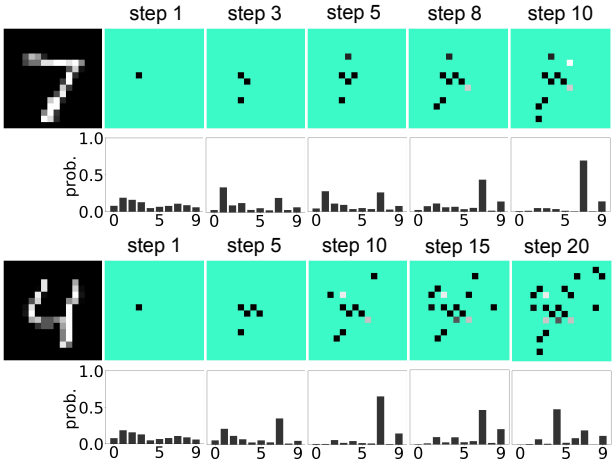


Figure D.2. Examples of the acquisition process for AFA task from GSM+Greedy.

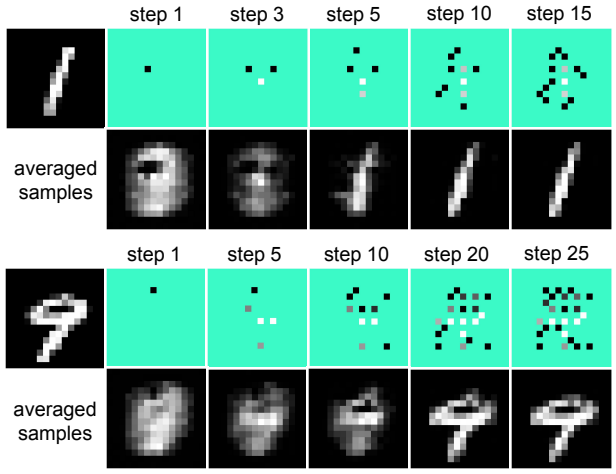


Figure D.3. Examples of the acquisition process for AIR task from GSMRL.

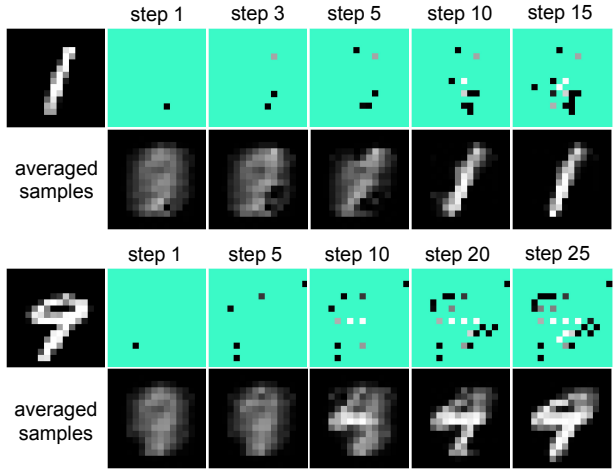


Figure D.4. Examples of the acquisition process for AIR task form GSM+Greedy.

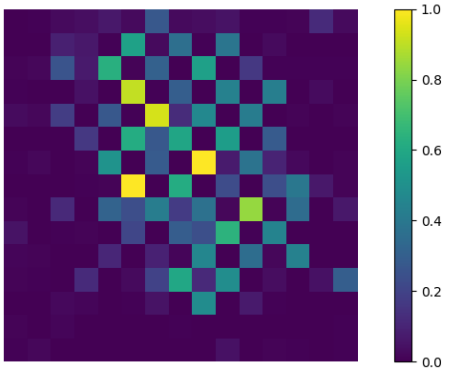


Figure D.5. Acquisition frequency for AFA.

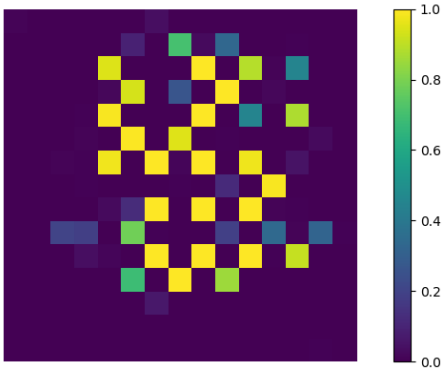
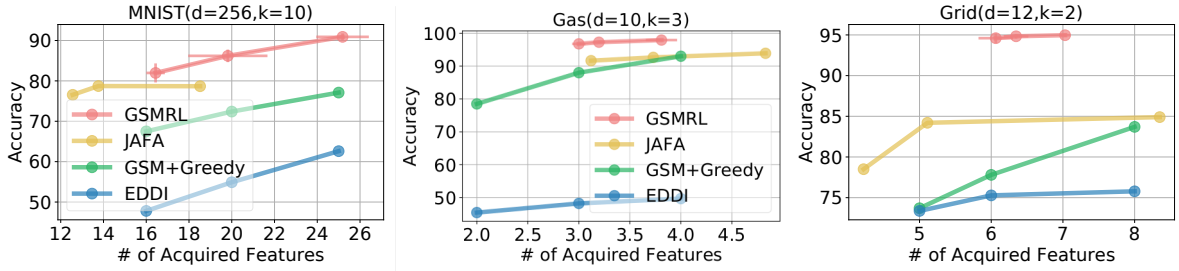
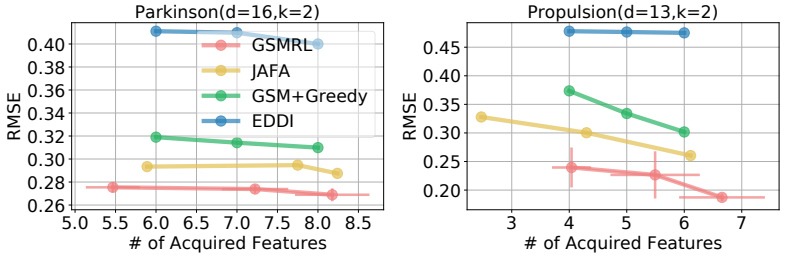


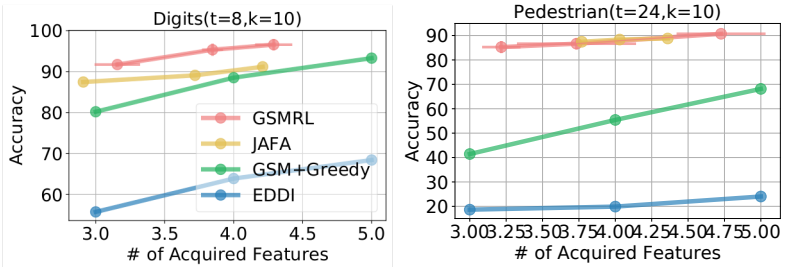
Figure D.6. Acquisition frequency for AIR.



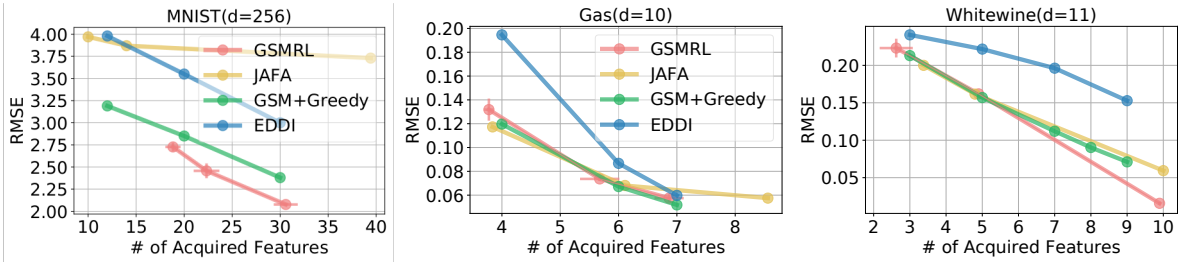
(a) Classification



(b) Regression



(c) Time Series



(d) Unsupervised

Figure D.7. Sensitivity analysis by running multiple times independently. Mean and standard deviation are reported for both the number of acquisitions and task performance.

the number of acquisitions. Here, we utilize another evaluation metric that directly compares the returned reward. We use a normalized reward for evaluation where for a d-dimensional instance, each feature costs $\frac{1}{d}$ and the final classification is rewarded 1 if the prediction is correct, otherwise the reward is zero. The normalized reward is within the rage of $[-1, 1]$ where a correct classification with no feature acquired obtains the highest reward 1, a wrong classification with all feature acquired obtains the lowest reward -1, and a correct classification with all features acquired obtains the reward 0. We report the normalized reward for MNIST classification in Table D.2.

Table D.2. Normalized rewards for MNIST AFA experiments.

Algorithm	Reward
GSMRL	0.7998
JAFA	0.7335
GSM+Greedy	0.7038
EDDI	0.6116