



# A Deeper Understanding of Active Feature Acquisition

Master's thesis in Data Science and AI

Reza Rezvan
Han Wu

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden, 2026

Master's Thesis 2026

# A Deeper Understanding of Active Feature Acquisition

Reza Rezvan
Han Wu

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden, 2026

A Deeper Understanding of Active Feature Acquisition

Reza Rezvan, Han Wu



**Supervisor:**
Linus Aronsson and Valter Schütz, Department of Computer Science and Engineering

**Examiner:**
Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31-772 10 00

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

Typeset in Typst
Gothenburg, Sweden, 2026

A Deeper Understanding of Active Feature Acquisition

Reza Rezvan
Han Wu
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

# Abstract

Data and decision making are increasingly becoming more prevalent. Reinforcement learning (RL) has in recent years become an important field for tackling sequential decision making problems. One of the strengths of RL is its general framework of decision making which has been successfully applied to a wide range of problems. However, the common assumption of fully observable and available data in RL can be a limitation in problems where data is costly to acquire or missing entirely. Active Feature Acquisition (AFA) is a subfield of machine learning concerned with the former problem. Namely, to sequentially acquire features to maximize predictive performance under acquisition costs. However, despite rapid methodological progress, evaluation comparison of AFA methods remains difficult because methods are often evaluated under incompatible protocols. Further, most existing work assumes access to fully available data even though real-world datasets often contain missing values. Our contributions in this thesis are twofold. First, we present `AFABench`, the first benchmarking framework for AFA methods in the classification setting. Along with it, we present the synthetic dataset `CUBE-NM`, which is designed to evaluate potential trade-offs between myopic and non-myopic methods. Second, we extend the existing theoretical framework of the Partially Observable Markov Decision Process (POMDP) of the AFA problem to the missing data setting. We present novel theoretical insights that can be used to understand AFA under missing data. These incremental steps are important for deeper evaluation and understanding of AFA methods. We hope that our `AFABench` framework gets adopted by the community and that our theoretical insights can drive future research for foundational understanding of AFA under missing data.

**Keywords**: Sequential Decision Making, Machine Learning, Reinforcement Learning, Active Feature Acquisition, Benchmarking, Missing Data

## Acknowledgements

First and foremost, we would like to express our deepest gratitude to both of our supervisors, Linus Aronsson and Valter Schütz. Your support, guidance, and encouragement have been instrumental in our journey. Your foundational work on `AFABench` and theoretical discussions about AFA have been invaluable to our research and have significantly shaped the direction of our thesis. We are truly grateful for the opportunity to work with you and for the invaluable knowledge and experience you have shared with us throughout this process. We would also like to thank our examiner, Professor Morteza Haghir Chehreghani, for his insightful feedback, guidance, support, and continued involvement in our work.

Reza Rezvan & Han Wu, Gothenburg, Sweden, 2026-05-26

# Contents

# List of Figures

# List of Tables

# Notation

| Symbol | Meaning |
|---|---|
| **General Conventions** | |
| $\mathsf{x}, \mathsf{y}, \mathsf{m}$ | Sans-serif symbols denote random variables. |
| $x, y, m$ | Ordinary symbols denote realizations of random variables. |
| $\mathcal{X}, \mathcal{Y}, \mathcal{S}$ | Calligraphic symbols denote sets, spaces, or index sets. |
| $x_{\mathcal{S}}, x_j$ | Subscripts denote selected subsets or feature-level components. |
| $x^{(i)}, y^{(i)}$ | Parenthesized superscripts denote dataset instances. |
| $a^l, \pi^\star$ | Other superscripts denote contextual annotations. |
| **Problem Setting** | |
| $\mathcal{X}$ | Set of possible feature vectors / inputs. |
| $\mathcal{X}_i$ | Set of possible values of feature $i$. |
| $\mathcal{S}$ | Indices of features that have been acquired. |
| $\mathcal{U}$ | Indices of features that have not yet been acquired. |
| $\mathcal{Y}$ | Set of possible labels / targets. |
| $[n]$ | Index set $\{1, ..., n\}$. |
| **Data and Features** | |
| $x$ | Realized feature vector / data instance. |
| $x_i$ | Value of feature $i$. |
| $x_{\mathcal{S}}$ | Values of the already acquired features. |
| $x_{\mathcal{U}}$ | Values of the not-yet-acquired features. |
| $y$ | Realized label / target value. |
| $\mathcal{D}$ | Dataset of instances and labels. |
| **Policies and Actions** | |
| $\pi$ | Acquisition policy. |
| $a_t$ | Action chosen at step $t$. |
| `STOP` | Stopping action. |
| $c_i$ | Cost of acquiring feature $i$. |
| $c(\pi[x])$ | Total acquisition cost incurred by policy $\pi$ on instance $x$. |
| $b$ | Hard acquisition budget. |
| $\alpha$ | Trade-off parameter between acquisition cost and predictive performance. |
| **Probabilities and Information** | |
| $\mathbf{1}\{\cdot\}$ | Indicator function, equal to 1 when the condition holds and 0 otherwise. |
| $H(\cdot)$ | Entropy or conditional entropy. |

# 1

# Introduction

181 zettabytes is roughly the equivalent of assigning every grain of sand on every beach on Earth with 24,000 bytes of data [1], [2]. Humanity generates around 181 zettabytes of data annually and this figure is expected to grow to 394 zettabytes by 2028 [3]. From defeating world champions in Go to outperforming human pilots in autonomous drone racing, sequential decision making through reinforcement learning (RL) has achieved remarkable real-world success in recent years [4], [5]. However, in many real-world scenarios, data collection and usage can be costly, time-consuming, and even risky [6]. Furthermore, it might not be necessary to use all features to make sufficient decisions [7], [8].

Consider healthcare as an example: the best treatment plan may be decided based on a small subset of test results. Performing every possible test on all patients would be infeasible in terms of cost and time, and might even pose risks to patients. Alternatively, one can sequentially decide which tests to perform based on available information and only perform the tests needed to create a treatment plan. Active Feature Acquisition (AFA) is a machine learning framework and problem setting concerning sequential decision-making and the trade-off between predictive performance and acquisition cost. Figure 1.1 illustrates the high-level idea behind the AFA decision-making process.

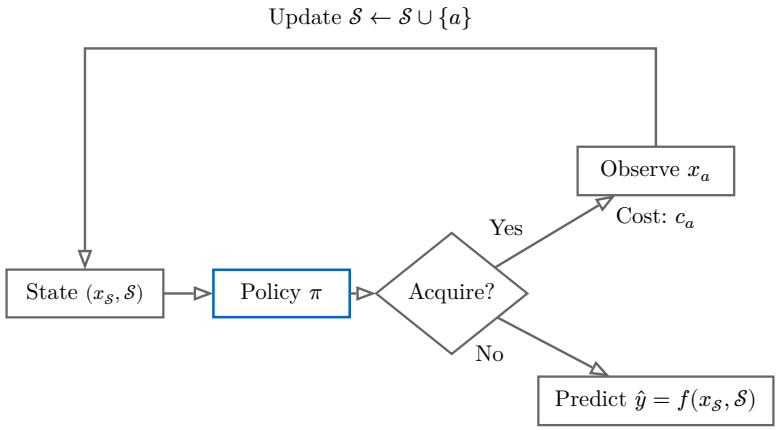



**Figure 1.1:** At each step the policy $\pi$ decides whether to acquire feature $x_a$ incurring cost $c_a$ or to `STOP` and make a prediction based on the acquired features.

Evaluation in machine learning is non-trivial, and the AFA setting introduces additional challenges. In traditional machine learning, evaluation is typically concerned with the predictive performance of a model, often measured by metrics such as accuracy, precision, recall, or F1 score. In contrast, the AFA setting requires a more comprehensive evaluation that considers not only predictive performance but also the acquisition cost trade-off. This necessitates evaluation protocols that capture these trade-offs and provide insight into the decision-making process.

Yet, a common assumption in machine learning, reinforcement learning, and AFA is that data is fully available. However, missing data is a common issue in many real-world datasets, and it can significantly affect performance [9]. A patient's medical record rarely contains all possible conducted tests and measurements. But, when a new (or the same) patient arrives, they can be asked to perform (new) tests based on the available information. This problem setting has previously been overlooked in the AFA literature.

## 1.1 Scope and Research Questions

The primary scope and investigation of this thesis is twofold.

1. First, we develop the `AFABench` evaluation framework, a modular benchmarking framework designed to expand the frontiers of AFA evaluation.
2. Second, we explore the current frontiers of AFA in the missing data setting by understanding previous work and developing new insights into AFA under missing data.

Thus, this thesis addresses the following research questions.

1. What key assumptions and components are necessary for the `AFABench` framework? How can it be designed to effectively and fairly evaluate AFA methods?
2. What insights can be uncovered, both theoretical and empirical, from AFA under missing data? How can studying their performance, optimality, and understanding the resulting trade-offs inform future research directions in AFA under missing data?

## 1.2 Limitations

While this thesis aims to answer the above research questions, there are certain limitations to acknowledge.

1. The evaluation framework developed in this thesis may not cover all possible scenarios of AFA. `AFABench` will focus on the predictive performance and cost efficiency in the classification setting, while other important factors such as interpretability and fairness will not be addressed.
2. We primarily focus on the setting where AFA methods are trained under missing data and evaluated with all features available. This may not cover all

possible scenarios of missing data in AFA, such as training and evaluating under missing data or only evaluating with missing data.

## 1.3 Thesis Outline

The structure of this thesis is organized as follows:

- Chapter 2 surveys related work on missing data in machine learning, the taxonomy of AFA methods, and previous and adjacent work on AFA under missing data.
- Chapter 3 provides the theoretical background on machine learning, neural networks, reinforcement learning, and the formal AFA problem setting.
- Chapter 4 presents the novel `AFABench` framework, describing its design principles and implementation details, along with a new synthetic AFA dataset `CUBE-NM`. The chapter also defines the formulation of AFA under missing data used in the thesis, and describes how `AFABench` implements this setting.
- Chapter 5 illustrates a standard experimental setup for evaluating AFA methods using `AFABench`, discussing previous shortcomings in AFA evaluation and how `AFABench` addresses them. It then presents the results of AFA methods under missing data and discusses the insights uncovered from these results.
- Chapter 6 summarizes the main contributions of the thesis, discusses their implications, and outlines directions for future research.

# 2

# Related Work

This chapter surveys related work on missing data in machine learning, the taxonomy of AFA methods, and previous and adjacent work on AFA under missing data. We first review the foundational taxonomy of missing data mechanisms along with classical and modern approaches to handling missing data in machine learning. We then provide an overview of the taxonomy of AFA methods and their key characteristics. Finally, we discuss previous and adjacent work on AFA under missing data.

## 2.1 Missing Data in Machine Learning

Missing data is a common challenge in real-world machine learning applications, where datasets contain missing or incomplete values for reasons such as data collection issues, privacy concerns, or cost constraints. Traditionally, missing data is often described in terms of missing and observed features, where missing features are those that are not available for a given data instance, and observed features are those that are available. The foundational taxonomy of D. B. Rubin categorizes missingness mechanisms into three types based on the relationship between the missingness pattern and the data.

- Missing Completely at Random (MCAR) means missingness is independent of both the observed and missing values.
- Missing at Random (MAR) means missingness may depend on observed values but not on missing values themselves.
- Missing Not at Random (MNAR) means missingness may depend on the missing values themselves.

These distinctions are critical because they determine which methods can provide valid inference [9]. Classical approaches to handling missing data include listwise deletion, mean or median imputation, and model-based imputation such as multiple imputation or expectation maximization. More recent approaches try to learn directly from the incomplete data. M. Le Morvan, J. Josse, T. Moreau, E. Scornet, and G. Varoquaux proposed NeuMiss networks, which are neural networks specifically designed for missing values. Rather than imputing missing features as a

preprocessing step, NeuMiss networks integrate the missingness pattern directly into the network architecture, enabling end-to-end training that accounts for the information contained in the missingness pattern itself.

## 2.2 Active Feature Acquisition Methods

AFA methods can be broadly categorized along two axes, whether they are myopic (one-step lookahead) or non-myopic (multi-step planning), and whether they use generative or discriminative models for estimating feature informativeness [12].

Myopic methods select actions using one-step estimates of immediate utility. Representative examples include `EDDI`, `GDFS`, and `DIME`. They are typically efficient and stable but do not explicitly optimize long-term effects [13], [14], [15].

Non-myopic methods optimize long-term trade-offs through sequential decision-making, often using reinforcement learning (RL). Representative RL-based methods include `JAFA`, `OL`, and `ODIN`, which can exploit multi-step structure but often at increased computational complexity [16], [17], [18]. There are also non-myopic methods that do not explicitly use RL, such as `AACO`, which uses an acquisition conditioned oracle to approximate non-greedy behavior [19].

In Section 4.1, we provide a more detailed overview of the aforementioned methods. We invite the reader to refer to the original works for an even more comprehensive understanding of these methods and their key characteristics.

## 2.3 Active Feature Acquisition Under Missing Data

Compared with traditional missing data learning, explicit treatment of missing data in AFA remains limited. J. Janisch, T. Pevný, and V. Lisý formulate "costly feature classification" as sequential decision making. "Costly feature classification" as sequential decision-making is precisely the AFA problem setting. In their work, they introduce a blocking approach to mitigate missing data. Actions that would acquire the missing feature(s) instead become blocked and cannot be performed. Their key result is that this simple yet effective blocking approach works in practice for real-world scenarios.

One of the first works to explicitly study the evaluation of AFA under missing data is H. von Kleist, A. Zamanian, I. Shpitser, and C. Ghanem, which introduces the Active Feature Acquisition Performance Evaluation (AFAPE). Their semi-offline framework handles blocked acquisitions similarly to J. Janisch, T. Pevný, and V. Lisý and introduces estimators based on inverse probability weighting (IPW), direct methods (DM), and double reinforcement learning (DRL). A key result from AFAPE is that availability assumptions and selection bias in data can strongly affect predictive performance. Although AFAPE focuses on evaluation under missing data, its debiasing perspective is important context when discussing biased utility estimates under missing data.

A very recent line of work bridges missing data with sequential decision-making. J. Wendland *et al.* introduce miss-MDPs. Their framework treats missing state features as the output of an explicit missingness function and studies how to estimate that function from histories in order to compute near-optimal policies. We describe the formal setup in Section 3.3.6. The key connection to this thesis is their distinction between missingness being ignorable for belief updates under MAR-type assumptions and still being relevant for planning and optimality.

Finally, we emphasize the difference in terminology between the missing data learning literature and the AFA literature. Missing data learning is a static setting where models are trained and used on full data instances that have missing (unavailable) features and observed (available) features. AFA, on the other hand, is a sequential decision-making setting where the goal is to acquire only a subset of features over time. In AFA, observed features refer to features that have been acquired throughout the decision-making process, while missing features refer to features that cannot be acquired during the decision-making process for a given instance.

# 3

# Background

This chapter provides the theoretical background necessary to understand the concepts and methods used in this thesis. We begin with an overview of different machine learning (ML) paradigms in Section 3.1. We then introduce the foundations of neural networks in Section 3.2, followed by reinforcement learning (RL) foundations and miss-MDPs in Section 3.3. Afterwards, we formalize the AFA problem setting in Section 3.4. Moreover, Section 3.1 and Section 3.2 are based mainly on [22], [23], [24], [25], [26], and [13]. Section 3.3 is based on [27], [28], and [21], and Section 3.4 is based on [12].

## 3.1 Machine Learning

Machine Learning (ML) is the field of study within artificial intelligence concerning the development and analysis of algorithms that can learn from data and make predictions or decisions without being explicitly programmed for specific tasks. Traditionally, ML approaches can be broadly categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. We introduce the first two paradigms here, with a focus on supervised learning, and then dedicate a more detailed section to reinforcement learning on its own in Section 3.3.

### 3.1.1 Supervised Learning

In a supervised learning setting, a dataset $\mathcal{D} \coloneqq \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ is provided, where $x^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ represents the input features and $y^{(i)} \in \mathcal{Y}$ represents the corresponding target output for the $i$-th example. The goal is to learn a mapping function $f : \mathcal{X} \to \mathcal{Y}$ that predicts the target output for new, unseen inputs. Commonly, supervised learning tasks are categorized into classification and regression. The classification task is usually formalized as follows.

**Definition 3.1** (The Classification Task).

Given a feature vector $x \in \mathcal{X} \subseteq \mathbb{R}^d$ that describes an object belonging to one of $C$ classes from the set $\mathcal{Y} \coloneqq \{1, 2, ..., C\}$, predict the class label $y \in \mathcal{Y}$.

From this, the classification learning problem can be formalized as follows.

**Definition 3.2** (The Classification Learning Problem).

Given a dataset $\mathcal{D} \coloneqq \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ where $x^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ is a feature vector and $y^{(i)} \in \mathcal{Y}$ is the corresponding class label, learn a function $f : \mathcal{X} \to \mathcal{Y}$ that accurately predicts the class label $y$ for any feature vector $x \in \mathcal{X}$.

Usually, the classification error is used to assess how poorly a function $f$ performs on the classification task.

**Definition 3.3** (The Classification Error).

Given a dataset $\mathcal{D} \coloneqq \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ and the function $f : \mathcal{X} \to \mathcal{Y}$, the classification error of $f$ on $\mathcal{D}$ is defined as,

$$\text{Error}(f, \mathcal{D}) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{f\left(x^{(i)}\right) \neq y^{(i)}\right\},$$

where $\mathbf{1}\{\cdot\}$ is the indicator function that equals 1 if the condition inside is true and 0 otherwise.

Conversely, the classification accuracy is commonly used to evaluate and optimize the performance of a classifier.

**Definition 3.4** (The Classification Accuracy).

Given a dataset $\mathcal{D} \coloneqq \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ and the function $f : \mathcal{X} \to \mathcal{Y}$, the classification accuracy of $f$ on $\mathcal{D}$ is defined as,

$$\text{Accuracy}(f, \mathcal{D}) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{f\left(x^{(i)}\right) = y^{(i)}\right\} = 1 - \text{Error}(f, \mathcal{D}).$$

Similarly, for regression tasks the target variable $y$ is instead continuous and performance is evaluated using regression metrics, such as Mean Squared Error (MSE) for error measurement and the $R^2$ score for goodness of fit.

### 3.1.2 Unsupervised Learning

Unsupervised learning involves training models on the dataset $\mathcal{D} \coloneqq \left\{x^{(i)}\right\}_{i=1}^{N}$, which consists solely of input features without corresponding target outputs. Inference in this case is not the task of predicting a target variable, but rather uncovering insights about the data distribution or structure. Common tasks include clustering, dimensionality reduction, or density estimation on the dataset

itself or on new (possibly generated) samples. Since the main focus of this thesis is set in a supervised learning or reinforcement learning context, we will not formally define common unsupervised learning tasks and their corresponding learning problems here.

## 3.2 Neural Networks

A feed-forward neural network is a composition of affine transformations and nonlinear activation functions. At the level of a single neuron, inputs are combined linearly and then transformed by an activation function, as illustrated in Figure 3.1.

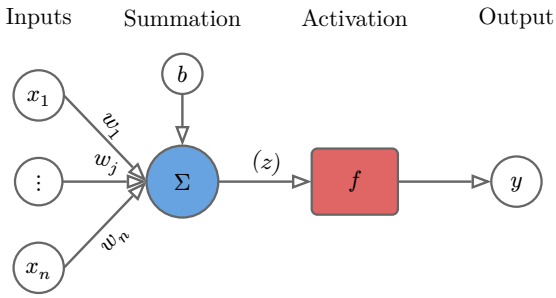



**Figure 3.1:** A single neuron consisting of $n$ inputs $x_i$ and their corresponding weights $w_i$, the bias $b$, the summation stage, and an activation function $f$ and outputting $y$.

**Definition 3.5** (Single Neuron Mapping).

Let $x \in \mathbb{R}^n$ be the input vector, $w \in \mathbb{R}^n$ be the weight vector, and $b \in \mathbb{R}$ be the bias term. The output of a single neuron is given by,

$$y := f(w^T x + b) = f\left(\sum_{i=1}^{n} w_i x_i + b\right),$$

where $f : \mathbb{R} \to \mathbb{R}$ is the activation function applied elementwise.

By stacking multiple layers of many neurons, we can build a feed-forward neural network, or as referred to in the literature, a multilayer perceptron (MLP). The layers are typically organized into an input layer, one or more hidden layers, and a final output layer. An example of a fully connected architecture with one hidden layer is shown in Figure 3.2.

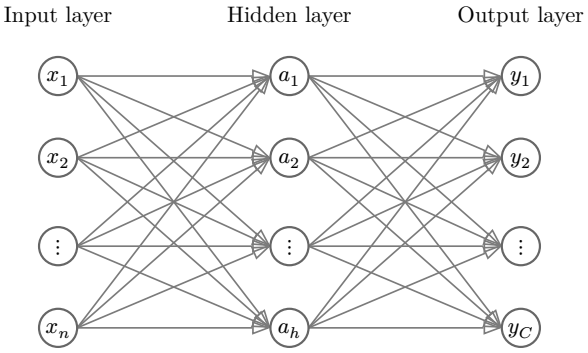



**Figure 3.2:** A feed-forward network with $n$ inputs, $h$ hidden neurons, and $C$ outputs.

**Definition 3.6** (Feed-Forward Neural Network Layer Mapping).

Let $l \in \{1, ..., L-1\}$ be the index of the hidden layers, let $a^0 := x$ be the input at the input layer, and let $a^l$ be the post-activation output at hidden layer $l$. The layer mapping is defined as,

$$z^l := W^l a^{l-1} + b^l, \quad a^l := f^l(z^l),$$

where $W^l \in \mathbb{R}^{h_l \times h_{l-1}}$ and $b^l \in \mathbb{R}^{h_l}$ are the learnable parameters for each connected layer and biases, respectively, and the activation function $f^l$ is applied elementwise.

Activation functions ought to be nonlinear; without nonlinearity, stacked affine layers collapse to a single affine map. Such a map can only represent linear decision boundaries and is insufficient for complex tasks (or even simple problems, e.g., the XOR problem) [23]. Common choices include sigmoid, ReLU, and GELU, shown in Figure 3.3.

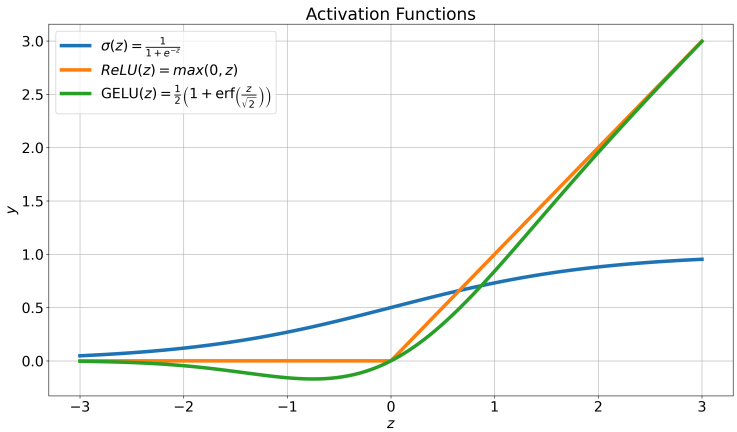



**Figure 3.3:** Three examples of activation functions used in machine learning, sigmoid, ReLU, and GELU. The sigmoid function maps inputs to the range (0, 1), ReLU outputs the input if it's positive and zero otherwise, and GELU is a smooth approximation of ReLU that allows for small negative outputs.

In the classification setting, the outputs are usually interpreted as class probabilities, this requires the final output layer to map the last pre-activation $z^L$ to a probability distribution over the classes. Usually, the softmax function is used for this purpose.

**Definition 3.7** (Softmax Output Layer).

For logits $z \in \mathbb{R}^C$, softmax is defined componentwise by,

$$\text{softmax}(z_c) = \frac{\exp(z_c)}{\sum_{k=1}^{C}} \exp(z_k), \quad c \in \{1, ..., C\}.$$

Using this probabilistic output map, we can define a classifier as a parameterized function.

**Definition 3.8** (Feed-Forward Neural Network Classifier).

Let $\theta := \{W^l, b^l\}_{l=1}^{L-1}$ be the collection of all trainable parameters in the network. A feed-forward neural network classifier is a function,

$$f_\theta : \mathcal{X} \to \Delta(\mathcal{Y}), \quad \Delta(\mathcal{Y}) := \left\{ p \in \mathbb{R}^C : \sum_{c=1}^{C} p_c = 1, p_c \geq 0 \forall c \right\},$$

defined by the layer recursion above, where $\Delta(\mathcal{Y})$ is the probability simplex over class labels.

Hence, for a $C$-class network, $p_\theta(x) := \text{softmax}(z^L)$ are the predicted class probabilities. Given the labeled dataset $\mathcal{D} = \left\{ \left( x^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$, neural networks are trained by empirical risk minimization of a loss function that quantifies the discrepancy between predicted probabilities and true labels.

**Definition 3.9** (Empirical Risk Minimization).

Let $\mathcal{L}(\theta)$ be an arbitrary loss function that measures the discrepancy between the predicted probabilities $f_\theta\left(x^{(i)}\right)$ and the true labels $y^{(i)}$ over the dataset $\mathcal{D} := \left\{ \left( x^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$, in a meaningful way. The empirical risk minimization objective is to find parameters that minimize,

$$\theta^\star := \arg\min_\theta \mathcal{L}(\theta).$$

For multiclass classification, the cross-entropy loss function is a common choice, which corresponds to maximizing the log-likelihood of the data under the model.

**Definition 3.10** (Cross-Entropy Objective for Multiclass Classification).

Let $\hat{p}^{(i)} := f_\theta\big(x^{(i)}\big) \in \Delta(\mathcal{Y})$ denote predicted class probabilities for sample $i$, with $\hat{p}_c^{(i)}$ being the predicted probability of class $c$ for sample $i$. The empirical cross-entropy objective is,

$$\theta^\star := \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbf{1}\big\{y^{(i)} = c\big\} \log \hat{p}_c^{(i)}.$$

To optimize this objective, we typically use gradient-based optimization methods such as stochastic gradient descent (SGD) or its variants. Backpropagation computes the gradient efficiently by applying the chain rule from output to input layers [29]. In this thesis, classifiers are neural networks, namely MLPs trained with the cross-entropy objective.

### 3.2.1 Generative vs. Discriminative Modeling

In the classification setting, two standard modeling viewpoints are discriminative and generative modeling [23]. Discriminative modeling focuses on the conditional distribution of the target given the features, while generative modeling models the joint distribution of features and targets. The classification objectives can be defined as follows,

**Definition 3.11** (Discriminative Classification Objective).

A discriminative model specifies a conditional distribution $p_\theta(y \mid x)$ and learns parameters by maximizing the conditional log-likelihood,

$$\theta^\star := \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} \log p_\theta\big(y^{(i)} \mid x^{(i)}\big).$$

The induced classifier predicts the most likely class label for a new input $x$ by,

$$\hat{y} := \arg\max_{y \in \mathcal{Y}} p_{\theta^\star}(y \mid x).$$

The complementary perspective is to model the joint data-generating process directly,

**Definition 3.12** (Generative Classification Objective).

A generative model specifies the joint distribution $p_\theta(x, y) := p_\theta(x \mid y)p_\theta(y)$ and learns parameters by maximizing the joint log-likelihood,

$$\theta^\star := \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} \log p_\theta\big(x^{(i)}, y^{(i)}\big).$$

Classification is then obtained by Bayes' decision rule,

$$\hat{y} := \arg\max_{y\in\mathcal{Y}} p_{\theta^\star}(x \mid y) p_{\theta^\star}(y).$$

### 3.2.2 Variational Autoencoders (VAEs)

Variational autoencoders (VAEs) are generative models that learn a low-dimensional stochastic representation of input data, realized in the form of neural networks. The generative model introduces a latent random variable $z$ with prior $p(z) = \mathcal{N}(0, I)$ and defines the likelihood of the observed data $p_\theta(x, z) := p(z)p_\theta(x \mid z)$. Exact inference of $p_\theta(z \mid x)$ is generally intractable, so VAEs use an amortized variational posterior $q_\varphi(z \mid x)$. The standard decomposition of the log-likelihood is then,

$$\log p_\theta(x) = \mathcal{L}^{\text{VAE}}_{\text{ELBO}}(x; \theta, \varphi) + \text{KL}\Big(q_\varphi(z \mid x) \parallel p_\theta(z \mid x)\Big).$$

where KL is the Kullback-Leibler divergence between the variational posterior and the true posterior and $\mathcal{L}^{\text{VAE}}_{\text{ELBO}}$ is the evidence lower bound (ELBO) on the log-likelihood.

**Definition 3.13** (Kullback-Leibler Divergence).

The Kullback-Leibler (KL) divergence between two probability distributions $P$ and $Q$ is defined as,

$$\text{KL}(P \parallel Q) := \mathbb{E}_{x\sim P}\left[\log\left(\frac{P(x)}{Q(x)}\right)\right].$$

The Kullback-Leibler divergence measures how much information is lost when $Q$ is used to approximate $P$. Since the KL term is non-negative by definition, maximizing the ELBO gives a tractable surrogate objective for likelihood maximization.

**Definition 3.14** (Variational Autoencoder Evidence Lower Bound (ELBO)).

For one input $x$, the ELBO can be written as,

$$\mathcal{L}^{\text{VAE}}_{\text{ELBO}}(x; \theta, \varphi) := \mathbb{E}_{z\sim q_\varphi(\cdot|x)}[\log p_\theta(x \mid z)] - \text{KL}\Big(q_\varphi(z \mid x) \parallel p(z)\Big).$$

Maximizing this objective jointly improves reconstruction quality and regularizes the latent posterior toward the prior. To backpropagate through stochastic latent samples, VAEs use the reparameterization trick,

$$z = \mu_\varphi(x) + \sigma_\varphi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I).$$

where $\odot$ denotes elementwise multiplication, and $\mu_\varphi(x)$ and $\sigma_\varphi(x)$ are the mean and standard deviation predicted by the encoder network.

VAEs assume a fully observed input vector; when the input can only be partially observed, however, the encoder $q_\varphi(z \mid x)$ is no longer directly applicable without ad hoc preprocessing.

### 3.2.3 Partial Variational Autoencoders (PVAEs)

Partial Variational Autoencoders (PVAEs) are an extension of VAEs that can handle partially observed inputs by conditioning the inference model on observed values and a binary missingness mask. Throughout this thesis, $m_i = 1$ means that feature $i$ is missing or blocked, while $m_i = 0$ means that it is observed or acquirable. The decoder keeps the same generative model $p_\theta(x \mid z)$, while the inference model conditions on observed values and a mask $q_\varphi\big(z \mid x^{\text{obs}}, m\big)$. The corresponding training objective only reconstructs observed coordinates.

**Definition 3.15** (Partial Variational Autoencoder ELBO).

For one input $x$ with binary mask $m$, the PVAE ELBO is,

$$\mathcal{L}^{\text{PVAE}}_{\text{ELBO}}(x, m; \theta, \varphi) := \mathbb{E}_{z \sim q_\varphi(\cdot \mid x^{\text{obs}}, m)} \left[ \sum_{i=1}^{n} (1 - m_i) \log p_\theta(x_i \mid z) \right] - \beta \, \mathrm{KL}\Big(q_\varphi\big(z \mid x^{\text{obs}}, m\big) \,\Big\|\, p(z)\Big),$$

where the factor $(1 - m_i)$ ensures that only observed features contribute to the reconstruction term and $\beta$ controls KL regularization.

When the latent variable is also used for prediction, a classifier head $p_\psi(y \mid z)$ can be trained jointly with a supervised loss term. To parameterize $q_\varphi\big(z \mid x^{\text{obs}}, m\big)$ under arbitrary masks, a common choice is a permutation-invariant set encoder based on PointNet-style aggregation [30], [31]. This gives an inference model that can consume arbitrary observed feature subsets without imposing an order on the acquired features.

## 3.3 Reinforcement Learning (RL)

Reinforcement Learning (RL) is the discipline of machine learning concerned with how an intelligent agent ought to take actions in a dynamic environment to

maximize a reward signal. An overview of the agent-environment interaction in RL is shown in Figure 3.4.

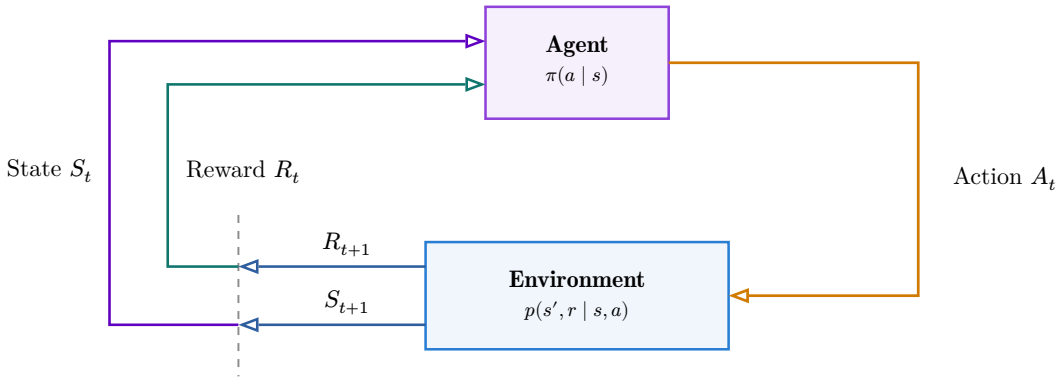



**Figure 3.4:** Illustration of the agent-environment interaction in RL. At time step $t$, the agent selects action $A_t$ and the environment returns reward $R_{t+1}$ and next state $S_{t+1}$.

Formally, the agent and environment interact at each discrete time step in the sequence $t = 0, 1, 2, 3, \ldots$. At time step $t$, the agent receives some representation of the environment's state $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, and based on that representation selects an action $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of actions available in state $S_t$. One time step later, at $t+1$, as a consequence of its action, the agent receives a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and finds itself in a new state $S_{t+1}$. At each time step, the agent uses a mapping from states to probabilities of selecting each possible action. This is formally defined as the policy,

**Definition 3.16** (Policy).

A policy $\pi$ is a mapping from states to probabilities of selecting each possible action. Thus, $\pi(a \mid s)$ is the probability that action $a$ is taken in state $s$,

$$\pi(a \mid s) := P(A_t = a \mid S_t = s).$$

### 3.3.1 Markov Decision Processes (MDPs)

The RL problem setting is commonly formalized using the framework of Markov Decision Processes (MDPs) [27]. As the name suggests, MDPs are based on the Markov property.

**Definition 3.17** (The Markov Property).

The Markov property states that the future state of a process depends only on the current state and action, and not on the sequence of events that preceded it. Formally, for any time step $t$, the probability of transitioning to the next state

$S_{t+1}$ given the current state $S_t$ and action $A_t$ is independent of all previous states and actions,

$$P(S_{t+1} \mid S_t, A_t) = P(S_{t+1} \mid S_0, A_0, ..., S_t, A_t).$$

Formally, an MDP is defined as follows.

**Definition 3.18** (Markov Decision Process (MDP)).

A Markov Decision Process (MDP) is fully characterized by the tuple $(\mathcal{S}, \mathcal{A}, R, p)$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of all possible actions, $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the (immediate) reward function under action $a$ in state $s$, and

$$p(s', r \mid s, a) : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$$

is the joint probability of next state $s'$ and reward $r$ given current state-action pair $(s, a)$.

### 3.3.2 Task Formulations and Return

The reward hypothesis in reinforcement learning states.

**Definition 3.19** (The Reward Hypothesis (Informal)).

Everything we could possibly want an agent to do can be formulated as maximizing expected cumulative reward.

Furthermore, reinforcement learning tasks can be categorized into episodic and continuing tasks.

**Definition 3.20** (Episodic and Continuing Tasks).

An episodic task has a well-defined terminal state or time step $T$ after which the episode ends, while a continuing task does not have a terminal state and continues indefinitely.

From these definitions, we can formalize the control problem by defining the return, which is the cumulative reward that the agent aims to maximize.

**Definition 3.21** (Return).

The return $G_t$ at time step $t$ is defined as the cumulative reward from time step $t$ onward.

$$G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k,$$

where $\gamma \in [0, 1]$ is the discount factor that determines the present value of future rewards, and $T$ is the terminal time step for episodic tasks or infinity for continuing tasks.

Based on these return definitions, the control problem is to optimize policy performance from an initial state distribution.

**Definition 3.22** (RL Control Objective).

Let $\mu_0$ be an initial state distribution. A policy $\pi$ is optimized by maximizing expected return,

$$\pi^\star \in \arg\max_{\pi} \mathbb{E}_{\pi, S_0 \sim \mu_0}[G_0].$$

### 3.3.3 (Action-)Value Functions and Bellman Equations

The so-called (action-)value functions are defined as the expected return under a policy, starting from a given state or state-action pair.

**Definition 3.23** (Value and Action-Value Functions).

For policy $\pi$, we define the value function $V_\pi$ and action-value function $Q_\pi$, respectively, as

$$V_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s],$$
$$Q_\pi(s, a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a].$$

The Bellman equations express these consistency relations, showing how the value of a state or state-action pair can be decomposed into immediate reward and the discounted value of subsequent states.

**Definition 3.24** (Bellman Equations).

In an MDP, the (action-)value functions satisfy

$$V_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma V_\pi(s')],$$

$$Q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a) \left[ r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a' \mid s') Q_\pi(s', a') \right].$$

Replacing policy averaging by maximization yields the optimal-control counterpart.

**Definition 3.25** (Optimal Value Functions and Bellman Optimality).

We define the optimal value functions as the maximum expected return achievable from each state or state-action pair under any policy,

$$V_\star(s) := \max_\pi V_\pi(s),$$

$$Q_\star(s,a) := \max_\pi Q_\pi(s,a).$$

It is easy to verify that these optimal value functions satisfy the Bellman optimality equations,

$$V_\star(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r \mid s, a)[r + \gamma V_\star(s')],$$

$$Q_\star(s,a) = \sum_{s',r} p(s', r \mid s, a) \left[ r + \gamma \max_{a' \in \mathcal{A}(s')} Q_\star(s', a') \right].$$

Thus, an optimal policy can be obtained by acting greedily with respect to the optimal action-value function,

**Definition 3.26** (Optimal Policy).

An optimal policy $\pi_\star$ is any policy that achieves the optimal value functions, which can be obtained by acting greedily with respect to $Q_\star$,

$$\pi_\star(s) \in \arg\max_{a \in \mathcal{A}(s)} Q_\star(s,a).$$

The finite-state equations above are written with sums. For continuous state or action spaces, the same definitions hold with expectations/integrals replacing summations. These recursive equations are the mathematical basis for dynamic programming (DP), temporal-difference (TD) learning, and many (deep) RL algorithms, e.g., Q-learning, DQN, and DDPG.

### 3.3.4 Model-Free and Model-Based RL

To further distinguish RL methods, we adopt the standard planning-learning viewpoint in [27] and [32]. The key distinction is whether the method uses an explicit model of environment dynamics during policy/value improvement.

An environment model predicts consequences of actions. In MDP notation, this is namely the transition reward distribution $p(s', r \mid s, a)$. A distribution model returns the full conditional distribution, while a sample model returns sampled outcomes $\left(\tilde{S}', \tilde{R}\right)$ for queried $(s, a)$. Planning is then any computation that uses this model to improve a policy or value function.

In action-value control, model-based planning can be expressed through either expected (distribution-model) backups or sampled (sample-model) backups. For a distribution model, a planning backup has the form,

$$Q(s,a) \leftarrow \sum_{s',r} \hat{p}(s', r \mid s, a)\left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s', a')\right].$$

For a sample model, a corresponding planning backup is,

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R + \gamma \max_{a' \in \mathcal{A}(S')} Q(S', a') - Q(s,a)\right],$$

where $(S', R)$ is sampled from the model given $(s, a)$.

The expected backup marginalizes over all model outcomes, while the sampled backup uses one model sample as a stochastic approximation of the same target. The learning target is produced by model-generated transitions rather than only by real interaction.

**Definition 3.27** (Model-Based RL).

A method is model-based if it maintains a known or learned model and performs policy/value improvements that explicitly depend on model-generated transitions. Equivalently, at least one update uses a model-based target of the form

$$y_{\mathrm{MB}}(s,a) := \mathbb{E}_{(s',r)\sim\hat{p}(\cdot,\cdot\mid s,a)}\left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s', a')\right],$$

or a sampled approximation where $(s', r) \sim \hat{p}(\cdot, \cdot \mid s, a)$.

Model-free methods, in contrast, avoid this model-planning step and update directly from experienced interaction with the environment.

**Definition 3.28** (Model-Free RL).

A method is model-free if policy/value updates are computed directly from experienced transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ and do not require model-generated transitions for planning. The corresponding one-step control target is

$$y_{\mathrm{MF},t} := R_{t+1} + \gamma \max_{a \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a),$$

where $(S_t, A_t, R_{t+1}, S_{t+1})$ comes from real environment interaction.

Conceptually, model-based RL uses a predictive world model to "think ahead" before acting, while model-free RL learns action values or policies directly from trial-and-error interaction. Equivalently, model-based methods optimize against imagined (model-generated) transition targets, whereas model-free methods optimize against experienced transition targets. In practice, model-based methods can be more sample-efficient when the model is accurate, but may suffer from model bias when the model is misspecified. Model-free methods avoid explicit model bias, but often require more environment interaction [27], [32].

### 3.3.5 Partially Observable MDPs (POMDPs)

Many practical RL problems have partially observable environments. The agent cannot observe the state directly and instead receives an observation as a noisy or incomplete signal about the underlying state. Consider the following example from [33],

**Example 3.1** (The Crying Baby Problem).

Our goal is to care for a baby, and we do so by choosing at each time step whether to feed the baby, sing to it, or ignore it.

The baby becomes hungry over time. One does not directly observe whether the baby is hungry, but instead receives a noisy observation in the form of whether the baby is crying. A hungry baby cries 80% of the time, whereas a sated baby cries 10% of the time. Singing to the baby changes these percentages to 90% and 0%, respectively.

Thus, the state, action, and observation spaces are,

$$\mathcal{S} := \{\text{Sated}, \text{Hungry}\},$$
$$\mathcal{A} := \{\text{Feed}, \text{Sing}, \text{Ignore}\},$$
$$\mathcal{O} := \{\text{Crying}, \text{Quiet}\},$$

with the transition dynamics,

$$\begin{aligned}
\mathbb{T}(\text{Sated} \mid \text{Hungry}, \text{Feed}) &= 100\%, \\
\mathbb{T}(\text{Hungry} \mid \text{Hungry}, \text{Sing}) &= 100\%, \\
\mathbb{T}(\text{Hungry} \mid \text{Hungry}, \text{Ignore}) &= 100\%, \\
\mathbb{T}(\text{Sated} \mid \text{Sated}, \text{Feed}) &= 100\%, \\
\mathbb{T}(\text{Hungry} \mid \text{Sated}, \text{Sing}) &= 10\%, \\
\mathbb{T}(\text{Hungry} \mid \text{Sated}, \text{Ignore}) &= 10\%,
\end{aligned}$$

and finally, with the observation dynamics,

$$\begin{aligned}
\mathbb{O}(\text{Crying} \mid \text{Feed}, \text{Hungry}) &= 80\%, \\
\mathbb{O}(\text{Crying} \mid \text{Sing}, \text{Hungry}) &= 90\%, \\
\mathbb{O}(\text{Crying} \mid \text{Ignore}, \text{Hungry}) &= 80\%, \\
\mathbb{O}(\text{Crying} \mid \text{Feed}, \text{Sated}) &= 10\%, \\
\mathbb{O}(\text{Crying} \mid \text{Sing}, \text{Sated}) &= 0\%, \\
\mathbb{O}(\text{Crying} \mid \text{Ignore}, \text{Sated}) &= 10\%.
\end{aligned}$$

This setting is modeled by a Partially Observable Markov Decision Process (POMDP), shown in Figure 3.5.

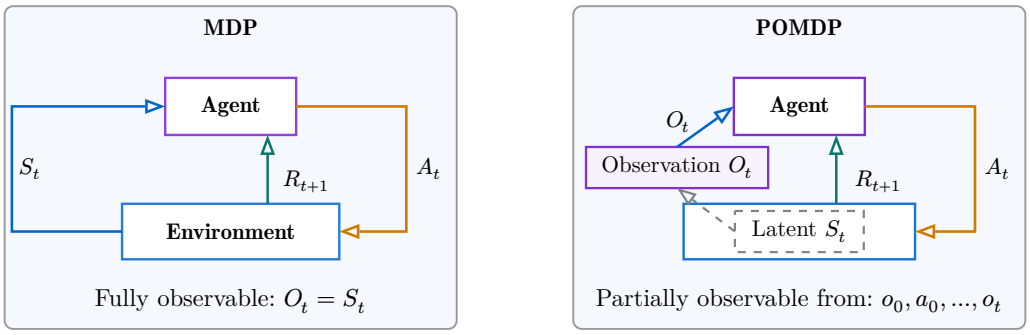



**Figure 3.5:** Comparison of MDP and POMDP interaction models. In an MDP, the agent observes the state directly ($O_t = S_t$). In a POMDP, the state is latent and the agent acts based on observations/history.

**Definition 3.29** (Partially Observable Markov Decision Process (POMDP)).

A Partially Observable Markov Decision Process (POMDP) is fully characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbb{T}, \mathbb{T}_0, \mathbb{O}, R, N)$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of all possible actions, and $\mathcal{O}$ is the set of all possible observations.

$\mathbb{T}(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function and $\mathbb{T}_0(s) : \mathcal{S} \to [0, 1]$ is the initial state distribution.

$\mathbb{O}(o \mid s, a) : \mathcal{O} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the observation probability function and $R : (\mathcal{O} \times \mathcal{A})^N \times \mathcal{O} \to \mathbb{R}$ is the reward function for episodes $\tau \in (\mathcal{O} \times \mathcal{A})^N \times \mathcal{O}$.

Finally, $N \in \mathbb{N} \cup \{+\infty\}$ is the time-horizon of the POMDP.

Initially, at time step $t = 0$, the agent receives an observation $O_0$ drawn from an initial observation distribution, where $s_0$ is drawn from the initial state distribution $\mathbb{T}_0(\cdot)$. At each subsequent time step $t = 1, 2, ...$, the agent is in some unobservable state $s_t \in \mathcal{S}$, chooses an action $a_t \in \mathcal{A}$, and updates the unobservable world state by sampling $s_{t+1}$ from the transition distribution $\mathbb{T}(\cdot \mid s_t, a_t)$. Furthermore, the agent receives an observation $o_{t+1}$ drawn from the distribution $\mathbb{O}(\cdot \mid s_{t+1}, a_t)$.

We observe an episode $\tau = (o_0, a_0, ..., o_{N-1}, a_{N-1}, o_N) \in (\mathcal{O} \times \mathcal{A})^N \times \mathcal{O} =: \mathcal{T}$ with reward $R(\tau) \in \mathbb{R}$. The agent's goal is to choose a policy $\pi = \{\pi_h : \mathcal{T}_h \to \Delta(\mathcal{A})\}$, where $\mathcal{T}_h := (\mathcal{O} \times \mathcal{A})^h \times \mathcal{O}$ is the space of $h$-trajectories (i.e., the history of observations and actions up to time step $h$) and $\Delta(\mathcal{A})$ is the $\mathcal{A}$-simplex, namely the space of probability measures on $\mathcal{A}$, to maximize the expected episodic reward $\mathbb{E}_{\mathcal{T} \sim \mathbb{T}_\pi}[R(\tau)]$, where $\mathbb{T}_\pi$ is the episodic distribution where actions are sampled according to the policy $\pi$.

The major difference between POMDPs and MDPs is that in POMDPs, the agent does not have direct access to the underlying state $S_t$ and must make decisions based on the history of observations and actions, which can be high-dimensional and complex. This makes the control problem more challenging, as the agent must infer the latent state from its observations and learn a policy that maps this inferred state to actions. One common approach to solving POMDPs is to use belief states, which are probability distributions over the latent states given the history of observations and actions.

**Definition 3.30** (Belief-State Formulation of a POMDP).

Let $H_t$ be the history of observations and actions up to time step $t$, then the belief state at time step $t$ is defined as the probability distribution over states given the history,

$$b_t(s) := P(S_t = s \mid H_t).$$

The belief state is a sufficient statistic for control in a POMDP, meaning that the optimal action can be determined solely based on the belief state without needing to consider the entire history. It is commonly updated via Bayes' rule,

$$b_{t+1}(s') = \eta \mathbb{O}(o_{t+1} \mid s', A_t) \sum_{s \in \mathcal{S}} \mathbb{T}(s' \mid s, A_t) b_t(s),$$

where $\eta$ is a normalization constant that ensures $b_{t+1}$ is a valid probability distribution.

### 3.3.6 Missingness-MDPs

Missingness-MDPs, or miss-MDPs, were introduced by J. Wendland *et al.* to make missing data an explicit part of sequential decision making. A POMDP can represent incomplete observations, but it does not say why components are missing, whether the missingness is MCAR, MAR, or MNAR, or whether the missingness mechanism can be learned from data. Miss-MDPs restrict the POMDP observation model so that partial observability is caused by a missingness function over state features.

**Definition 3.31** (Missingness-MDP).

A miss-MDP is a tuple $(\mathcal{S}, \mathcal{A}, T, b_0, R, \mathcal{Z}, M, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, $T$, $b_0$, $R$, and $\gamma$ are the state space, action space, transition function, initial belief, reward function, and discount factor as in a POMDP. The state space is finite and factored as $\mathcal{S} = \bigtimes_{i \in I} \mathcal{S}_i$. The observation space has the same feature indices,

$$\mathcal{Z} = \bigtimes_{i \in I} (\mathcal{S}_i \cup \{\perp\}),$$

where $\perp$ denotes a missing-value symbol. The missingness function $M : \mathcal{S} \to \Delta(\mathcal{Z})$ satisfies that for every state $s$, every possible observation $z$ generated from $M(s)$, and every feature $i \in I$, either $z_i = s_i$ or $z_i = \perp$.

The main problem studied by J. Wendland *et al.* is not merely how to define this model class, but how to compute a good policy when the missingness function is unknown. Given a miss-MDP $P$ with unknown $M$, a dataset of observation and action histories collected under an unknown but fair policy, a precision $\varepsilon$, and a confidence level $\delta$, their goal is to estimate a missingness function $\widehat{M}$ and then solve the induced approximate miss-MDP. The desired guarantee is that the resulting policy is near-optimal in the true miss-MDP with high probability,

$$\Pr\left(\sup_{\pi \in \Pi} V_{P(\pi)} - V_{P(\pi_{\text{alg}})} \leq \varepsilon\right) \geq 1 - \delta.$$

They give PAC-style (Probably Approximately Correct) guarantees for missingness functions that are identifiable from observations.

For this thesis, the most important insight is the distinction between belief updating and planning. For MAR missingness, including MCAR, the precise probabilities in $M$ can cancel in the Bayes belief update. Thus, when executing a fixed policy, the agent may be able to maintain the current belief without knowing the exact missingness probabilities. However, computing a belief-based policy still requires the missingness function, because policy computation uses probabilities over successor beliefs, and those probabilities depend on which observations are

likely to occur. In short, missingness can be ignorable for belief maintenance while still being non-ignorable for planning.

This distinction is the part of the miss-MDP paper that we use in Section 4.3. Our setting is different from the one studied by J. Wendland *et al.*: we do not estimate an unknown missingness function from histories, and we do not compute an optimal policy for the same miss-MDP in which the agent will later act. Instead, the missingness mechanism is part of the training data generation process, and the learned AFA policy is evaluated in a different problem where all features are acquirable. The miss-MDP framework is therefore useful not because we solve their learning problem, but because it cleanly separates posterior effects from planning and control effects under missing observations.

## 3.4 Active Feature Acquisition (AFA)

Active Feature Acquisition (AFA) is a subfield of machine learning concerned with learning to acquire only a subset of (informative) features for prediction under cost constraints. The AFA problem setting can be formalized in various ways; here, we describe a common formulation adopted from [12].

Let $\mathcal{X} := \bigtimes_{j=1}^{n} \mathcal{X}_j$ be the collective feature space, and let $\mathcal{D} := \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ be a dataset of $N$ data instances with $x^{(i)} \in \mathcal{X}$ and $y^{(i)} \in \mathcal{Y}$. Furthermore, we assume that the samples are distributed according to some unknown distribution $p(x, y)$.

For any subset $\mathcal{S} \subseteq [n] := \{1, ..., n\}$, we write $x_\mathcal{S} \in \mathcal{X}_\mathcal{S}$ for the restriction of $x$ to the features indexed by $\mathcal{S}$. For a data instance $x$, acquiring feature $i \in [n]$ reveals the $i$-th feature $x_i$ and incurs a cost $c_i \in \mathbb{R}^+$. Furthermore, we denote the total cost of acquiring the features in $\mathcal{S}$ with $c(\mathcal{S}) := \sum_{i \in \mathcal{S}} c_i$. An agent can sequentially query features, incurring costs, until it decides to `STOP` and make a prediction or exhausts a cost budget $b$. The corresponding agent's action space can be defined as follows.

**Definition 3.32** (Action Space in AFA).

Let $\mathcal{U} := [n] \setminus \mathcal{S}$ denote the set of unobserved features. The action space at any point in the acquisition process is then,

$$\mathcal{A}_b(\mathcal{S}) := \{a \in \mathcal{U} : c(\mathcal{S}) + c_a \leq b\} \cup \{\texttt{STOP}\}, \tag{3.1}$$

where $b$ is a per-instance budget that limits the total acquisition cost.

Formally, the AFA problem is to learn a policy $\pi(x_\mathcal{S}, \mathcal{S}) \in \mathcal{A}_b(\mathcal{S})$ and a classifier $f(x_\mathcal{S}, \mathcal{S}) \in \mathcal{Y}$ (possibly jointly) that optimizes a cost-performance trade-off.

**Definition 3.33** (AFA Optimization Objective).

Let $\pi[x] \subseteq [n]$ denote the final acquired feature set under policy $\pi$ on instance $x$. Then, the AFA optimization problem can be written as,

$$\begin{aligned} &\min_{f,\pi} \mathbb{E}_{(x,y)\sim p(x,y)} \left[ \mathbb{E}_{\pi} \left[ \ell\Big( f\Big( x_{\pi[x]}, \pi[x] \Big), y \Big) + \alpha c(\pi[x]) \right] \right] \\ &\text{subject to } c(\pi[x]) \leq b \text{ for all } x \in \mathcal{X}. \end{aligned} \tag{3.2}$$

where $\alpha \geq 0$ controls the cost-accuracy trade-off and $\ell(\cdot, \cdot)$ is the loss function.

The parameter $\alpha$ induces a soft budget and is therefore referred to as a soft-budget parameter. Conversely, the parameter $b$ induces a hard budget and is referred to as a hard-budget parameter. The soft budget formulation is beneficial because it adapts the cost spent per instance. However, selecting an appropriate value of $\alpha$ can be hard or unintuitive in many applications. The hard budget formulation $b$ is more natural when resources are strictly limited per instance and is often easier to interpret. Its drawback is that some instances may require acquiring features whose total cost exceeds $b$ to achieve accurate predictions, or that the budget may be too large for some instances, leading to unnecessary costs.

### 3.4.1 POMDP Formulation

The AFA problem can be formulated as a POMDP. Formally, the AFA POMDP is defined as follows.

**Definition 3.34** (The Active Feature Acquisition (AFA) POMDP).

The AFA POMDP is fully characterized by the tuple $(\mathfrak{S}, \mathcal{A}, \mathcal{X}, \mathcal{Y}, \mathbb{T}, \mathbb{O}, R)$, where $\mathfrak{S} \coloneqq \{(x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}) : \mathcal{S} \in 2^{[n]}, x \in \mathcal{X}, y \in \mathcal{Y}\}$ is the state space, $\mathcal{A}$ is the action space defined in Eq. (3.1), $\mathcal{X}$ is the feature space, $\mathcal{Y}$ is the label space, $\mathbb{T}$ is the transition model, $\mathbb{O}$ is the observation model, and $R$ is the reward function.

The natural decomposition $x \coloneqq (x_{\mathcal{S}}, x_{\mathcal{U}})$ allows us to separate the state into an observed component $(x_{\mathcal{S}}, \mathcal{S})$ and an unobserved component $(y, x_{\mathcal{U}})$. This means that given a state $(x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}) \in \mathfrak{S}$, both the transition model and the observation model are deterministic.

**Definition 3.35** (Transition and Observation Models in AFA).

If the action $a \in \mathcal{U}$ is an acquisition action, then feature $x_a$ is deterministically observed and we transition to the state,

$$\Big( x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}, y, x_{\mathcal{U}\setminus\{a\}} \Big).$$

The AFA POMDP belief state is the posterior distribution over the unobserved features and label given the currently observed features, which is a sufficient statistic for control in this setting.

**Definition 3.36** (Belief State in AFA).

The AFA POMDP belief state is defined as,

$$b(x_{\mathcal{S}}, \mathcal{S}) := (x_{\mathcal{S}}, \mathcal{S}, p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S})).$$

Thus, the probability of observing $o \in \mathcal{X}_a$ after taking action $a \in \mathcal{U}$ in the current belief state is,

$$\begin{aligned} p(x_a = o \mid x_{\mathcal{S}}, \mathcal{S}) &= \mathbb{E}_{y \mid x_{\mathcal{S}}, \mathcal{S}}[p(x_a = o \mid x_{\mathcal{S}}, \mathcal{S}, y)] \\ &= \mathbb{E}_{x_{\mathcal{U}\setminus\{a\}}, y \mid x_{\mathcal{S}}, \mathcal{S}}\left[p\Big(x_a = o \;\Big|\; x_{\mathcal{S}}, \mathcal{S}, x_{\mathcal{U}\setminus\{a\}}, y\Big)\right] \end{aligned}$$

Furthermore, after taking action $a \in \mathcal{U}$ and observing $o = x_a \in \mathcal{X}_a$, the updated partial-information state is $\big(x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\big)$, and the belief state becomes,

$$b\big(x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\big) := \Big(x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}, p\Big(y, x_{\mathcal{U}\setminus\{a\}} \;\Big|\; x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\Big)\Big).$$

Equivalently,

$$p\Big(y, x_{\mathcal{U}\setminus\{a\}} \;\Big|\; x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\Big) = p\Big(y, x_{\mathcal{U}\setminus\{a\}} \;\Big|\; x_{\mathcal{S}}, \mathcal{S}, x_a = o\Big).$$

Thus, the updated belief can be obtained via Bayesian updating. For example, for the label $y$, the update is given by,

$$p\Big(y \;\Big|\; x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\Big) = \frac{p(x_a = o \mid x_{\mathcal{S}}, \mathcal{S}, y) p(y \mid x_{\mathcal{S}}, \mathcal{S})}{p(x_a = o \mid x_{\mathcal{S}}, \mathcal{S})},$$

which is the standard belief update rule in POMDPs [28].

Finally, we can model the acquisition cost and the prediction loss of AFA as (negative) rewards in the POMDP framework.

**Definition 3.37** (Reward Structure in AFA).

For any state $(x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}) \in \mathfrak{S}$, we assume that the acquisition action $a \in \mathcal{U}$ yields immediate reward $R((x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}), a) := -\alpha c_a$ (i.e., the acquisition cost), while the stop action $a = \texttt{STOP}$ yields terminal reward $R((x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}), \texttt{STOP}) := -\ell(f(x_{\mathcal{S}}, \mathcal{S}), y)$ (i.e., the negative loss of the prediction).

For the stopping action, the expected reward in the current belief state $b(x_{\mathcal{S}}, \mathcal{S})$ is given by,

$$R(b(x_{\mathcal{S}}, \mathcal{S}), \texttt{STOP}) := \mathbb{E}_{y|x_{\mathcal{S}},\mathcal{S}}[-\ell(f(x_{\mathcal{S}}, \mathcal{S}), y)].$$

Similarly, for any action $a \in \mathcal{U}$, we have $R(b(x_{\mathcal{S}}, \mathcal{S}), a) := -\alpha c_a$.

### 3.4.2 From POMDP to MDP

Commonly in the AFA literature, the POMDP formulation is simplified to a fully observable MDP by treating the belief state as the state of the MDP and directly conditioning on the observed features instead of the belief state. The first step of this transformation is that any POMDP can be rewritten as a fully observable belief-MDP whose state is the current belief [34].

**Definition 3.38** (Truncated Optimal Value and Action-Value Functions for the AFA belief-MDP).

Let $V_k(b(x_{\mathcal{S}}, \mathcal{S}))$ denote the optimal truncated value function when at most $k$ additional acquisition actions may be taken before `STOP` must be selected. The base case is,

$$V_0(b(x_{\mathcal{S}}, \mathcal{S})) := R(b(x_{\mathcal{S}}, \mathcal{S}), \texttt{STOP}).$$

For $k \geq 1$, define the corresponding truncated action values by,

$$Q_k(b(x_{\mathcal{S}}, \mathcal{S}), \texttt{STOP}) := V_0(b(x_{\mathcal{S}}, \mathcal{S})),$$
$$Q_k(b(x_{\mathcal{S}}, \mathcal{S}), a) := -\alpha c_a + \mathbb{E}_{x_a|x_{\mathcal{S}},\mathcal{S}}\left[V_{k-1}\left(b\left(x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\right)\right)\right]$$

for each acquisition action $a \in \mathcal{A}_b(\mathcal{S}) \setminus \{\texttt{STOP}\}$.

The truncated value function is then,

$$V_k(b(x_{\mathcal{S}}, \mathcal{S})) := \max_{a \in \mathcal{A}_b(\mathcal{S})} Q_k(b(x_{\mathcal{S}}, \mathcal{S}), a).$$

Intuitively, this captures the question "Is it better to stop now or acquire another feature?". G. Dulac-Arnold, L. Denoyer, and P. Gallinari were the first to prove that this is equivalent to the original AFA objective in Eq. (3.2).

The last step of the transformation is to note that, in this AFA setting, there are no hidden temporal dynamics. Normally, the belief state would be dependent on the history. However, here $(x_{\mathcal{S}}, \mathcal{S})$ is a sufficient statistic for the belief state because the underlying instance is static and does not change over time. Thus, we can define a fully observable MDP with state space $\{(x_{\mathcal{S}}, \mathcal{S}) : \mathcal{S} \subseteq [n], x_{\mathcal{S}} \in \mathcal{X}_{\mathcal{S}}\}$, and the corresponding Bellman equations remain the same as in the belief-MDP, except that we can directly condition on the observed state instead of the belief

state. However, the AFA (PO)MDP is known to be highly intractable, motivating the development of myopic heuristics [12].

# 4

# Methods

This chapter presents the **AFABench** framework, the synthetic **CUBE-NM** dataset, and our formulation of AFA under missing data, along with theoretical insights about the effects of missing data in AFA.[1] We begin with the design of the **AFABench** framework in Section 4.1. Section 4.2 presents the **CUBE-NM** dataset and its theoretical properties that make it suitable for evaluating myopic and non-myopic methods. Section 4.3 develops the AFA formulation under missing data and presents the XOR example to illustrate the effects of missing data in AFA. Section 4.4 then describes how the **AFABench** framework introduces missing data along with the adaptations for three included methods.

## 4.1 AFABench: A General Framework for Evaluating AFA Methods

To evaluate AFA methods, a clear specification of the input-output structure of the evaluation protocol is necessary. Figure 4.1 illustrates **AFABench**'s proposed components and their interactions for a single evaluation episode.

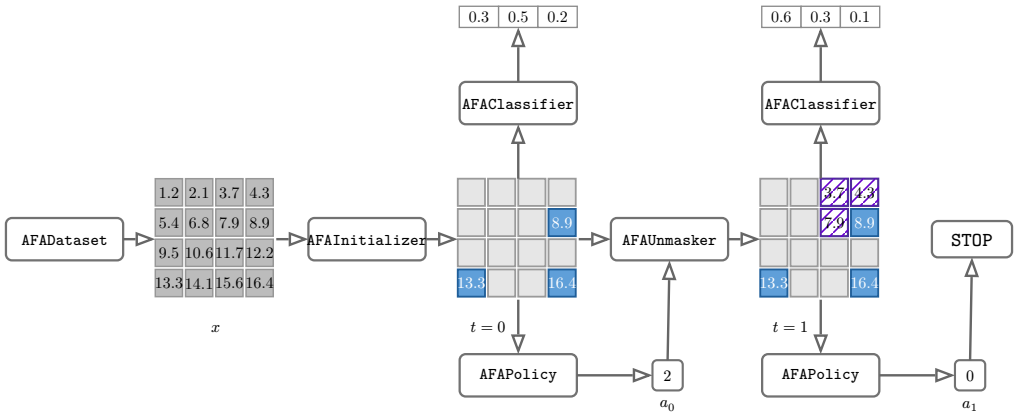



**Figure 4.1:** Visualization of one **AFABench** evaluation episode. The **AFADataset** provides a sample $x$, the **AFAInitializer** chooses how to initialize the sample, the **AFAPolicy** selects an action $a_t$, and the **AFAUnmasker** reveals the selected feature(s) before the policy eventually stops or exhausts the budget.

[1]The **AFABench** framework and the synthetic **CUBE-NM** dataset are presented in [36].

The components of the AFA problem in the benchmark interact episodically with each data instance $x$. Initially, the `AFADataset` provides a data instance $x$ with corresponding target label $y$. Then, the `AFAInitializer` determines how to initialize the sample, using either deterministic or stochastic rules. The resulting sample is then passed in parallel to the `AFAClassifier` and `AFAPolicy`. The `AFAClassifier` produces intermediate predictions $\hat{y}_t$ for subsequent performance evaluation and plays no role in episode dynamics. In parallel, the `AFAPolicy` outputs an action $a_t$, interpreted either as a `STOP` decision or as a request to reveal and acquire additional feature(s).

The framework does not assume a one-to-one correspondence between actions and features to reveal. Instead, this mapping is handled by a separate component, the `AFAUnmasker`, which translates actions into the next set of feature(s) to reveal. In Figure 4.1, at time step $t = 0$, the `AFAPolicy` outputs $a_0 = 2$, which in this specific example corresponds to the upper-right $2 \times 2$ pixel patch. After the `AFAUnmasker` reveals the selected feature(s), the time index increments and the episode continues until a `STOP` action is selected or the feature budget is exhausted by the `AFAPolicy`.

Each action selected by the `AFAPolicy` incurs a cost that depends on both the specific `AFADataset` and `AFAUnmasker`, as illustrated in Figure 4.2. Each `AFADataset` specifies a per-feature cost, which may be uniform or non-uniform. Since one action may reveal multiple features, the `AFAUnmasker` computes the average cost of the revealed features and adds it to the average cumulative cost of the episode. If the `AFAPolicy` outputs an action that would exhaust the budget, the action is overridden and treated as a `STOP`.

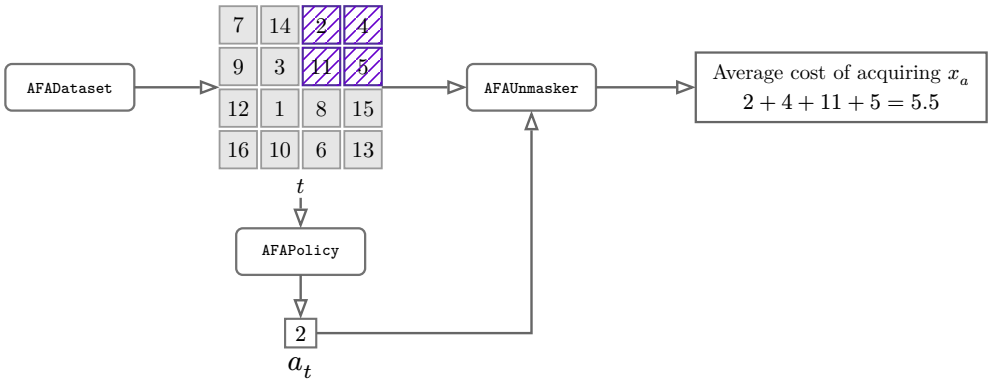



**Figure 4.2:** An illustration of how a `AFAUnmasker` could calculate the cost of an action.

### 4.1.1 Flexible Connectors: Accommodating Diverse Methods

Some AFA methods estimate feature informativeness using generative models such as PVAEs that require pretraining, while others learn acquisition behavior through model-free RL that requires joint training of the classifier and policy. Furthermore, some methods do not require explicit training because their core computations are performed at evaluation. This requires a flexible pipeline structure that can

accommodate pretraining of components, joint or separate training of the classifier and policy, or no training at all. `AFABench` respects these differences by allowing methods to implement only the stages they need as illustrated in Figure 4.3.

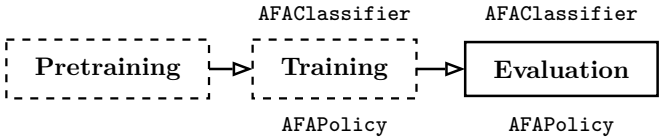



**Figure 4.3:** The three pipeline stages illustrated for `AFABench`. The optional stages of the pipeline are indicated by dotted lines.

Since `AFABench` is an evaluation framework, the only required stage is the evaluation stage, where the method's acquisition policy is evaluated on the test set. As long as the method respects the shared episode structure and interfaces during evaluation, it can implement the training and pretraining stages in any way that is compatible with its underlying learning algorithm. In other words, as long as a feasible `AFAPolicy` and `AFAClassifier` can be implemented for the evaluation stage, the method is free to choose how to train those components, or whether to train them at all.

Furthermore, the hard budget parameter $b$ and the soft-budget parameter $\alpha$ are specified at both training and evaluation time (possibly with different values). This is necessary because some methods use only the soft-budget parameter during training, while others use it only at evaluation time. More generally, arbitrary soft/hard-budget choices can be combined across training and evaluation.

### 4.1.2 Batteries Included: `AFABench`'s Included Methods and Datasets

The `AFABench` framework includes implementations of the methods presented in Table 4.1. The methods can be categorized into three main groups based on their underlying strategy for feature acquisition:

1. Myopic methods, which optimize one-step feature utility, often through a conditional mutual information (CMI) criterion,

**Definition 4.1** (Conditional Mutual Information (CMI) Policy).

A myopic CMI policy is any policy that implements the rule,

$$\pi_{\text{CMI}}(x_{\mathcal{S}}, \mathcal{S}) \in \underset{a \in \mathcal{A}_b(\mathcal{S}) \setminus \{\texttt{STOP}\}}{\arg\max} \frac{I(y; x_a \mid x_{\mathcal{S}}, \mathcal{S})}{c_a}. \tag{4.1}$$

where $\mathcal{A}_b(\mathcal{S})$ is the set of legal actions under budget $b$ given current acquisitions $\mathcal{S}$, $I(\cdot;\cdot)$ denotes mutual information, and $c_a$ is the cost of acquiring feature $a$.

2. Non-myopic RL-based methods, which exploit the sequential decision structure of AFA, and
3. Non-myopic methods that do not explicitly rely on RL.

**Table 4.1:** Taxonomy of the methods included in `AFABench`.

| Strategy | Myopic? | Method(s) |
|---|---|---|
| Generative CMI | Yes | `EDDI` |
| Discriminative CMI | Yes | `GDFS`, `DIME` |
| Model-free RL | No | `JAFA`, `OL`, `ODIN`, |
| Model-based RL | No | `ODIN` |
| Oracle-based planning | No | `AACO` |

All myopic methods in the benchmark implement the rule in Eq. (4.1) in some form, but they differ in how the required conditional distributions are estimated. In hard-budget evaluation, they continue acquiring until no further legal action in $\mathcal{A}_b(\mathcal{S}) \setminus \{\texttt{STOP}\}$ remains or the policy explicitly stops. In soft-budget evaluation, they use threshold-based stopping, but the exact thresholding rule is method dependent because the methods expose different uncertainty surrogates.

`EDDI` is a generative myopic method based on PVAEs [13]. A pretrained PVAE models the conditional distribution of unobserved features given $x_{\mathcal{S}}$ and thereby estimates the one-step label-information gain of candidate acquisitions $I(y; x_a \mid x_{\mathcal{S}}, \mathcal{S})$ for each $a \in \mathcal{U}$. When the relevant predictive distributions admit closed-form entropy calculations, the CMI can be computed analytically, otherwise it is approximated by Monte Carlo sampling.

`GDFS` is a discriminative myopic method that learns a one-step acquisition rule by optimizing expected predictive improvement after revealing one additional feature [14]. Under the Bayes' classifier, this one-step loss-reduction objective yields the same greedy choice as the CMI rule. In practice, the network outputs the next action directly rather than a calibrated value for every candidate feature. Because of that design, the soft-budget adaptation uses a stopping rule based on classifier entropy $H(y \mid x_{\mathcal{S}}, \mathcal{S})$ rather than a direct threshold on estimated CMI values.

`DIME` estimates the conditional mutual information directly in a discriminative way [15]. The implementation uses a classifier $f_\theta$ together with a value network $g_\varphi$ that approximates $I(y; x_a \mid x_{\mathcal{S}}, \mathcal{S})$ for each candidate $a \in \mathcal{U}$. When the classifier approaches Bayes' optimality, the expected cross-entropy reduction from observing $a$ matches the corresponding CMI, which yields a consistent training signal for $g_\varphi$. Among the discriminative myopic methods, this makes `DIME` the closest direct implementation of Eq. (4.1).

RL-based methods inherit the AFA MDP formulation from Section 3.4.2. In practice, hard-budget evaluation does not allow early stopping by restricting the corresponding MDP's action space. In the soft-budget evaluation, it introduces an explicit or implicit acquisition penalty through the soft-budget parameter $\alpha$.

**JAFA** is a model-free RL method that learns an acquisition policy over sequential AFA states using an order-invariant set encoder and deep Q-learning [16]. The classifier and policy are trained jointly so that the value function is aligned with the final classification objective. The implementation of the reward follows the (sparse terminal-loss) formulation from Section 3.4.1, making **JAFA** a direct long-horizon counterpart to the one-step myopic methods.

**ODIN** is a model-based RL method that uses a pretrained PVAE to approximate the conditional distribution of unobserved features and then performs model-based rollouts for policy learning [18]. This turns the generative model into an approximate transition model for the AFA MDP and can improve data efficiency on smaller datasets. The method uses a dense reward that favors intermediate predictive improvement,

$$R((x_{\mathcal{S}}, \mathcal{S}), a) := p_f\Big(y_{\text{true}} \mid x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}\Big) - \alpha c_a.$$

Compared with the sparse terminal-loss objective, this reward can improve optimization stability, but it also biases the method toward more immediately informative actions.

**ODIN** also has a model-free variant that uses the same reward but does not perform model-based rollouts. It keeps the same overall acquisition framing, but removes model-generated rollouts and trains only on real trajectories from the original dataset.

**OL** is a model-free DQN-based method derived from the opportunistic-learning formulation of [17]. Its reward is based on predictive-confidence improvement per unit cost rather than on the true label, so it can be trained without an explicit label-conditioned reward at every step.

**AACO** treats AFA as a subset-optimization problem rather than a one-step feature-ranking problem [19]. Starting from the current state $(x_{\mathcal{S}}, \mathcal{S})$, it evaluates candidate subsets $\mathcal{T} \subseteq \mathcal{U}$ by approximating the expected post-acquisition prediction loss plus acquisition cost. The required conditional distributions are estimated non-parametrically with $k$-nearest-neighbor density estimation, and only a sampled subset of all possible feature subsets is evaluated in high-dimensional settings. This yields non-myopic behavior without training a full RL policy.

**AFABench** includes synthetic datasets, real-world datasets, and one image dataset to test the methods across a range of settings. The synthetic suite consists of **CUBE**, its non-uniform-cost variant **CUBE-NUC**, the novel proposed **CUBE-NM**, and a noiseless **CUBE-NM** variant used for controlled analysis.

The real-world tabular datasets include **Diabetes**, **PhysioNet**, **MiniBooNE**, **ACTG175**, **CKD**, and **BankMarketing**. Additionally, **MNIST** [24] and **FashionMNIST** [37] are implemented as tabular data, treating each pixel as a separate feature.

Finally, the `Imagenette` [38] dataset is included as a patch-based image-acquisition task. Dataset sizes and preprocessing notes are summarized in Table A.1 and described in Section A.3.

### 4.1.3 Comparing Apples to Apples: Fair Evaluation Protocol

A benchmark is only useful when differences in performance can be attributed to the acquisition strategy rather than to incidental implementation choices. For this reason, `AFABench` fixes the episode interface before comparing methods. All methods receive the same train/validation/test splits, the same `AFAInitializer`, the same `AFAUnmasker`, the same feature costs, and the same budget semantics. All reported results are averaged over multiple data splits and random seeds, with variability reported alongside mean performance. The `AFAPolicy` is therefore always evaluated as a policy for the same AFA problem, even when the internal learning algorithms differ substantially.

Hard-budget and soft-budget settings are evaluated separately. In the hard-budget setting, the main question is how well a method uses a fixed acquisition allowance. In the soft-budget setting, the method must also decide when further acquisitions are no longer worth their cost. Combining these settings would mix feature-ranking ability with stopping behavior and make the resulting comparison difficult to interpret.

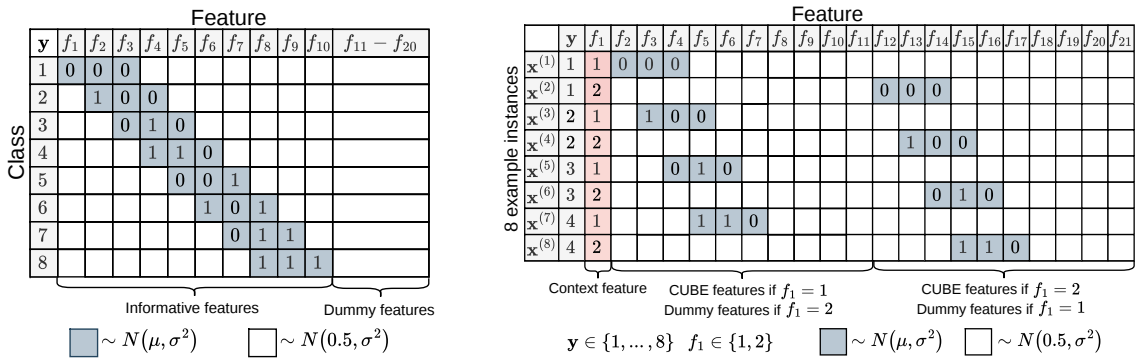
The benchmark also separates acquisition quality from the `AFAClassifier` quality as far as possible. Some methods jointly learn a classifier and an acquisition policy, while others assume an external classifier. Therefore, `AFABench` reports performance with a shared external classifier and additionally reports internal-classifier performance when the method naturally provides one. This makes it possible to see whether a method acquired useful subsets, rather than only whether its own predictor happened to be stronger.

The included components in `AFABench` are kept fixed whenever this does not change the method being evaluated. For example, the methods that rely on a PVAE use the same pretrained model, and RL methods share the same environment interface, cost accounting, and rollout infrastructure. The remaining differences are the intended ones, the acquisition objective, the policy class, the reward design, and any method-specific approximation such as model-based rollouts or subset scoring.

## 4.2 `CUBE-NM`: A Novel Synthetic Dataset for AFA

We now introduce `CUBE-NM`, a dataset for benchmarking myopic against non-myopic AFA methods. We first introduce the original `CUBE`, shown in Figure 4.4. `CUBE` consists of 20-dimensional real-valued vectors from 8 classes. For each class, three informative features are sampled from $\mathcal{N}(\mu, \sigma^2)$ with class-specific means, while remaining features are sampled as noise from $\mathcal{N}(0.5, \sigma^2)$. This structure

allows instance-specific acquisition but offers limited separation between myopic and non-myopic methods.




(a) `CUBE`. (b) `CUBE-NM` for $n_c = 2$.

**Figure 4.4:** Visualization of (a) `CUBE`, and (b) the proposed synthetic dataset `CUBE-NM`.

`CUBE-NM` extends `CUBE` by adding a context feature that selects which feature block is informative, while the context itself is not predictive under myopic criteria. Each instance has integer context feature $f_1 \in \{1, ..., n_c\}$. For each context $j \in [n_c]$, we associate a block $b_j = \left\{ f_{1+10(j-1)+r} : r = 1, ..., 10 \right\}$. Conditioned on $f_1 = j$, block $b_j$ follows CUBE-like informative/noise structure. The total dimensionality is $d := 1 + 10n_c$.

This creates a setting where myopic methods struggle. The key first step is to acquire $f_1$, but $f_1$ has no immediate label information $I\left(y; x_{f_1}\right) = 0$. As long as other candidate features have positive one-step mutual information, the rule in Eq. (4.1) tends to avoid $f_1$ and instead keeps probing blocks. Non-myopic methods can pay a small upfront cost to observe $f_1$ and then route subsequent acquisitions to the informative block.

**Theorem 4.1** (Cube-NM Formal Properties).

Consider `CUBE-NM` with $n_c$ contexts in the noiseless setting ($\sigma = 0$), so that each instance $x \in \mathcal{X}$ determines the label exactly, i.e., $H(y \mid x) = 0$. Let $f(x_\mathcal{S}, \mathcal{S}) := p(y \mid x_\mathcal{S}, \mathcal{S})$ be a Bayes' optimal classifier and assume uniform feature costs, $c_i = 1$ for all $i \in [d]$. Let $\Pi$ denote the set of policies that keep acquiring features until $H(y \mid x_\mathcal{S}, \mathcal{S}) = 0$. First, define the optimal policy as,

$$\pi^\star \in \arg\min_{\pi \in \Pi} \mathbb{E}_x \mathbb{E}_\pi [c(\pi[x])].$$

Then the optimal policy satisfies,

$$\mathbb{E}_x \mathbb{E}_{\pi^\star}[c(\pi^\star[x])] = \mathbf{1}\{n_c \geq 2\} + 2.25,$$
$$\min_{x \in \mathcal{X}} c(\pi^\star[x]) = \mathbf{1}\{n_c \geq 2\} + 1,$$
$$\max_{x \in \mathcal{X}} c(\pi^\star[x]) = \mathbf{1}\{n_c \geq 2\} + 3.$$

Conversely, let $\pi_{\text{CMI}}$ be the myopic CMI policy in Eq. (4.1) with uniform tie-breaking over CMI-maximizing acquisition actions. In particular, at the initial state it chooses uniformly among the CMI-maximizing block-feature actions, and when several untouched blocks remain tied for the currently selected coordinate, it chooses uniformly among those tied block actions. Then,

$$\mathbb{E}_x \mathbb{E}_{\pi_{\text{CMI}}}[c(\pi_{\text{CMI}}[x])] = 13\frac{2n_c + 1}{16},$$
$$\min_{x \in \mathcal{X}} c(\pi_{\text{CMI}}[x]) = 1,$$
$$\max_{x \in \mathcal{X}} c(\pi_{\text{CMI}}[x]) = 3n_c.$$

The full proof is given in Section A.2, but the key intuition is that the myopic policies will always miss the context feature $f_1$ at the first step (for $n_c \geq 2$), and then they keep acquiring features from random blocks until they happen to hit the informative block by chance. For $n_c \geq 2$, this gives an expected cost of $1 + 2.25$ for the optimal context-first policy, one query for $f_1$, followed by the optimal noiseless `CUBE` policy inside the active block. The myopic policy, in contrast, has an expected cost that grows linearly with $n_c$ because it keeps probing blocks until it finds the informative one.

## 4.3 Active Feature Acquisition under Missing Data

This section develops our formulation of AFA under missing data and presents theoretical insights about the effects of missingness on AFA. As discussed in Section 1.1 and Section 1.2, we primarily study the setting where missing data is present during training, while evaluation follows the standard AFA protocol with full feature availability.

### 4.3.1 Connection to miss-MDPs

The miss-MDP framework in Section 3.3.6 gives a precise point of comparison for AFA under training missingness. In both settings, incomplete observations are not treated as arbitrary noise, but as the result of an explicit missingness mechanism. The difference is the problem being solved. In a miss-MDP, the goal is to model or estimate the missingness function well enough to compute a policy for the same partially observed decision process. Here, the missingness mechanism is specified by the training initializer, and the learned AFA policy is later evaluated in the ordinary AFA problem where all features are acquirable.

Thus, our setting can be viewed as a static training-time version of the miss-MDP idea. The latent state is a data instance and label, the agent gradually reveals features, and a fixed missingness mask determines which features are unavailable for that training instance. The central question is not whether we can learn the missingness function. It is whether training on a restricted acquisition problem transfers to evaluation on the unrestricted acquisition problem. This makes AFA under training missingness a useful case where belief ignorability and control optimality must be considered separately.

### 4.3.2 AFA POMDP with Training Missingness

Consider a data instance $x \in \mathcal{X}$ with target variable $y \in \mathcal{Y}$, distributed according to $p(x, y)$. Let $m \in \{0,1\}^n$ denote the corresponding binary missingness mask, where $m_i = 1$ indicates that feature $i$ is missing and not acquirable during training, while $m_i = 0$ indicates that feature $i$ is acquirable. The realized mask is sampled according to a missingness distribution $p(\boldsymbol{m} \mid \boldsymbol{x})$ and remains fixed for the corresponding data instance. One can extend Definition 3.34 to model this setting by including the missingness mask $m$ in the state and observation spaces, and by modifying the action space to only allow acquisition of features that are not missing.

**Definition 4.2** (The AFA POMDP under missing data).

The AFA problem under missing data can be formalized with the POMDP $\mathcal{P}_b^{\text{train}}$ with state space,

$$\mathfrak{S}^{\text{train}} := \left\{(x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}, m) : \mathcal{S} \in 2^{[n]}, x \in \mathcal{X}, y \in \mathcal{Y}, m \in \{0,1\}^n\right\},$$

action space,

$$\mathcal{A}_b^{\text{train}}(\mathcal{S}, m) := \{a \in \mathcal{U} : m_a = 0, c(\mathcal{S}) + c_a \leq b\} \cup \{\texttt{STOP}\}.$$

observation space,

$$\mathcal{O}^{\text{train}} := \left\{(x_{\mathcal{S}}, \mathcal{S}, m) : \mathcal{S} \in 2^{[n]}, x \in \mathcal{X}, m \in \{0,1\}^n\right\}.$$

For $a \in \mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \setminus \{\texttt{STOP}\}$ the transition function is deterministically defined as,

$$(x_{\mathcal{S}}, \mathcal{S}, y, x_{\mathcal{U}}, m) \mapsto \left(x_{\mathcal{S}\cup\{a\}}, \mathcal{S} \cup \{a\}, y, x_{\mathcal{U}\setminus\{a\}}, m\right).$$

The reward structure is the same as in the standard AFA POMDP: each non-stop action incurs its acquisition cost and `STOP` yields the terminal prediction reward. The corresponding evaluation POMDP recovers Definition 3.34, since the missingness mask $m$ is not present during evaluation and all features are acquirable.

One can define this POMDP as a fully observable MDP on the observed state space $\mathcal{O}^{\text{train}}$.

**Proposition 4.1** (Equivalent observed-state MDP under missingness).

Because the latent variables $(x, y, m)$ do not evolve over time for a given realization, the current observation $o_t \in \mathcal{O}^{\text{train}}$ is a sufficient state variable for control. Hence $\mathcal{P}_b^{\text{train}}$ admits an equivalent fully observable MDP on $\mathcal{O}^{\text{train}}$.

The process in Definition 4.2 can therefore also be viewed as a miss-MDP with static latent state and instance-wise missingness. However, we present it in AFA notation to stay aligned with the standard AFA POMDP formulation.

### 4.3.3 Missingness Mechanisms and the XOR Example

The POMDP definition above is agnostic to the type of missingness. The distinction between MCAR, MAR, and MNAR enters only through the missingness mechanism $p(\boldsymbol{m} \mid \boldsymbol{x})$. Let $\mathcal{J} \coloneqq \{i \in [n] : m_i = 0\}$ denote the set of indices of the acquirable features for a data instance $(x, y, m)$. Thus, the missingness mechanisms can be defined as follows.

- MCAR if $p(m \mid x) = p(m)$.
- MAR if $p(m \mid x) = p(m \mid x_{\mathcal{J}})$.
- MNAR otherwise.

We now study a simple example that makes the effect of missingness on AFA clear.

**Example 4.1** (The Noisy XOR Problem).

Consider three binary features $x_1, x_2, x_3 \in \{0, 1\}$ with $x_1$ and $x_2$ drawn independently and uniformly with cost $c_1 = c_2 = 1$. The target label is $y \coloneqq \text{XOR}(x_1, x_2)$ and $x_3$ is a "noisy" copy of the label satisfying $P(x_3 = y) = 0.51$ with cost $c_3 = 2$. Only $x_2$ can be missing and $m_2$ is sampled according to one of the missingness mechanisms,

- **MCAR rule**: $P(m_2 = 1 \mid x) = \frac{1}{2}$ for all $x$.
- **MAR rule**: $P(m_2 = 1 \mid x) = \frac{3}{4}$ if $x_1 = 0$, and $P(m_2 = 1 \mid x) = \frac{1}{4}$ if $x_1 = 1$.
- **MNAR rule**: $P(m_2 = 1 \mid x) = \frac{3}{4}$ if $x_2 = 0$, and $P(m_2 = 1 \mid x) = \frac{1}{4}$ if $x_2 = 1$.

A key property of the noisy XOR problem is that each input alone is marginally independent of the output, so neither $x_1$ nor $x_2$ alone is informative about $y$. Under no missingness, acquiring both $x_1$ and $x_2$ determines $y$ perfectly at total cost 2, while the shortcut $x_3$ costs 2 but yields only 51% accuracy.

Under the MCAR rule, the event $m_2 = 1$ is independent of $(x_1, x_2, y)$, so acquiring $x_1$ still leaves the label completely uncertain. Hence the optimal prediction after

acquiring $x_1$ has accuracy $\frac{1}{2}$. Therefore, at the blocked initial state, querying $x_3$ is preferable.

Under the MAR rule, the event $m_2 = 1$ is informative about $x_1$ but once $x_1$ has been acquired it gives no additional information about $x_2$. Thus, querying $x_3$ is preferable in this case as well.

Under the MNAR rule, however, the blocked event itself is informative about the hidden feature. By Bayes' rule,

$$P(x_2 = 0 \mid m_2 = 1) := \frac{P(m_2 = 1 \mid x_2 = 0)P(x_2 = 0)}{P(m_2 = 1)} = \frac{\left(\frac{3}{4}\right)\left(\frac{1}{2}\right)}{\left(\frac{3}{4}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{4}\right)\left(\frac{1}{2}\right)} = \frac{3}{4}.$$

Consequently, after acquiring $x_1$ we obtain,

$$P(y = x_1 \mid x_1, m_2 = 1) = P(x_2 = 0 \mid m_2 = 1) = \frac{3}{4},$$

since $y = \mathrm{XOR}(x_1, x_2)$ equals $x_1$ exactly when $x_2 = 0$. Thus, on blocked episodes the first acquisition $x_1$ yields 75% accuracy and is preferable to the noisy shortcut $x_3$.

This example shows that missingness can affect AFA in two different ways. First, blockedness can change the current posterior. Second, even when the current posterior is unchanged, blocked actions can still change the best acquisition because they remove future follow-up actions. The rest of this section analyzes these two effects directly in the AFA formulation.

### 4.3.4 State-Dependent Ignorability

The first question concerns belief updating at a fixed partial-information state. J. Wendland *et al.* address this question for miss-MDPs, showing that under MCAR and MAR-type assumptions missingness can be ignorable for belief updating. In AFA, the analogous question is more local. At a given partial-information state, does the blockedness of a currently unacquired feature still provide information about the label and the remaining unacquired features?

**Lemma 4.1** (Local Ignorability of a Blocked Feature).

Let $(x_\mathcal{S}, \mathcal{S})$ and $i \in \mathcal{U}$ denote the current partial-information state and feature index, respectively. If the currently acquired information in $(x_\mathcal{S}, \mathcal{S})$ fully explains the missingness of feature $i$ and blockedness carries no additional information about the label nor the remaining unacquired features, in the sense that

$$p(m_i = 1 \mid y, x_\mathcal{U}, x_\mathcal{S}, \mathcal{S}) = p(m_i = 1 \mid x_\mathcal{S}, \mathcal{S}),$$

then, whenever the conditional distribution is well-defined, conditioning on the blocked event does not change the posterior over the label and the remaining unacquired features,

$$p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S}, m_i = 1) = p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S}).$$

If the currently acquired information already explains why feature $i$ is blocked, then observing $m_i = 1$ does not further change the current belief state. In that case, blockedness is ignorable for the current belief update. Under the MAR rule of Example 4.1, once $x_1$ has been acquired, the blockedness of $x_2$ depends only on an already observed variable, so $m_2 = 1$ is locally ignorable and the posterior over $y$ is unchanged. Conversely, under the MNAR rule, the blockedness of $x_2$ depends on $x_2$ itself, so $m_2 = 1$ is informative about a missing feature and changes the posterior over $y$.

### 4.3.5 Belief Updating and Optimal Actions

Lemma 4.1 identifies when blockedness changes the current posterior at the current state. We now ask a different question. Even if training and evaluation agree at the current state, can they still prefer different acquisitions? In AFA, the answer depends on horizon. A one-step method compares actions only through the immediate value of the next acquisition. A multi-step method also values the follow-up acquisitions that remain available after that choice. This can be analyzed directly through the truncated values in Definition 3.38.

**Proposition 4.2** (One-Step Values Agree on Common Actions).

Let $Q_1^{\text{train}}$ and $Q_1^{\text{eval}}$ denote the one-step truncated action values in the training and evaluation problems, respectively. Fix a realized training mask $m$ and a training observation state $(x_{\mathcal{S}}, \mathcal{S}, m)$. Let $b^m$ and $b$ denote the corresponding current beliefs in the training and evaluation problems. Suppose that for every action $a \in \mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \setminus \{\texttt{STOP}\}$,

$$p(y, x_a \mid x_{\mathcal{S}}, \mathcal{S}, m) = p(y, x_a \mid x_{\mathcal{S}}, \mathcal{S}).$$

meaning that the current posterior over the label and the next acquired feature $x_a$ is unchanged by conditioning on the blockedness of the currently unacquired features. Then for every such action,

$$Q_1^{\text{train}}(b^m, a) = Q_1^{\text{eval}}(b, a).$$

Consequently, any one-step method assigns the same ranking to the actions that are feasible in both problems.

Proposition 4.2 identifies what is special about one-step methods. If blockedness does not change the relation between the label and the next acquired feature at the current state, then the remaining one-step values agree. The only discrepancy is that some evaluation actions may be unavailable during training.

For multi-step methods, the value of an acquisition depends not only on the immediate reward and the next acquired feature, but also on the follow-up acquisitions that remain available after that choice. Even when the current posterior and the remaining one-step values agree, the value of a currently feasible action can still change because its follow-up options differ between training and evaluation.

**Proposition 4.3** (Multi-Step Values Can Still Differ).

Let $V_k^{\text{eval}}$ denote the optimal $k$-step truncated value function under the evaluation action set $\mathcal{A}_b^{\text{eval}}(\mathcal{S})$, and let $V_k^m$ denote the corresponding value function under the training action set $\mathcal{A}_b^{\text{train}}(\mathcal{S}, m)$ for a realized mask $m$. Then for any mask realization $m$, any $k \geq 2$, and any belief state $b$ that is feasible in both problems,

$$V_k^m(b) \leq V_k^{\text{eval}}(b).$$

Equality holds whenever, at every belief reachable within the next $k$ selections, the evaluation problem has an optimal action that also belongs to the training action set. Proposition 4.3 shows why the previous proposition does not extend to planning methods. Even when the current posterior and the remaining one-step values agree, the value of a currently feasible action can still change because its follow-up options differ between training and evaluation.

Under the MCAR rule of Example 4.1, at the blocked initial state Proposition 4.2 applies, so the one-step values are unchanged. Even so, Proposition 4.3 implies that the route through $x_1$ is less valuable during training because it can no longer be followed by $x_2$. The optimal first action can therefore change even though the current posterior is unchanged. Under MNAR, Proposition 4.2 need not apply, because blockedness itself changes the current posterior. The mismatch is then stronger, since both the one-step values and the continuation values can change.

There are therefore two ways in which training with missing data can differ from evaluation. Blockedness can change the current posterior over the missing features and the label, which can change the value of the next acquisition. Even when it does not, blocked actions can reduce the value of acquisition routes that remain feasible at the current state. This can make `STOP` a more feasible choice and can shift the policy away from acquisitions whose value depends on follow-up features that are often unavailable during training. Proofs of the formal results in this section are given in Section A.1.

## 4.4 AFABench and Missing Data

To adapt AFABench from the standard AFA setting to training under missing data, it introduces a new component, the MissingnessInitializer, which samples a missingness mask for each training instance and restricts the training action set accordingly.

**Definition 4.3** (MissingnessInitializer).

For each training instance $j$, MissingnessInitializer samples a missingness mask $m^{(j)} \in \{0,1\}^n$ according to a chosen missingness mechanism. Training starts from the state $\mathcal{S} = \emptyset$, at any later state $\mathcal{S}$, the training action set is restricted to,

$$\mathcal{A}_b^{\text{train}}\big(\mathcal{S}, m^{(j)}\big) := \Big\{a \in \mathcal{U} : m_a^{(j)} = 0, c(\mathcal{S}) + c_a \leq b\Big\} \cup \{\texttt{STOP}\},$$

where $\mathcal{U} := [n] \setminus \mathcal{S}$.

The MissingnessInitializer first generates $m^{(j)}$ using a missingness model. The implementation includes MCAR, MAR, MNAR logistic, MNAR self-masking, and MNAR quantile censoring. In MCAR, each feature is missing independently with probability $p$. In MAR, a fraction $p_{\text{obs}}$ of the features is selected as always available and missingness in the remaining features is sampled from a logistic model depending on those available features. In MNAR logistic, a fraction $p_{\text{params}}$ of the features is selected as driver features for the missingness model. Missingness in the remaining features is sampled from a logistic model depending on those driver features, while the driver features receive MCAR-style missingness.

In the current implementation of AFABench, DIME, AACO, and OL are the methods that have been adapted to the missing data setting, because they represent three different ways of learning acquisition behavior. DIME is a myopic discriminative method that learns a one-step utility estimate that closely resembles the CMI criterion. AACO is non-myopic but not an RL-based method, and it scores candidate feature subsets using an oracle over the training data. OL is a model-free RL method that learns from sequential interaction with the (blocked) training environment. The missingness extensions therefore test whether the issue appears in one-step utility, subset scoring, or sequential RL training (planning), and whether the proposed mitigations are effective in each case.

### 4.4.1 DIME under Missing Data

At every acquisition step, DIME predicts a value for each selection, masks out selections that are already acquired or forbidden, and chooses among the remaining selections by the usual exploration rule. For a training instance $j$ at current feature set $\mathcal{S}$, let $a$ be the selected action and let $\mathcal{S}^+ := \mathcal{S} \cup \{a\}$ be the post-

acquisition feature set. The empirical target is the reduction in classification loss after revealing $a$,

$$\delta_a^{(j)} := \ell\Big(f\Big(x_{\mathcal{S}}^{(j)}, \mathcal{S}\Big), y^{(j)}\Big) - \ell\Big(f\Big(x_{\mathcal{S}^+}^{(j)}, \mathcal{S}^+\Big), y^{(j)}\Big).$$

The value network is trained by regressing its selected prediction $\hat{\delta}_a^{(j)}$ onto this target. The block-only variant uses the ordinary squared-error target, but only along trajectories and actions compatible with the sampled missingness mask.

Inverse Probability Weighting (IPW) is a missing-data and sampling correction that reweights observed terms by the inverse probability of being observed [39].

**Definition 4.4** (Inverse Probability Weighting (IPW)).

Let $L_a$ be a loss term associated with action $a$, and let $z_a \in \{0,1\}$ indicate whether this term is present in the empirical objective. Let $p_a := P(z_a = 1)$ be its inclusion probability, with estimate $\hat{p}_a$. The clipped IPW contribution of this term is defined as,

$$L_a^{\mathrm{IPW}} := z_a w_a L_a, \quad w_a := \min\left(\frac{1}{\max(\hat{p}_a, p_{\min})}, w_{\max}\right).$$

Thus, available actions are weighted by the inverse estimated inclusion probability, while blocked actions are weighted by zero. The constants $p_{\min}$ and $w_{\max}$ clip the weight and prevent extremely rare actions from dominating the mini-batch objective. The IPW mitigation changes only the `DIME` value-network regression loss.

**Definition 4.5** (IPW-weighted `DIME` value loss).

Consider a mini-batch of size $B$. For training instance $j$, let $a_j$ be the selected action and define the ordinary `DIME` squared-error term

$$e_j := \left(\hat{\delta}_{a_j}^{(j)} - \delta_{a_j}^{(j)}\right)^2.$$

The implementation estimates the inclusion probability of any action $a$ by,

$$\hat{p}_a := \frac{1}{B}\sum_{k=1}^{B} \mathbf{1}\left\{a \in \mathcal{A}_b^{\mathrm{train}}\left(\mathcal{S}^{(k)}, m^{(k)}\right)\right\}$$

which is the batch-marginal probability that action $a$ belongs to the current training action set. For each selected action $a_j$, define the clipped inverse-probability weight and its batch-normalized version by

$$v_j := \min\left(\frac{1}{\max\left(\hat{p}_{a_j}, p_{\min}\right)}, w_{\max}\right), \quad \omega_j := \frac{v_j}{\frac{1}{B}\sum_{k=1}^{B} v_k}.$$

The IPW-weighted `DIME` value-network loss is then

$$\mathcal{L}_{\text{DIME-IPW}} := \frac{1}{B}\sum_{j=1}^{B} \omega_j e_j.$$

The implementation normalizes the clipped inverse-probability weights to have mean one in each mini-batch. The predictor loss and the one-step architecture are otherwise unchanged.

### 4.4.2 `AACO` under Missing Data

At evaluation time `AACO` repeatedly scores candidate post-acquisition subsets and chooses the next acquisition implied by the best subset. The missingness variants differ in how the oracle stores the incomplete training data, how it finds nearest neighbors, and how it evaluates candidate losses. Let $h$ denote the `AACO` hide value. For each training instance $j$, let

$$\mathcal{R}_j := \left\{ i \in [n] : m_i^{(j)} = 0 \right\}$$

be the set of acquirable features for that instance. Let $\mathcal{C}$ denote a candidate post-acquisition feature set considered by the `AACO` oracle. The set $\mathcal{C}$ includes the features already acquired in the query state and the additional features proposed by the candidate subset. The simplest mitigation overwrites blocked entries with an internal hide value and then uses the original oracle unchanged, so the oracle receives no information about which features are blocked and treats them as if they were observed with value $h$.

**Definition 4.6** (Zero-filled `AACO`).

Let $h$ be the internal hide value used by the `AACO` oracle, and let $\mathcal{R}_j$ be the acquirable set for training instance $j$. For each training instance $j$, the zero-filled stored representation is defined by,

$$\tilde{x}_i^{(j)} := \begin{cases} x_i^{(j)}, & i \in \mathcal{R}_j \\ h, & i \in [n] \setminus \mathcal{R}_j. \end{cases}$$

Thus, the oracle cannot distinguish a blocked entry from an entry whose value is actually equal to the hide value $h$. Nearest-neighbor search therefore compares a query with current feature set $\mathcal{S}$ to every neighbor over all dimensions in $\mathcal{S}$,

including dimensions that were only filled with $h$. Candidate scoring has the same limitation: if $i \in \mathcal{C}$ but $i \notin \mathcal{R}_j$, the classifier still receives feature $i$ with value $h$.

The mask-aware oracle receives the same stored representation as the zero-filled oracle, but it also uses the missingness mask for each training instance, so it can distinguish between blocked and observed entries. The mask then controls both nearest-neighbor search and candidate-loss evaluation.

**Definition 4.7** (Mask-aware `AACO`).

Let $d$ be the base distance used by the `AACO` nearest-neighbor oracle. For each training instance $j$, define the local shared sets,

$$I_j := \mathcal{S} \cap \mathcal{R}_j, \quad J_j := \mathcal{C} \cap \mathcal{R}_j.$$

The mask-aware neighbor distance is,

$$d_j^{\text{mask}}(\mathcal{S}) := \begin{cases} \frac{d\left(x_{I_j}, \tilde{x}_{I_j}^{(j)}\right)}{|I_j|}, & |I_j| > 0 \\ \infty, & |I_j| = 0. \end{cases}$$

The mask-aware candidate loss is,

$$\hat{L}_{\text{mask}}(\mathcal{C}) := \frac{1}{K} \sum_{j=1}^{K} \ell\left(f\left(\tilde{x}_{J_j}^{(j)}, J_j\right), y^{(j)}\right).$$

Thus, the mask-aware oracle only compares a neighbor on dimensions that exist in both the query and that neighbor. When scoring a candidate, blocked candidate features are hidden and are not marked as acquired for that neighbor. They are therefore not passed to the classifier as acquired hide values.

The direct-plus-IPW mitigation keeps the mask-aware neighbor search and stored representation, but adds a correction to the mask-aware candidate score whenever a neighbor contains the whole candidate set.

**Definition 4.8** (Direct-plus-IPW `AACO`).

For candidate set $\mathcal{C}$, define the estimated full-candidate inclusion probability,

$$\hat{p}(\mathcal{C}) := \prod_{i \in \mathcal{C}} \hat{p}_i, \quad \hat{p}_i := \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\{i \in \mathcal{R}_j\}.$$

Define the clipped inverse-probability weight,

$$w(\mathcal{C}) := \min\left(\frac{1}{\max(\hat{p}(\mathcal{C}), p_{\min})}, w_{\max}\right).$$

The direct-plus-IPW candidate score is then,

$$\hat{L}_{\mathrm{D+IPW}}(\mathcal{C}) \coloneqq \hat{L}_{\mathrm{mask}}(\mathcal{C}) + \frac{1}{K}\sum_{j=1}^{K} \mathbf{1}\{\mathcal{C} \subseteq \mathcal{R}_j\} w(\mathcal{C}) \Big( \ell\Big( f\Big( x_{\mathcal{C}}^{(j)}, \mathcal{C} \Big), y^{(j)} \Big) - \hat{L}_{\mathrm{mask}}(\mathcal{C}) \Big).$$

The first term is the mask-aware candidate score. The second term adds an inverse-propensity correction only for selected neighbors where the entire candidate set is acquirable, i.e. $\mathcal{C} \subseteq \mathcal{R}_j$. Since $\mathcal{C}$ is the post-acquisition set, the propensity is estimated for the whole candidate set rather than only the newly added feature.

### 4.4.3 `OL` under Missingness

For `OL`, only a simple action-blocking mitigation is implemented, where the training action set is restricted according to the sampled missingness mask, as described in Definition 4.3. The variant `OL-with-mask` passes the current acquired-feature mask to the policy and classifier network, so the DQN can condition on which features are currently visible.

# 5

# Results

This chapter reports the main empirical findings of the thesis, organized in two sections. We first use **AFABench** to evaluate included methods and datasets in the standard AFA setting. We then focus on the missing data setting with the adapted methods from Section 4.4, evaluating performance under different training missingness rates and mechanisms.

## 5.1 AFABench in the Standard AFA Setting

As described in Section 4.1, the benchmark includes a diverse set of methods and datasets. To showcase the ability of **AFABench**, we illustrate an example evaluation of standard methods and datasets in the hard-budget and soft-budget settings. The training and evaluation protocols are followed as described in Section 4.1, using a shared external classifier and uniform acquisition costs to isolate the effect of the acquisition policy itself. Figure 5.1 and Figure 5.2 are generated end-to-end by **AFABench**. Each panel corresponds to a dataset, the $x$-axis is the hard budget or average accumulated cost per episode, and the $y$-axis is the predictive performance of the external classifier. The shaded bands show the mean plus or minus one standard deviation across five seeds.

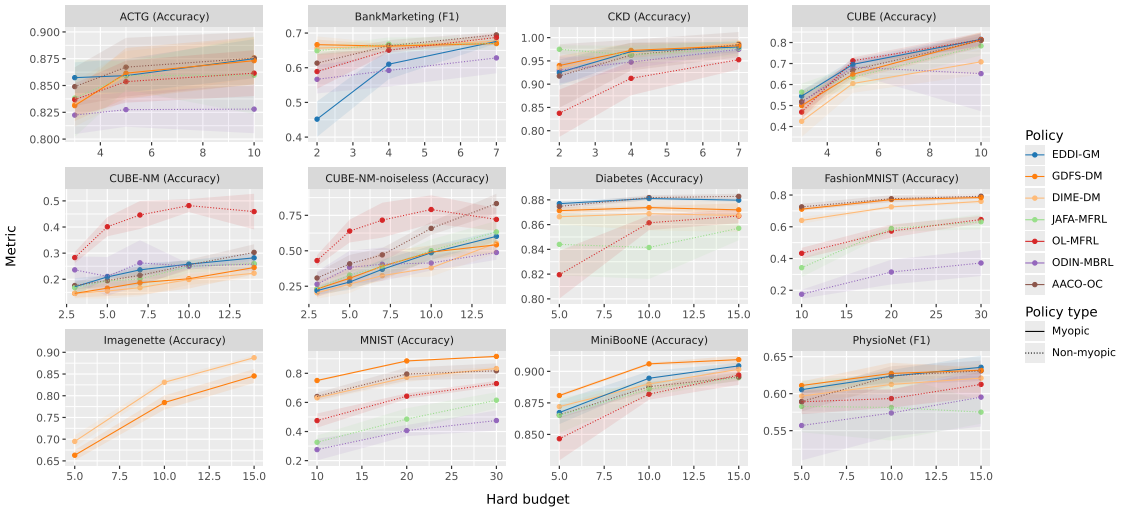



**Figure 5.1:** Hard-budget results with external classifier.

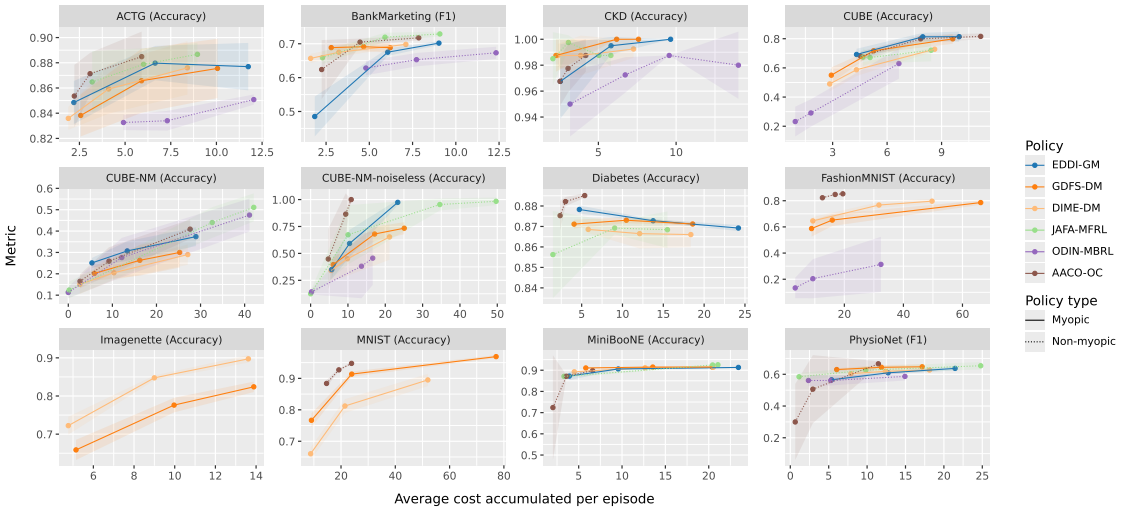



**Figure 5.2:** Soft-budget results with external classifier.

Figure 5.1 illustrates a common hard budget issue in standard AFA evaluation. In the standard AFA evaluation protocol, real-world datasets and previous synthetic datasets often fail to reveal meaningful differences between methods. Either the differences between methods are minuscule, even between myopic and non-myopic approaches, or the gaps are so large that they cannot be attributed to a specific cause, such as method formulation, dataset design, or the evaluation protocol itself. Furthermore, the common evaluation protocol of plotting performance as a function of acquisition budget, while intuitive and widely used, can be misleading when the acquisition order or trajectory differs between methods and is meaningful for the task.

Figure 5.2 reveals a similar picture in the soft-budget setting. As discussed in Section 3.4, tuning of the hyperparameter $\alpha$ that controls the cost-performance trade-off is both crucial and difficult, and the resulting performance can be highly sensitive to the choice of $\alpha$. As a result, some methods vary greatly in their average accumulated cost per episode while others have a significantly smaller spread. Ultimately, the large variance in average accumulated cost per episode across methods and datasets makes it difficult to uncover important insights.

`AFABench` collects the full acquisition trajectories of each method and calculates further statistical metrics to provide a more complete picture. Figure 5.3 and Figure 5.4 are the corresponding full acquisition trajectory and 2D error plots of Figure 5.1 and Figure 5.2, respectively. As in Figure 5.1 and Figure 5.2, each panel corresponds to a dataset, and the $y$-axis is the predictive performance of the external classifier. In Figure 5.3, the $x$-axis is the number of selections, and each curve corresponds to a method and shows the full acquisition trajectory across budgets. In Figure 5.4, the $x$-axis is the average accumulated cost per episode, each point corresponds to a soft-budget setting, and the 2D error bars show the mean plus or minus one standard deviation across five seeds.

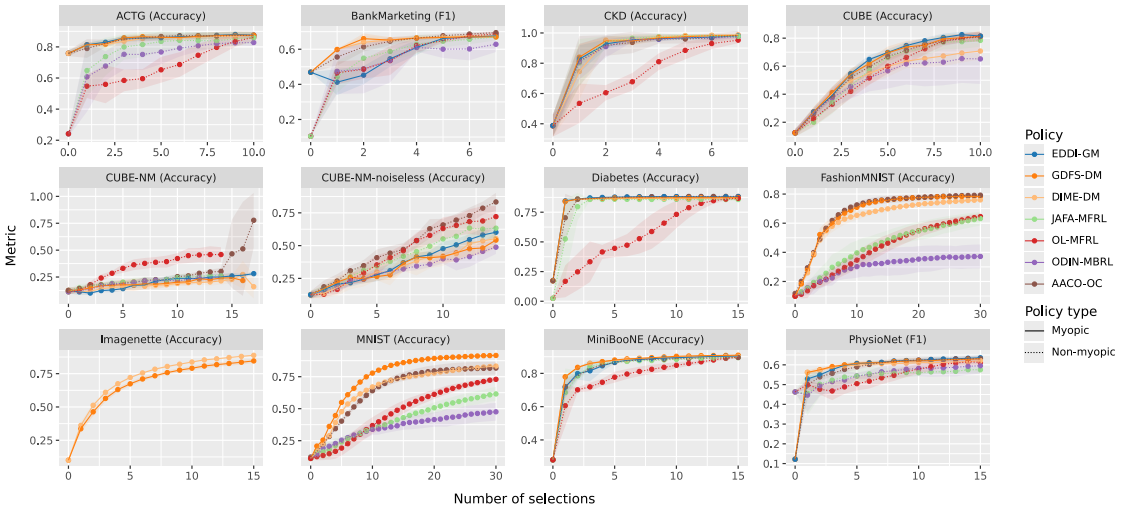



**Figure 5.3:** Acquisition trajectories in the hard-budget setting with external classifier.

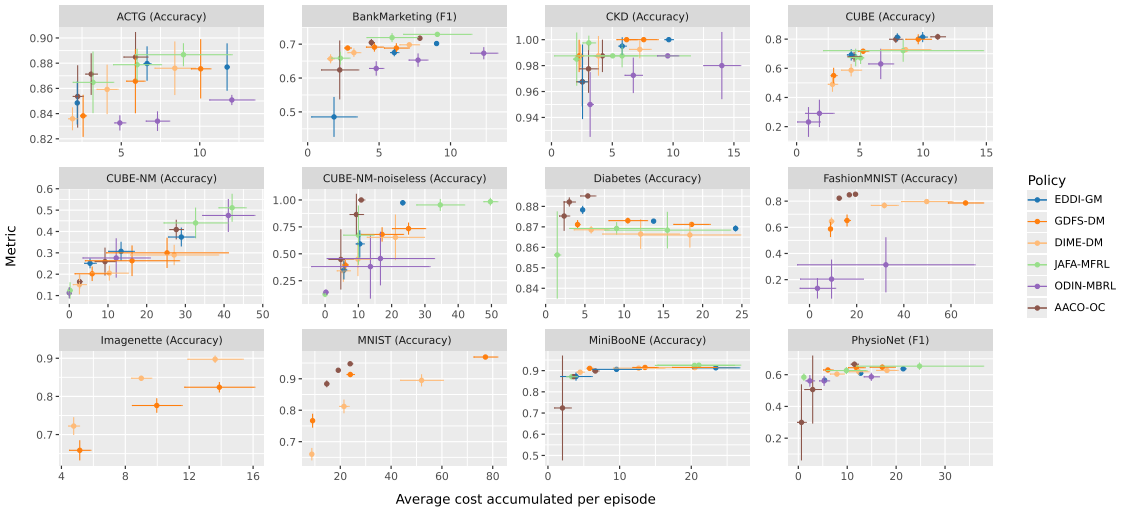



**Figure 5.4:** Soft-budget performance with external classifier, with 2D error bars showing the mean plus or minus one standard deviation across five seeds.

Figure 5.3 reveals that the same final hard-budget score can hide very different acquisition paths. On several datasets, the early part of the trajectory is more informative than the endpoint, because methods that end near the same metric can reach that level after different numbers of selections. Conversely, the `CUBE-NM` variants expose qualitatively different acquisition behavior: on the noisy variant `OL-MFRL` separates from the other methods, while on `CUBE-NM-noiseless AACO` separates most clearly in the last few steps. The trajectory view can separate early acquisition efficiency from final-budget performance, which the standard AFA evaluation protocol cannot.

Figure 5.4 reveals that some methods have a much wider spread in their cost-performance trade-off, while others are more consistent. Some methods occupy

similar metric ranges but spend very different average costs, and in several panels the horizontal uncertainty is as important as the vertical uncertainty. This matters because soft-budget evaluation is not only a question of predictive performance, but also a question of whether the stopping rule produces stable cost behavior under different values of $\alpha$.

### 5.1.1 Feature Selection Behavior on `CUBE-NM-noiseless`

Using the full acquisition trajectories, **`AFABench`** can compute an action heatmap that shows how frequently each feature is acquired at each acquisition step. These insights are not visible from the average performance curves, and they are crucial for understanding how methods differ in their acquisition behavior for optimality and robustness. Figure 5.5 visualizes the action heatmap for the hard-budget setting on **`CUBE-NM-noiseless`**.

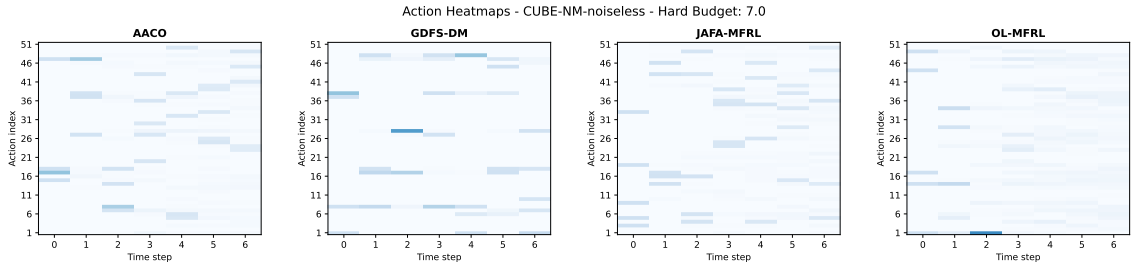



**Figure 5.5:** Action heatmap on **`CUBE-NM-noiseless`** in the hard-budget setting ($b = 7$). Action index 1 corresponds to the context feature.

From Figure 5.5, we can inspect whether a method learns the optimal policy, namely to acquire the context feature first and then acquire the corresponding **`CUBE`** features. In this evaluation example, the context feature is almost never selected. Only the **`OL`** model selects the context feature with non-negligible frequency. This insight alone is meaningful for understanding myopic vs. non-myopic behavior in AFA methods. Even in the easier noiseless setting, no AFA method learns the (simple) optimal policy, yet the achieved performance is still high and the methods are still ranked in a meaningful way. This shows that further work on current AFA methods is needed to solve **`CUBE-NM`** as intended, and that the current methods are not learning the optimal policy, even though they are still learning something useful.

## 5.2 AFA Training and Evaluation under Missing Data

We now present the main results for AFA under missing data. As described in Section 4.3 and Section 4.4, missingness is only applied during training. Evaluation is then performed in the ordinary AFA setting where all features are available to query. The experiment uses MCAR, MAR, and MNAR logistic missingness at rates 30%, 50%, and 70%. To make the relative effect of missingness visible for each method, Figure 5.6, Figure 5.7, and Figure 5.8 show the results as a percentage of

the corresponding full-data baseline. Here, 100% means that the method matches its full-data baseline, values below 100% indicate degradation, and values above 100% indicate improvement.

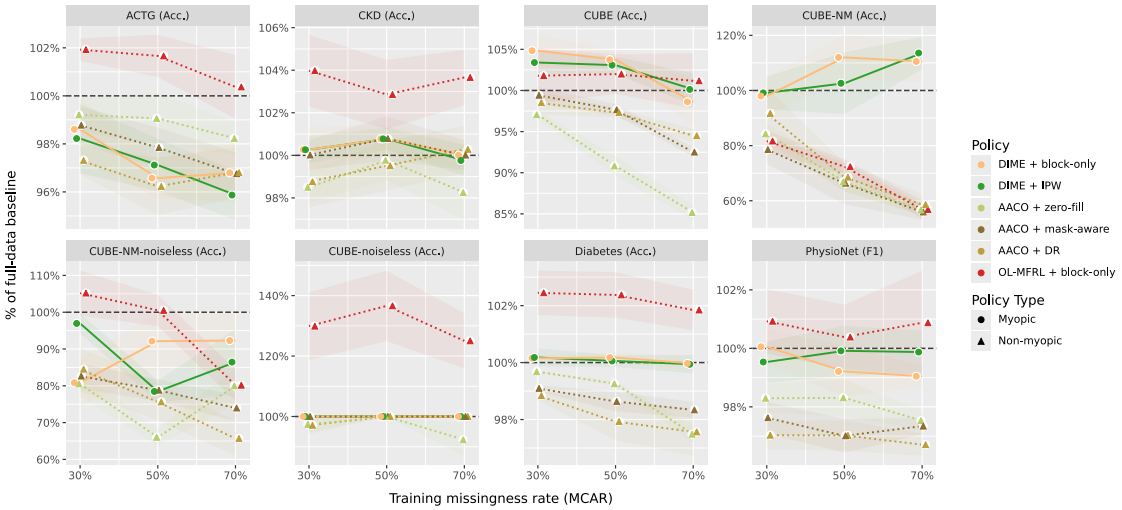



**Figure 5.6:** Hard-budget results under MCAR training missingness, normalized as a percentage of the corresponding full-data baseline.


**Figure 5.7:** Hard-budget results under MAR training missingness, normalized as a percentage of the corresponding full-data baseline.

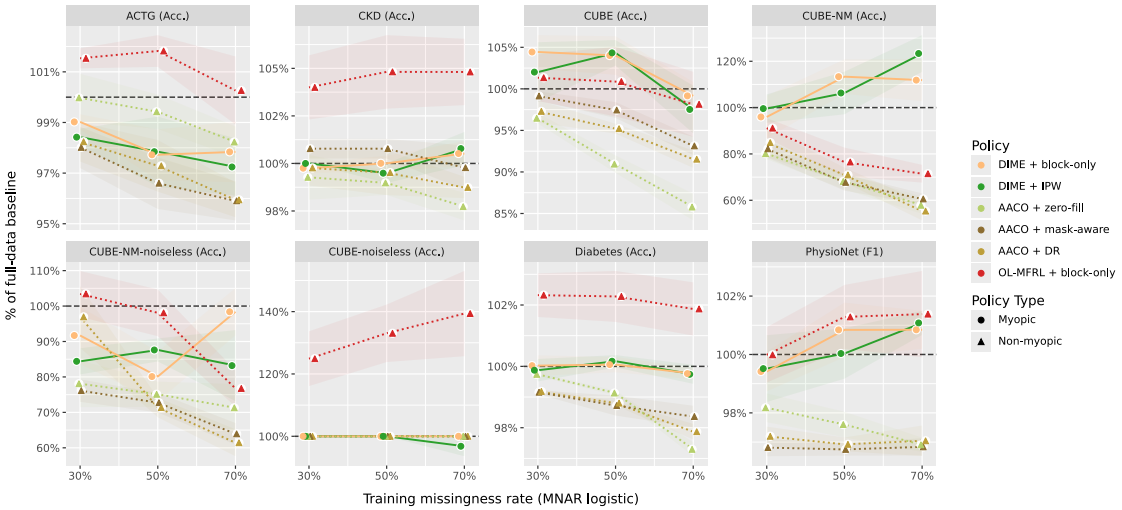



**Figure 5.8:** Hard-budget results under MNAR logistic training missingness, normalized as a percentage of the corresponding full-data baseline.

Furthermore, we summarize the same effect across dataset groups to make the main patterns more visible. Figure 5.9 reports the retained percentage of the full-data baseline. Each column corresponds to a missingness mechanism, each row corresponds to a dataset group, and the $x$-axis is the missingness rate. The vertical bars show 95% confidence intervals across dataset-level means. Finally, Figure 5.10 focuses on the 70% missingness setting and pools policy-level relative changes by method family. For this plot, each exact policy is first normalized against its own full-data baseline, and the bootstrap confidence intervals are then computed over the rows within the same myopic or non-myopic family.

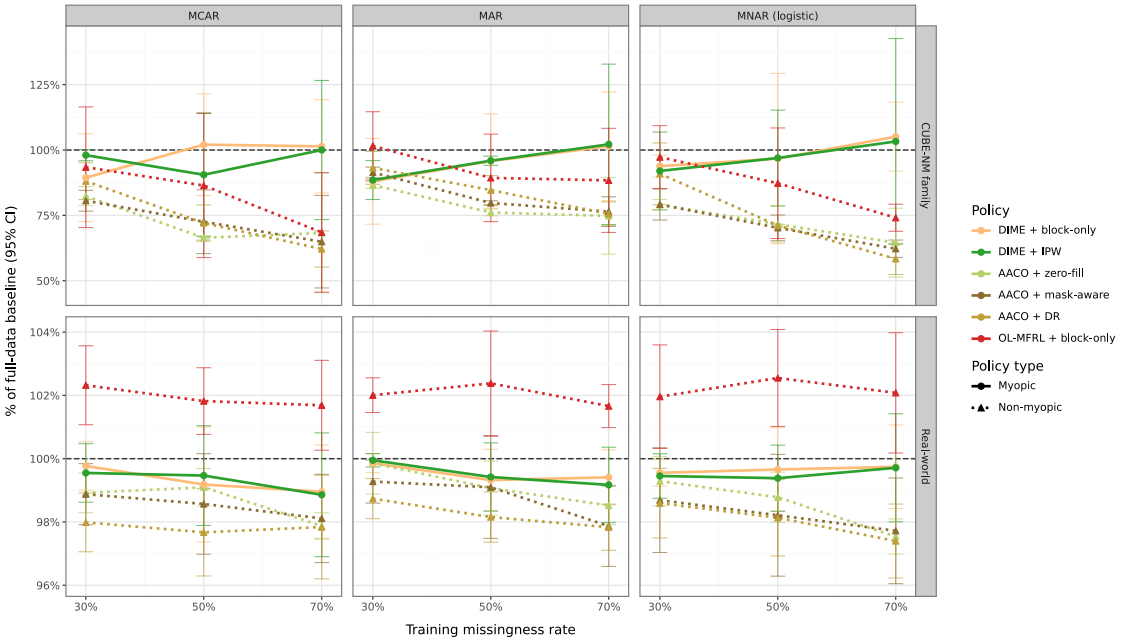



**Figure 5.9:** Hard-budget metric retained under training missingness as a percentage of the full-data baseline, with 95% confidence intervals across dataset-level means.

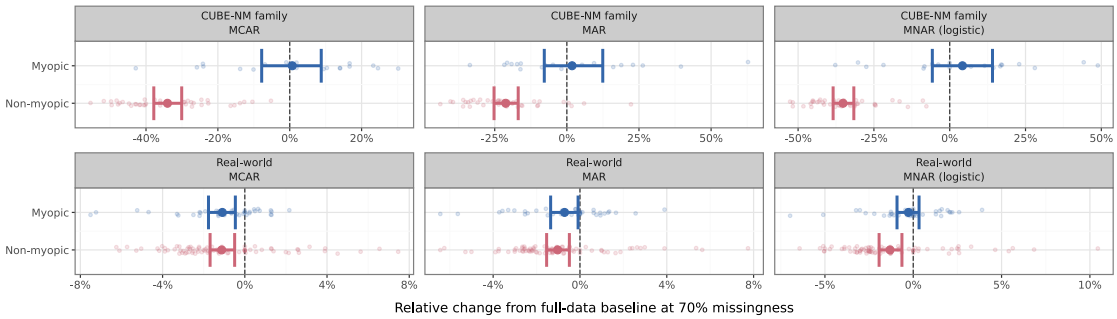



**Figure 5.10:** Relative hard-budget metric change from each policy's own full-data baseline at 70% training missingness, aggregated by method family with bootstrap 95% confidence intervals. Faint points show the policy–dataset–seed rows used in the bootstrap.

We now discuss the main insights from these results, organized by theme.

### 5.2.1 Non-Myopic Methods Collapse on `CUBE-NM`

In Figure 5.9 and Figure 5.10, the clearest pattern appears on the `CUBE-NM` family. The non-myopic variants move strongly below their full-data baseline as the missingness rate increases. This is expected from the construction of `CUBE-NM`. A useful policy must acquire the context feature and then choose the `CUBE` features based on that context. Because of missing data, crucial follow-up actions that make the current action valuable may be unavailable during training. This supports Proposition 4.3. The result is not merely that non-myopic methods perform

worse under missingness. The CUBE-NM family exposes a control-level failure where training missingness removes future actions that make certain current actions valuable. Conversely, the myopic variants show improved or stable performance under missingness on the CUBE-NM family. By construction of CUBE-NM, myopic methods never select the context feature, and therefore only linearly search for the best CUBE features. Thus, missing data can help by hiding context or noisy features that distract from the best CUBE features.

### 5.2.2 Myopic Methods Are (More) Insensitive to Missing Data

In Figure 5.9 and Figure 5.10, the DIME variants stay comparatively close to their full-data baseline. This does not mean that DIME solves CUBE-NM. As shown in Section 5.1.1, CUBE-NM is constructed so that the context feature is not immediately predictive of the label. A myopic method therefore has little incentive to acquire it. This agrees with Proposition 4.2. One-step methods are less affected when the local action ranking is unchanged. Multi-step methods can change behavior when later actions are unavailable during training.

### 5.2.3 Real-World Datasets Are Close to Full-Data Baselines

In Figure 5.9 and Figure 5.10, the real-world datasets stay close to their full-data baselines. This suggests that these datasets contain more redundant acquisition paths, or that a small number of strong features dominate prediction. In either case, removing some actions during training does not necessarily remove the path needed by a good evaluation policy. This limits the scope of the CUBE-NM result. Training missingness can be severe for structured multi-step tasks, but it is not automatically severe for every AFA dataset.

### 5.2.4 Mitigations Do Not Close the CUBE-NM Gap

In Figure 5.9, the AACO variants remain below the full-data AACO baseline on the CUBE-NM family. Zero-fill changes the input representation. Mask-aware search changes neighbor matching and candidate scoring. DR additionally applies inverse-propensity weighting. These changes affect the training signal, but they do not change which follow-up actions were available during training. This supports the two-part view from Section 4.3.5. Correcting the training signal is not enough if missingness also changes the control problem. Even a better weighted objective cannot restore acquisition routes that were absent during training.

### 5.2.5 Mechanism Effects

The mechanism comparison in Figure 5.9 is descriptive. In this benchmark configuration, MAR is often less damaging than MCAR and MNAR logistic on the CUBE-NM family. This likely reflects the implemented missingness process. MAR keeps a set of pivot features available and applies missingness to the remaining

features. MCAR and MNAR logistic remove actions more broadly at the same nominal rate.

# 6

# Discussion and Conclusion

This chapter summarizes the thesis contributions, connects the empirical findings to the formal development in Section 4.3, and outlines what we see as the most promising directions for future work.

## 6.1 Discussion

This thesis pursued two complementary goals. First, we developed the `AFABench` evaluation framework to make AFA methods directly comparable under a shared episode interface, and shipped it with a novel synthetic dataset `CUBE-NM` that exposes non-myopic behavior more cleanly than earlier synthetic tasks. Second, we studied AFA under training missingness, developing a POMDP extension that distinguishes posterior distortion from control distortion.

### 6.1.1 `AFABench`: Expanding the Evaluation Frontier

As discussed in Section 5.1, the standard AFA evaluation protocol can make certain insights hard to capture. The `AFABench` framework is designed to fix the episode interface and the evaluation protocol before any method-specific choice is made, so that all methods can be compared on a shared footing. Furthermore, the separation of hard-budget vs. soft-budget and internal vs. external evaluation allows us to isolate the effects of different design choices and to understand how they interact with the evaluation protocol itself. Finally, because `AFABench` collects all intermediate information, it allows us to go beyond final accuracy and analyze method behavior through full acquisition trajectories, additional statistical metrics, and action-level diagnostics. While this is not a new idea in AFA evaluation, the `AFABench` framework makes it more systematic and more accessible to future work.

### 6.1.2 AFA under Missing Data

The missing data experiments produced several striking patterns. Non-myopic methods collapse on `CUBE-NM` when trained with missingness, while their baselines and the myopic `DIME` curves are essentially unchanged. At 70% MCAR on `CUBE-NM`,

all three AACO variants end up significantly below their baseline, and none of the implemented mitigations meaningfully narrows that gap.

The formal development in Section 4.3 explains this outcome. Proposition 4.2 implies that myopic methods should agree with their baseline counterparts on feasible actions whenever the local posterior is unchanged, which is exactly what we see for DIME. Proposition 4.3, on the other hand, predicts that multi-step value-based methods can still degrade when previously useful follow-up acquisitions disappear during training, even if each single-step value is unchanged. The CUBE-NM collapse is the cleanest empirical realization of that proposition we could construct, because CUBE-NM was designed so that the optimal plan is a two-step one. The policy should acquire the context feature and then route to the informative block. The context-first route therefore looks less valuable during training, and the policy learned on the restricted problem transfers poorly to the unrestricted evaluation problem.

The three AACO mitigations deserve a more careful inspection, as they differ in how they use the training signal. zero-fill does nothing beyond the masked representation, mask-aware changes neighbor search and candidate scoring so that blocked entries never enter the loss, and DR adds an inverse-propensity correction on candidates that are fully observed in the training row. All three are recognizably sensible corrections when viewed as remedies for posterior distortion. None of them addresses the action-space restriction that Proposition 4.3 identifies as the dominant failure mode on CUBE-NM. Their empirical indistinguishability on that dataset is therefore not a sign that the individual corrections are wrong. It is a sign that the corrections target the wrong part of the problem in this regime.

### 6.1.3 When Missing Data Does Not Hurt

The robust behavior on ACTG, CKD, Diabetes, and PhysioNet is equally informative. These datasets either offer multiple redundant acquisition routes to the label or are dominated by a small number of strong features. In both cases, the evaluation-optimal policy has many equivalent continuations after any feasible first action, and the equality condition in Proposition 4.3 is close to being met on most training episodes. Non-myopic methods therefore remain close to their baselines even at 70% missingness, because the follow-up actions that the multi-step value depends on are usually available anyway.

This is a useful insight: the failure we identified on CUBE-NM is not a generic property of AFA training under missing data, but a specific consequence of tight multi-step structure interacting with a restricted training action set. Datasets where the label is recoverable through many different acquisition paths absorb moderate missingness without visible harm.

### 6.1.4 Societal, Ethical, and Sustainability Considerations

The main societal motivation for AFA is to support decisions when information is costly, delayed, invasive, or otherwise limited. This is especially visible in domains such as medical diagnosis, where each additional test can carry monetary cost, time cost, patient burden, privacy exposure, or clinical risk. AFA methods can therefore be useful when they reduce unnecessary measurements while preserving predictive performance. At the same time, this benefit depends on how the method is deployed. An acquisition policy that performs well on an average benchmark can still acquire the wrong information for underrepresented groups or for cases where the training data has systematic missingness. For this reason, AFA should not be interpreted only as an accuracy-cost optimization problem, but also as a problem about which information is requested from whom and under what assumptions.

The missing data results in this thesis sharpen that ethical concern. They show that availability assumptions can change the learned acquisition behavior even when evaluation is performed with all features available. In practical systems, this means that data collection patterns can influence which measurements a deployed policy values. If missingness is correlated with demographic, clinical, or operational factors, the resulting policy may reproduce those patterns. The formal distinction between posterior distortion and control distortion is therefore not only a technical distinction. It is also a way to identify whether a method fails because its current beliefs are biased, or because parts of the decision process were absent during training.

The benchmark itself also has sustainability implications. Training and evaluating many AFA methods across datasets, seeds, and missingness mechanisms is computationally expensive, and most large experiments in this thesis were run on shared high-performance computing resources. The purpose of `AFABench` is partly to make such computation more reusable: fixed configurations, stored outputs, and shared evaluation protocols reduce the need for each future comparison to rebuild the same infrastructure independently. Nevertheless, benchmark growth should be deliberate. Adding methods, datasets, or seeds is only justified when it answers a meaningful scientific question or materially improves reliability.

### 6.1.5 Limitations

We highlight three limitations that shape the scope of our claims. First, we study a single missingness regime, training with missingness and evaluating with the full action space. This is deliberately the setting where the training-versus-evaluation gap is easiest to analyze, but it is not the only regime of practical interest. Second, our mitigations were all applied at the level of the training signal or the state representation. The analysis suggests that the stronger intervention would be at the level of the training action set itself, for instance by generating synthetic continuations. However, we did not evaluate that class of mitigations in this thesis.

Third, the mechanism comparison between MCAR, MAR, and MNAR logistic is tied to our specific implementation; a different implementation could produce different relative patterns.

## 6.2 Conclusion

We return to the two research questions from Section 1.1.

The first question asked what components and evaluation protocol are necessary to compare AFA methods fairly. Chapter 4 and Section 5.1 together answer this question with the `AFABench` framework. It fixes the episode interface before any method-specific choice is made, separates hard-budget from soft-budget evaluation, uses a shared external classifier for the main comparison, and ships with synthetic tasks and action-level diagnostics that expose policy behavior rather than only final accuracy.

The second question asked what theoretical and empirical insights can be found for AFA under missing data. Section 4.3 extends the AFA POMDP to a missingness-aware training process, identifies when blockedness is ignorable for the current belief state, and formalizes how training-time action-space restriction can distort multi-step action values. Section 5.2 shows that this distortion is not a theoretical curiosity. On `CUBE-NM`, non-myopic methods lose roughly half of their accuracy under high training missingness, while myopic methods and baselines are stable. Mitigations that target the training signal alone, namely `mask-aware` and `DR` for `AACO` and IPW weighting for `DIME`, do not close that gap.

Taken together, these results suggest that training AFA under missingness is not primarily a reweighting problem. It is a problem of which acquisition routes the policy ever sees during training, and the most promising future directions are those that intervene on the training distribution itself rather than on the loss function alone.

### 6.2.1 Future Work

The most immediate follow-up is to evaluate mitigations that enlarge the training action set rather than merely reweighting it. One possibility is to use the PVAE-based generative model from the existing AFA methods to synthesize plausible values for blocked features and re-expand the training trajectories accordingly, so that the policy observes a full-information route even on blocked instances. Finally, `AFABench` was designed as a living framework, and we hope that the community adopts and extends it with additional methods, datasets, and evaluation protocols, so that future work on AFA can be compared on a genuinely shared footing.

# Bibliography


[1] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World: From Edge to Core," technical report, 2018. [Online]. Available: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

[2] D. Blatner, *Spectrums: Our Mind-boggling Universe from Infinitesimal to Infinity*. Bloomsbury Publishing, 2012. [Online]. Available: https://books.google.se/books?id=dvVXgmeosLgC

[3] International Data Corporation, "Global DataSphere Forecast, 2024–2028." [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[4] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, p. 354–, Oct. 2017, [Online]. Available: http://dx.doi.org/10.1038/nature24270

[5] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning.," *Nat.*, vol. 620, no. 7976, pp. 982–987, 2023, [Online]. Available: http://dblp.uni-trier.de/db/journals/nature/nature620.html#KaufmannBL0K023

[6] P. D. Turney, "Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm," *J. Artif. Int. Res.*, vol. 2, no. 1, pp. 369–409, Apr. 1995, [Online]. Available: https://dl.acm.org/doi/abs/10.5555/1622826.1622838

[7] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003, [Online]. Available: https://www.jmlr.org/papers/v3/guyon03a.html

[8] J. Janisch, T. Pevný, and V. Lisý, "Classification with costly features using deep reinforcement learning," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium*

*on Educational Advances in Artificial Intelligence*, in AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013959.

[9] R. Little and D. Rubin, *Statistical analysis with missing data*. in Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley, 2002. [Online]. Available: http://books.google.com/books?id=aYPwAAAAMAAJ

[10] D. B. Rubin, "Inference and Missing Data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976, doi: 10.2307/2335739.

[11] M. Le Morvan, J. Josse, T. Moreau, E. Scornet, and G. Varoquaux, "NeuMiss networks: differentiable programming for supervised learning with missing values," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, in NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. [Online]. Available: https://papers.nips.cc/paper/2020/file/42ae1544956fbe6e09242e6cd752444c-Paper.pdf

[12] L. Aronsson, A. Rahbar, and M. H. Chehreghani, "A Survey on Active Feature Acquisition Strategies." [Online]. Available: https://arxiv.org/abs/2502.11067

[13] C. Ma, S. Tschiatschek, K. Palla, J. M. Hernandez-Lobato, S. Nowozin, and C. Zhang, "EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., in Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 4234–4243. [Online]. Available: https://proceedings.mlr.press/v97/ma19c.html

[14] I. Covert, W. Qiu, M. Lu, N. Kim, N. White, and S.-I. Lee, "Learning to maximize mutual information for dynamic feature selection," in *Proceedings of the 40th International Conference on Machine Learning*, in ICML'23. Honolulu, Hawaii, USA: JMLR.org, 2023. [Online]. Available: https://openreview.net/pdf?id=dOaCuOsdmb

[15] S. Gadgil, I. C. Covert, and S.-I. Lee, "Estimating Conditional Mutual Information for Dynamic Feature Selection," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=Oju2Qu9jvn

[16] H. Shim, S. J. Hwang, and E. Yang, "Joint Active Feature Acquisition and Classification with Variable-Size Set Encoding," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/e5841df2166dd424a57127423d276bbe-Paper.pdf

[17] M. Kachuee, O. Goldstein, K. Kärkkäinen, and M. Sarrafzadeh, "Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eOHo09KX

[18] S. Zannone, J. M. Hernandez Lobato, C. Zhang, and K. Palla, "ODIN: Optimal Discovery of High-value INformation Using Model-based Deep Reinforcement Learning," in *Real-world Sequential Decision Making Workshop, ICML*, June 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/odin-optimal-discovery-of-high-value-information-using-model-based-deep-reinforcement-learning/

[19] M. Valancius, M. Lennon, and J. Oliva, "Acquisition Conditioned Oracle for Nongreedy Active Feature Acquisition," in *Proceedings of the 41st International Conference on Machine Learning*, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., in Proceedings of Machine Learning Research, vol. 235. PMLR, 2024, pp. 48957–48975. [Online]. Available: https://proceedings.mlr.press/v235/valancius24a.html

[20] H. von Kleist, A. Zamanian, I. Shpitser, and C. Ghanem, "Evaluation of Active Feature Acquisition Methods for Static Feature Settings," *arXiv preprint arXiv:2312.03619*, 2023, [Online]. Available: https://arxiv.org/pdf/2312.03619

[21] J. Wendland *et al.*, "Missingness-MDPs: Bridging the Theory of Missing Data and POMDPs." [Online]. Available: https://openreview.net/forum?id=8jYuRCHYxv

[22] P. de B. Harrington, "Machine Learning in Action," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:261329242

[23] C. Bishop, *Pattern recognition and machine learning*, vol. 4. Springer New York, 2006. [Online]. Available: http://scholar.google.com/scholar.bib?q=info:jYxggZ6Ag1YJ:scholar.google.com/&output=citation&hl=en&as_sdt=0%20,5&as_vis=1&ct=citation&cd=0

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[25] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. [Online]. Available: https://openreview.net/forum?id=33X9fd2-9FyZd

[26] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., in Proceedings of Machine Learning Research, vol. 32. Beijing, China: PMLR, 2014, pp. 1278–1286. [Online]. Available: https://proceedings.mlr.press/v32/rezende14.html

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[28] K. Åström, "Optimal control of Markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965, doi: https://doi.org/10.1016/0022-247X(65)90154-X.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Qi_PointNet_Deep_Learning_CVPR_2017_paper.pdf

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5105–5114. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf

[32] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based Reinforcement Learning: A Survey," *Found. Trends Mach. Learn.*, vol. 16, no. 1, pp. 1–118, Jan. 2023, doi: 10.1561/2200000086.

[33] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022. [Online]. Available: https://mitpress.mit.edu/9780262047012/algorithms-for-decision-making/

[34] R. D. Smallwood and E. J. Sondik, "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973, doi: 10.1287/opre.21.5.1071.

[35] G. Dulac-Arnold, L. Denoyer, and P. Gallinari, "Text Classification: A Sequential Reading Approach," in *Advances in Information Retrieval*, P. Clough, C. Foley, C. Gurrin, G. J. F. Jones, W. Kraaij, H. Lee, and V.

Murdock, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 411–423. [Online]. Available: https://doi.org/10.1007/978-3-642-20161-5_41

[36] V. Schütz, H. Wu, R. Rezvan, L. Aronsson, and M. Haghir Chehreghani, "AFABench: A Generic Framework for Benchmarking Active Feature Acquisition," in *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. [Online]. Available: https://doi.org/10.1145/3770855.3817493

[37] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms." [Online]. Available: http://arxiv.org/abs/1708.07747

[38] J. Howard, "Imagenette." [Online]. Available: https://github.com/fastai/imagenette

[39] D. G. Horvitz and D. J. Thompson, "A Generalization of Sampling Without Replacement from a Finite Universe," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 663–685, 1952, doi: 10.1080/01621459.1952.10483446.

# A

# Appendix

## A.1 Proofs for Missingness Results

**Proof of Proposition 4.1.**

*Proof*: In Definition 4.2, the latent variables $(x, y, m)$ remain fixed for a given data-instance realization. The only evolving component of the latent state is the acquisition set $\mathcal{S}$ together with the revealed coordinates $x_{\mathcal{S}}$. Let $h_t$ denote any history up to time $t$ and let,

$$o_t = O^{\text{train}}(s_t) = \Big(x_{\mathcal{S}_t}, \mathcal{S}_t, m\Big).$$

Because the observation includes the realized mask, the current admissible action set $\mathcal{A}_b^{\text{train}}(\mathcal{S}_t, m)$ is a deterministic function of $o_t$. Moreover, after any admissible non-stop action $a \neq \texttt{STOP}$, the successor latent state is determined uniquely by adding $a$ to $\mathcal{S}_t$ and revealing $x_a$. Therefore, for every measurable set $B \subseteq \mathcal{O}^{\text{train}}$, every admissible action $a \in \mathcal{A}_b^{\text{train}}(\mathcal{S}_t, m)$, and every history $h_t$ compatible with $o_t$, the transition probabilities satisfy,

$$P(o_{t+1} \in B, r_t \mid h_t, a_t = a) = P(o_{t+1} \in B, r_t \mid o_t, a_t = a).$$

Thus, the controlled process is Markov in the observed state $o_t$, which yields an equivalent fully observable MDP on $\mathcal{O}^{\text{train}}$. Consequently, an optimal policy can be chosen Markov in the blocked observation state, and a model-free method that trains directly in the blocked environment targets the optimal policy of $\mathcal{P}_b^{\text{train}}$ under standard convergence assumptions. ■

**Proof of Lemma 4.1.**

*Proof*: The assumption means precisely that,

$$p(m_i = 1 \mid y, x_{\mathcal{U}}, x_{\mathcal{S}}, \mathcal{S}) = p(m_i = 1 \mid x_{\mathcal{S}}, \mathcal{S}),$$

whenever the conditional distribution given $m_i = 1$ is well-defined.

Then Bayes' rule gives,

$$p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S}, m_i = 1) = \frac{p(m_i = 1 \mid y, x_{\mathcal{U}}, x_{\mathcal{S}}, \mathcal{S}) p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S})}{p(m_i = 1 \mid x_{\mathcal{S}}, \mathcal{S})} = p(y, x_{\mathcal{U}} \mid x_{\mathcal{S}}, \mathcal{S}).$$

■

**Proof of Proposition 4.2.**

*Proof*: Let $b^m$ and $b$ denote the current beliefs in the training and evaluation problems corresponding to $(x_{\mathcal{S}}, \mathcal{S}, m)$ and $(x_{\mathcal{S}}, \mathcal{S})$, respectively. For any action $a \in \mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \setminus \{\texttt{STOP}\}$, the Bellman recursion gives,

$$Q_1^{\text{train}}(b^m, a) = -\alpha c_a + \mathbb{E}_{x_a \mid x_{\mathcal{S}}, \mathcal{S}, m}[V_0^{\text{train}}(b')]$$

where $b'$ is the updated belief after observing $x_a$. Using the stopping reward from Section 3.4.1 and iterated expectation,

$$Q_1^{\text{train}}(b^m, a) = -\alpha c_a - \mathbb{E}_{\mathcal{P}^{\text{train}}}\Big[\ell\Big(f\Big(x_{\mathcal{S}\cup\{a\}}, \mathcal{S}\cup\{a\}\Big), y\Big) \;\Big|\; x_{\mathcal{S}}, \mathcal{S}, m\Big].$$

The corresponding quantity in the evaluation problem is

$$Q_1^{\text{eval}}(b, a) = -\alpha c_a - \mathbb{E}_{\mathcal{P}^{\text{eval}}}\Big[\ell\Big(f\Big(x_{\mathcal{S}\cup\{a\}}, \mathcal{S}\cup\{a\}\Big), y\Big) \;\Big|\; x_{\mathcal{S}}, \mathcal{S}\Big].$$

By assumption, the joint distribution of $(y, x_a)$ at the current state agrees in the training and evaluation problems. Therefore the two expectations above are equal, which gives,

$$Q_1^{\text{train}}(b^m, a) = Q_1^{\text{eval}}(b, a).$$

This holds for every currently feasible action $a$, so maximizing the one-step value yields the same ranking on the common feasible action set. ■

**Proof of Proposition 4.3.**

*Proof*: For every state and mask realization, the training action set is a subset of the evaluation action set,

$$\mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \subseteq \mathcal{A}_b^{\text{eval}}(\mathcal{S}).$$

We prove the value inequality by induction on $k$. For $k = 0$, both problems coincide with immediate stopping, so $V_0^{m(b)} = V_0^{\text{eval}}(b) = V_0(b)$.

Now assume $V_{k-1}^{m(b)} \le V_{k-1}^{\text{eval}}(b)$ for every feasible belief state $b$. Fix any common feasible belief state $b$ and any acquisition action $a \in \mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \setminus \{\texttt{STOP}\}$. Using the Bellman recursion from Section 3.4.2 and the induction hypothesis,

$$Q_k^{m(b,a)} = -\alpha c_a + \mathbb{E}_{x_a \mid b}\Big[V_{k-1}^{m(b')}\Big] \le -\alpha c_a + \mathbb{E}_{x_a \mid b}\big[V_{k-1}^{\text{eval}}(b')\big] = Q_k^{\text{eval}}(b, a),$$

where $b'$ is the updated belief after observing $x_a$. For the stopping action, we have,

$$Q_k^{m(b,\,\texttt{STOP})} = Q_k^{\text{eval}}(b, \texttt{STOP}) = V_0(b).$$

Taking the maximum over the smaller training action set and using,

$$\mathcal{A}_b^{\text{train}}(\mathcal{S}, m) \subseteq \mathcal{A}_b^{\text{eval}}(\mathcal{S})$$

therefore yields,

$$V_k^{m(b)} \leq V_k^{\text{eval}}(b).$$

This proves the claim for all $k$.

Equality holds whenever, at every belief reachable within the next $k$ selections, there exists an evaluation-optimal maximizing action that also belongs to the restricted action set and the induction inequalities are tight along those maximizing continuations. ■

## A.2 Proof of Cube-NM Formal Properties

**Proof of Theorem 4.1.**

*Proof*: Let $n := n_c$, in the noiseless setting ($\sigma = 0$) each pair of class label and context determines a unique feature vector. For noiseless `CUBE`, let $z^y \in \{0, 0.5, 1\}^{10}$ denote the 10-dimensional prototype for class $y \in \{1, ..., 8\}$ and let $z_i^y$ be its $i$-th coordinate. For `CUBE-NM` with $n$ contexts, let $f_1$ denote the grouped context action and let $g_{b,i}$ denote the $i$-th coordinate in block $b \in \{1, ..., n\}$. If the active block is $j$ and the label is $y$, then $x_{g_{j,i}} = z_i^y$ for all $i \in \{1, ..., 10\}$ and $x_{g_{b,i}} = 0.5$ for all $b \neq j$. Because all acquisition costs are one, $c(\pi[x])$ is exactly the number of queried actions.

Consider the noiseless 10-feature core with 8 equally likely classes. Query feature 7 first, if $z_7^y = 1$, then $y = 5$ and we stop. If $z_7^y = 0$, then $y \in \{6, 7\}$ and feature 6 resolves the ambiguity. If $z_7^y = 0.5$, then $y \in \{1, 2, 3, 4, 8\}$, querying feature 2 isolates classes 1 and 2, while one additional query resolves the remaining three-class branch. Hence the expected number of queries is,

$$1 + \left(\frac{2}{8}\right) \cdot 1 + \left(\frac{5}{8}\right) \cdot \left(1 + \left(\frac{3}{5}\right) \cdot 1\right) = 2.25.$$

The best case is therefore 1 query and the worst case is 3 queries. To show optimality, let $V(\mathcal{B}, \mathcal{S})$ denote the minimum expected number of additional queries needed when the consistent label set is $\mathcal{B} \subseteq \{1, ..., 8\}$ and the queried feature set is $\mathcal{S} \subseteq \{1, ..., 10\}$. The boundary condition is $V(\{y\}, \mathcal{S}) = 0$. For any $i \notin \mathcal{S}$,

$$V(\mathcal{B},\mathcal{S}) = 1 + \min_{i\in\{1,\dots,10\}\setminus\mathcal{S}} \sum_{v\in\{0,0.5,1\}} P(v \mid \mathcal{B},\mathcal{S},i)V(\mathcal{B}_{i=v},\mathcal{S}\cup\{i\}),$$

where $\mathcal{B}_{i=v} := \{y \in \mathcal{B} : z_i^y = v\}$.

Evaluating this recursion on the noiseless prototypes yields $V(\{1,...,8\},\emptyset) = 2.25$, so the policy above is optimal. The same recursion implies best-case cost 1 and worst-case cost 3.

If $n = 1$, the problem reduces to noiseless `CUBE`, so the expected, best-case, and worst-case costs are 2.25, 1, and 3, respectively. Assume instead that $n \geq 2$. Querying the context action $f_1$ reveals the active block in one step. After that, the remaining task is exactly noiseless `CUBE` restricted to the active block, whose optimal expected cost is 2.25 by Step 1. Therefore the context-first policy has expected cost $1 + 2.25$, best-case cost $1 + 1$, and worst-case cost $1 + 3$.

This policy is optimal for the noiseless instance family considered here. Before the active block is known, a within-block query can spend cost on an inactive block, which deterministically returns 0.5 and gives no label information. The context action identifies the active block at the minimum possible cost, after which Step 1 gives the optimal remaining cost. Hence,

$$\mathbb{E}_x\mathbb{E}_{\pi^\star}\left[c\left(\pi^{\star[x]}\right)\right] = \mathbf{1}\{n \geq 2\} + 2.25,$$

and,

$$\min_{x\in\mathcal{X}} c\left(\pi^{\star[x]}\right) = \mathbf{1}\{n \geq 2\} + 1,$$
$$\max_{x\in\mathcal{X}} c\left(\pi^{\star[x]}\right) = \mathbf{1}\{n \geq 2\} + 3.$$

We now analyze the myopic CMI policy with the uniform tie-breaking convention stated in Theorem 4.1. By construction, the context is independent of the label, so $I\left(y; x_{f_1}\right) = 0$. The myopic CMI policy therefore ignores $f_1$ whenever some block feature still has strictly positive one-step information gain. Initially, the CMI-maximizing coordinates are $i \in \{4,5,6,7\}$ in any block. The uniform tie-breaking convention chooses uniformly among the corresponding maximizing block-feature actions. After the policy queries $g_{b,i}$ and observes 0.5, the queried block becomes less likely to be active than any untouched block, because 0.5 is guaranteed in inactive blocks but occurs only for a strict subset of labels in the active block. Consequently, the policy continues by testing the same within-block index across previously untouched blocks before it explores a different coordinate in the same block. This induces a stage decomposition. In one stage, the policy fixes an index $i$ and tests features $g_{b,i}$ block by block until it either observes a non-0.5 value

in the active block or exhausts all $n$ blocks and concludes that the active block also has value 0.5 at index $i$.

Let,

$$q(p) := p \cdot \frac{n+1}{2} + (1-p) \cdot n$$

denote the expected number of block tests in a stage when the active block has a non-0.5 value with probability $p$. There are two symmetry classes for the first-stage coordinate chosen by the uniform tie-breaking rule. Exactly 3 of the 8 noiseless `CUBE` prototypes satisfy $z_4^y \neq 0.5$, so the first stage has $p = \frac{3}{8}$ and cost $q\left(\frac{3}{8}\right)$. If that first-stage observation is non-0.5, then two labels remain with probability $\frac{2}{8}$, which requires one more query. Otherwise the posterior shrinks to a five-label set, and the next maximizing index is 7, where again 3 of the 5 remaining labels have non-0.5 values. Thus,

$$G(n) = q\left(\frac{3}{8}\right) + \left(\frac{2}{8}\right) \cdot 1 + \left(\frac{5}{8}\right) \cdot H(n),$$

where,

$$H(n) = q\left(\frac{3}{5}\right) + \left(\frac{2}{5}\right) \cdot 1 + \left(\frac{2}{5}\right) \cdot q\left(\frac{1}{2}\right).$$

This simplifies to $G(n) = \frac{23n+15}{16}$. Again exactly 3 of the 8 classes have non-0.5 values, so the first stage contributes $q\left(\frac{3}{8}\right)$ and leaves a two-label ambiguity with probability $\frac{2}{8}$. If the first-stage value is 0.5, the posterior reduces to a five-label set and the next maximizing index is 2, where 2 of the 5 remaining labels have non-0.5 values. The resulting recursion is,

$$B(n) = q\left(\frac{3}{8}\right) + \left(\frac{2}{8}\right) \cdot 1 + \left(\frac{5}{8}\right) \cdot M(n),$$

where,

$$M(n) = q\left(\frac{2}{5}\right) + \left(\frac{3}{5}\right) \cdot C(n),$$

and,

$$C(n) = \left(\frac{1}{3}\right) \cdot \frac{n+1}{2} + \left(\frac{2}{3}\right) \cdot \left(n + q\left(\frac{1}{2}\right)\right) = \frac{4n+1}{3}.$$

Hence $B(n) = \frac{29n+11}{16}$. The initial CMI tie is split uniformly between the two coordinate symmetry classes, so,

$$\mathbb{E}_x \mathbb{E}_{\pi_{\mathrm{CMI}}}\left[c\left(\pi_{\mathrm{CMI}[x]}\right)\right] = \frac{G(n) + B(n)}{2} = 13\frac{2n+1}{16}.$$

For the best case, there exist labels for which the first informative query inside the active block already identifies the class, so $\min_{x \in \mathcal{X}} c\left(\pi_{\mathrm{CMI}[x]}\right) = 1$. For the worst case, there exist labels for which three successive stages each require all $n$ block tests, which gives $\max_{x \in \mathcal{X}} c\left(\pi_{\mathrm{CMI}[x]}\right) = 3n$. This concludes the proof. ■

## A.3 Detailed Description of Included Datasets

**Table A.1:** Summary of datasets used in the benchmark.

| Dataset | Type | Modality | Train | Val | Test | Features | Groups | Classes |
|---|---|---|---|---|---|---|---|---|
| CUBE | Synthetic | Tabular | 600 | 200 | 200 | 20 | 20 | 8 |
| CUBE-NUC | Synthetic | Tabular | 600 | 200 | 200 | 20 | 20 | 8 |
| CUBE-NM | Synthetic | Tabular | 600 | 200 | 200 | 55 | 51 | 8 |
| CUBE-NM-noise-less | Synthetic | Tabular | 600 | 200 | 200 | 55 | 51 | 8 |
| MNIST | Real-world | Image/tabular | 36,000 | 12,000 | 12,000 | 784 | 784 | 10 |
| FashionMNIST | Real-world | Image/tabular | 36,000 | 12,000 | 12,000 | 784 | 784 | 10 |
| Diabetes | Real-world | Tabular | 55,237 | 18,412 | 18,413 | 45 | 45 | 3 |
| PhysioNet | Real-world | Tabular | 7,200 | 2,400 | 2,400 | 41 | 41 | 2 |
| MiniBooNE | Real-world | Tabular | 78,038 | 26,012 | 26,014 | 50 | 50 | 2 |
| ACTG175 | Real-world | Tabular | 1,283 | 427 | 429 | 23 | 23 | 2 |
| CKD | Real-world | Tabular | 240 | 80 | 80 | 24 | 24 | 2 |
| BankMarketing | Real-world | Tabular | 27,126 | 9,042 | 9,043 | 16 | 16 | 2 |
| Imagenette | Real-world | Image | 5,681 | 3,788 | 3,925 | 150,528 | 196 | 10 |

Table A.1 summarizes the benchmark datasets. The main text introduces the role of each dataset in the benchmark; here we collect implementation details that are useful when interpreting the experiments and reproducing the evaluation setup.

### A.3.1 Synthetic Datasets

`CUBE` is a standard synthetic AFA benchmark with 20 real-valued features and 8 classes. For each class, three coordinates are informative and the remaining coordinates are Gaussian noise. We use it as the canonical dynamic-acquisition testbed and use `CUBE-NUC` as its non-uniform-cost counterpart.

`CUBE-NM` extends `CUBE` with a context variable that determines which feature block is informative, while the context itself is not immediately predictive under the myopic criterion in Eq. (4.1). This makes `CUBE-NM` useful for separating genuinely non-myopic policies from one-step heuristics. In the implementation, the context variable is encoded as a one-hot vector for easier learning, and the corresponding coordinates are grouped into a single acquisition action through a dedicated `AFAUnmasker`. The variant `CUBE-NM-noiseless` sets $\sigma = 0$ and is used for the formal analysis in Theorem 4.1; the proof is given in Section A.2.

### A.3.2 Image Datasets Treated as Tabular

`MNIST` is the standard handwritten-digit benchmark [24]. Following prior AFA work such as [15], we treat each pixel as a separate feature and therefore evaluate

it as a high-dimensional tabular problem rather than as an image-recognition task with spatial inductive bias.

`FashionMNIST` has the same image format as `MNIST`, but contains grayscale clothing items instead of digits. We apply the same tabularization so that acquisition policies act on individual pixels and can be compared directly with the other tabular benchmarks.

### A.3.3 Real-World Tabular Datasets

`Diabetes` is a three-class diabetes-diagnosis task derived from NHANES data and widely used in the AFA literature. We use the common preprocessed version used in earlier AFA work and keep the class split into normal, pre-diabetes, and diabetes.

`PhysioNet` is derived from the PhysioNet Challenge 2012 and targets binary ICU mortality prediction from electronic health records. The dataset is naturally incomplete and clinically motivated, making it an important bridge between controlled synthetic experiments and realistic medical AFA.

`MiniBooNE` is a tabular particle-identification dataset from the MiniBooNE experiment at Fermilab. The task is binary classification from detector-level features and is useful because it is large, fully observed, and substantially higher dimensional than the smaller clinical datasets.

`ACTG175`, `CKD`, and `BankMarketing` are standard UCI-style tabular classification datasets. They broaden the evaluation beyond the datasets that dominate the earlier AFA literature and help test how robust the methods are across domains, scale, and feature semantics.

### A.3.4 Patch-Based Image Acquisition

`Imagenette` is a 10-class subset of ImageNet used here for grouped image acquisition. Unlike `MNIST` and `FashionMNIST`, it is kept in image form and evaluated with a patch-based `AFAUnmasker`. This lets the benchmark cover acquisition settings in which one action reveals a structured region rather than a single scalar feature.

## A.4 Reproducibility Notes

The experiments in this thesis were implemented in the `AFABench` repository. The public repository is available at github.com/Linusaronsson/AFA-Benchmark.

The project uses Python 3.12 and manages dependencies with `uv`. The benchmark pipeline is orchestrated with `Snakemake` and stores configuration in YAML files under `extra/workflow/conf/`. The standard hard-budget and soft-budget benchmark runs use the orchestration workflow `extra/workflow/snakefiles/orchestration/pipeline.smk` together with the method, dataset, unmasker, budget, classifier, and pretraining configura-

tion files. The missing-data experiments use the cold-start missingness evaluation configuration files, including `train_missing_eval_cold_hard.yaml`, `train_missing_eval_cold_initializer_comparison.yaml`, and cluster-specific missingness configuration files for Alvis and Vera. Most large experiments were run on the Alvis and Vera high-performance computing systems, while smoke tests and selected plotting steps were run locally.

The repository README gives the basic local pipeline command. In compact form, a full local run has the structure,

```
WANDB_PROJECT=afabench uv run snakemake \
  -s extra/workflow/snakefiles/orchestration/pipeline.smk \
  all \
  --configfile \
    extra/workflow/conf/eval_hard_budgets.yaml \
    extra/workflow/conf/methods.yaml \
    extra/workflow/conf/method_sets.yaml \
    extra/workflow/conf/method_options.yaml \
    extra/workflow/conf/pretrain_mapping.yaml \
    extra/workflow/conf/soft_budget_params.yaml \
    extra/workflow/conf/unmaskers.yaml \
    extra/workflow/conf/classifier_names.yaml \
    extra/workflow/conf/datasets_main.yaml \
  --config \
    eval_dataset_split=test \
    "dataset_instance_indices=[0,1,2,3,4]" \
    smoke_test=false \
    use_wandb=true \
    device=cpu
```

The main thesis figures were generated from the benchmark outputs with plotting and analysis scripts under `scripts/plotting/` and `scripts/analysis/`. In particular, the missingness figures use the train-missing analysis and plotting scripts, while the `CUBE-NM` action diagnostics use the action-heatmap and `CUBE-NM` plotting utilities. The resulting figure files were copied into `deliverables/thesis/imgs/results/` before compiling the report.

Randomness enters through dataset splits, model initialization, method training, and missingness-mask generation. The main reported evaluations aggregate over five dataset-instance seeds unless otherwise stated.

## A.5 Author Contributions

This thesis was carried out jointly by Reza Rezvan and Han Wu. Both authors contributed to the thesis formulation, literature study, implementation work, experiment design, result interpretation, and writing of the final report. The `AFABench` framework was developed in close collaboration with the supervisors, and the thesis work builds on that shared research codebase.

- Reza Rezvan contributed mainly to the missing data extension of the benchmark, the missingness workflow configuration, cluster execution on Alvis and Vera, `AACO` missingness variants, aggregation and plotting scripts for the missingness experiments, and the formalization and writing around AFA under training missingness.
- Han Wu contributed mainly to the implementation and configuration of discriminative and greedy AFA methods, including `DIME` related training and missingness-aware variants, image and patch-based acquisition support, method training configuration, and the writing around benchmarked methods and empirical evaluation.
- Joint work included refining the research questions, aligning the benchmark interface with the thesis scope, selecting datasets and evaluation protocols, interpreting the main empirical patterns, revising the thesis text, and checking that the final figures and conclusions were consistent with the implemented experiments.