Generator Assisted Mixture of Experts For Feature Acquisition in Batch

Vedang Asgaonkar, Aditya Jain, Abir De

IIT Bombay

{vedang, adityajainjhs, abir}@cse.iitb.ac.in

Abstract

Given a set of observations, feature acquisition is about finding the subset of unobserved features which would enhance accuracy. Such problems have been explored in a sequential setting in prior work. Here, the model receives feedback from every new feature acquired and chooses to explore more features or to predict. However, sequential acquisition is not feasible in some settings where time is of the essence. We consider the problem of feature acquisition in batch, where the subset of features to be queried in batch is chosen based on the currently observed features, and then acquired as a batch, followed by prediction. We solve this problem using several technical innovations. First, we use a feature generator to draw a subset of the synthetic features for some examples, which reduces the cost of oracle queries. Second, to make the feature acquisition problem tractable for the large heterogeneous observed features, we partition the data into buckets, by borrowing tools from locality sensitive hashing and then train a mixture of experts model. Third, we design a tractable lower bound of the original objective. We use a greedy algorithm combined with model training to solve the underlying problem. Experiments with four datasets show that our approach outperforms these methods in terms of trade-off between accuracy and feature acquisition cost.

1 Introduction

Supervised learning algorithms often assume access to a complete set of features $\mathbf{x} \in \mathbb{R}^d$ to model the underlying classifier $\Pr(y \,|\, \mathbf{x})$. However, in applications like healthcare, information retrieval, *etc.*, a key goal is feature acquisition (**??**), where the learner may observe only a subset of features $\mathcal{O} \subset \{1, .., d\}$ and the goal is to query for a new subset U from the unobserved set of features: $U \subset \{1, ..., d\} \backslash \mathcal{O}$. For example, when a patient visits a doctor with a new health issue, the doctor can observe only few symptoms. If the symptoms are not informative enough to diagnose a disease with high confidence, the doctor may ask for additional medical tests.

1.1 Prior work and their limitations

Driven by these motivations, feature acquisition is widely studied in literature. Earlier works used tools from active learning techniques (**?????**), which optimize measures based on variance, uncertainty or information gain. To improve their performance, a recent line of work explicitly optimizes the prediction accuracy (**??????????**), predominantly using reinforcement learning (RL).

The above methods are tailored for *sequential* feature acquisition. In such scenarios, it is feasible to observe the value of a newly acquired feature immediately after its acquisition, allowing the use of its true value to inform the acquisition of additional features. However, certain situations involve a substantial delay between querying one feature and observing its value. In these cases, it may be more practical to batch-query a subset of features instead of acquiring them one by one in an online fashion. For instance, in healthcare, the analysis of pathological samples can introduce significant delays after collection. Thus, doctors may need to obtain results from multiple tests at once for rapid diagnosis. We provide a detailed survey of related work in Appendix C.

1.2 Our contributions

Responding to the above challenge, we propose GenEx, a novel feature acquisition method to acquire features in batch. Specifically, we make the following contributions.

Using feature generator to reduce oracle queries Feature generators are commonly used in feature acquisition tasks to guide feature selection policies (**??**). However, these generated features typically are not utilized for final label prediction. In our work, instead of querying all features from an oracle, we draw a feature subset from the generator and directly employ them for classification, reducing the number of oracle queries with only a marginal loss in accuracy.

Mixture of experts on heterogeneous feature space The observed features $\mathcal{O}$ can vary significantly across instances. This leads to a diverse set of acquired features and, consequently, a range of heterogeneous data domains. Generalizing across such heterogeneity using a single model is challenging. To address this, we partition the dataset into clusters or domains using a random hyperplane-based approximate nearest neighbor technique (**?**). We then build a mixture of experts model on these clusters, with each cluster representing instances likely to share a similar set of optimal features for acquisition. Each mixture component specializes in generalizing on a specific data subset.

Discrete continuous training framework The original feature acquisition problem is intractable due to the coupling of a large number of set variables. To tackle this, we design a surrogate loss guided by generator confidence and seek to minimize it alongside the feature subsets. This leads to a discrete-continuous optimization framework which is NP-hard. To tackle this problem, we reformulate it into a cardinality-constrained set function optimization task. Subsequently, we introduce novel set function properties, $(\underline{m}, \overline{m})$-partial monotonicity and $(\underline{\gamma}, \overline{\gamma})$-weak submodularity, extending recent notions of partial monotonicity (**?**) and approximate submodularity (**???**). These properties allow us to design a greedy algorithm GENEX, to compute near-optimal feature subsets, with new approximation guarantee.

We experiment with four real datasets and show that, GENEX outperforms several baselines. Moreover, our extensive ablation study shows that our use of a generative model reduces the cost of querying at a minimal accuracy drop.

2 Problem formulation

Notations and problem setup We use $\mathbf{x} \in \mathbb{R}^n$ to represent a feature vector and $y \in \mathcal{Y}$ for the associated label. $\mathcal{I} = [n]$ denotes the set of feature indices. $\mathcal{O} \subset \mathcal{I}$ represents the indices of observed features, while $U \subseteq \mathcal{I}\backslash\mathcal{O}$ represents indices of subset of features to be queried. Given a feature vector $\mathbf{x}$, $\mathbf{x}[\mathcal{O}]$ consists of features indexed by $\mathcal{O}$. We denote a singleton feature as $x[s] \in \mathbb{R}$ for index s.

Our classifier is denoted as $h \in \mathcal{H}$, where $h(\mathbf{x})[y] = \Pr(y \,|\, \mathbf{x})$ and $\mathcal{H}$ is the hypothesis class. We also employ a generative model for features, denoted as $p(\mathbf{x}[V] \,|\, \mathbf{x}[\mathcal{O}])$, which generates new features $\mathbf{x}[V]$ conditioned on observed features $\mathbf{x}[\mathcal{O}]$. For clear disambiguation from oracle features $\mathbf{x}[\bullet]$, we use $\mathbf{x}'[\bullet]$ for features drawn from the generator p. We utilize it to draw a subset of unseen features $V \subset U$, rather than querying the oracle. In our work, we use the cross-entropy loss $\ell(h(\mathbf{x}), y)$.

High level objective Given an instance $\mathbf{x}$, we initially observe only a subset of the features $\mathbf{x}[\mathcal{O}]$ indexed by $\mathcal{O}$ which varies across the instances. In general, this small subset is not sufficient for accurate prediction. Hence, we would seek to query new features $\mathbf{x}[U]$ subject to a maximum number of oracle queries. Thus, our key goal is to use $\mathbf{x}[\mathcal{O}]$ to determine the optimal choice of U among all such possible subsets, such that $\mathbf{x}[\mathcal{O} \cup U]$ results in high accuracy. Note that, here, we aim to acquire the oracle features in *batch* and not in sequence, *i.e.*, we may not observe a part of the unobserved features, before we query the rest.

Now, suppose that by some means, we have determined such subset U, so that $\mathbf{x}[U]$ obtained via querying from the oracle would result in high accuracy. Still, it may not be always necessary to query the value of every feature $x(u)$ for all $u \in U$ from the oracle. For some subset $V \subset U$, the predicted features $\mathbf{x}'[V]$, which are drawn the feature generator p can lead to similar accuracy as the oracle features $\mathbf{x}[V]$. Now, since cost is only involved in oracle queries, generating $\mathbf{x}'[V]$ from the generator leads to a reduced cost on $U\backslash V$. Here, we aim to draw $\mathbf{x}'[V]$ from the feature generator p_ϕ, conditioned on the observed features $\mathbf{x}[\mathcal{O}]$ and the rest of the features $\mathbf{x}[U\backslash V]$, where the latter is queried from the oracle. Formally, we have $\mathbf{x}'[V] \sim p(\bullet \,|\, \mathbf{x}[\mathcal{O}])$.

Problem statement During training, we are given the architectures of a classifier h and the feature generator p as well as the training set $\{(\mathbf{x}_i, y_i, \mathcal{O}_i)\}_{i\in D}$ and a budget $q_{\max}$ for maximum number of oracle queries for each instance. The budget is per instance since test instances occur in isolation. Our goal is to train h and p, as well as simultaneously compute the optimal values of U_i and V_i for each $i \in D$ and $|U_i\backslash V_i| \leq q_{\max}$, so that the oracle features $\mathbf{x}_i[\mathcal{O}_i \cup U_i\backslash V_i]$ and the generated features $\mathbf{x}'_i[V_i] \sim p(\bullet \,|\, \mathbf{x}[\mathcal{O}_i])$ provide high accuracy on h. In theory, one can encode the above task in the following training loss:

$$\text{loss}(h, p; U_i, V_i \,|\, \mathcal{O}_i) = \mathbb{E}_{\mathbf{x}'_i[V_i]\sim p(\bullet \,|\, \mathbf{x}_i[\mathcal{O}_i])}\Big[\ell\left(h\big(\mathbf{x}_i[\mathcal{O}_i \cup U_i\backslash V_i] \cup \mathbf{x}'_i[V_i]\big), y_i\right)\Big]; \tag{1}$$

and solve the following optimization problem.

$$\underset{h,p,\{V_i\subseteq\mathcal{I},U_i\subseteq\mathcal{I}\}_{i\in D}}{\text{minimize}} \sum_{i\in D} \text{loss}(h, p; U_i, V_i \,|\, \mathcal{O}_i) \tag{2}$$

$$\text{subject to,} \quad |U_i\backslash V_i| \leq q_{\max} \text{ for all } i \in D \tag{3}$$

If we could solve this problem, then, for a test instance, we can directly use the optimal U_i^* and V_i^* from the nearest training example.

There is no cost or budget associated with drawing features from the generator. Therefore, in principle, V_i can be as large as possible. However, a large V_i may not always be an optimal choice in practice, because the generator may be inaccurate. For example, even if the generated feature $\mathbf{x}'[V]$ is close to its gold value $\mathbf{x}[V]$, a small difference $|\mathbf{x}'[V]-\mathbf{x}[V]|$ may manifest in large prediction error (**??**).

Bottlenecks The above optimization problem involves simultaneous model training and selection of a large number of subsets $\{U_i, V_i\}$. As a result, it suffers from several bottlenecks as described below.

— *(1) Large number of sets as optimization variables:* The observed subset $\mathcal{O}_i$ varies widely across $i \in D$. Moreover, the observed feature values $\mathbf{x}_i[\mathcal{O}]$ for the same $\mathcal{O}$ also vary across instances. Hence, the optimal choice of U_i, V_i varies across instances, leading to $O(|D|)$ optimization variables.

— *(2) Heterogeneous feature space:* The final set of features that are fed into the classifier $\mathbf{x}_i[\mathcal{O}_i \cup U_i\backslash V_i]$ are very diverse, owing to a large variety of $\mathcal{O}_i$, U_i and V_i. This results in a number of heterogeneous domains, which makes it difficult for one single model to generalize across the entire data.

—*(3) Coupling between different optimization variables:* Two types of couplings exist between optimization variables U_i and V_i. Given one instance $i \in D$, the optimization variables U_i and V_i are coupled and so are U_i, V_i and U_j, V_j for different instances $i \in D$ and $j \in D$. This complexity renders the joint optimization problem (2) intractable.

3 Proposed approach

In this section, we introduce GENEX, a generator-assisted mixture of expert model addressing identified challenges. We present a tractable alternative to optimization problem (2) in three steps: (I) data partitioning, (II) designing mixture models, and (III) decoupling cross-instance coupling of optimization variables. Finally, we offer a set function centric characterization of this alternative optimization and a greedy algorithm for its solution.

3.1 Data partitioning

In the first step, we reduce the number of optimization variables (bottleneck 1) and transform the heterogeneous set of instances into homogeneous clusters (bottleneck 2).

Clustering methods like K-means and Gaussian mixture models maximize the *average* intra-cluster similarity. We observed that (Section 4) this has led to highly suboptimal partitioning, with fewer highly similar instances in one cluster and others with moderately high similarity in different clusters. To address this, we adopt a random hyperplane-based clustering technique, mitigating bucket imbalance and achieving more equitable cluster assignments.

Random hyperplane based clustering We partition observations $\{\mathbf{x}_i[\mathcal{O}_i]\}_{i\in D}$ into B clusters using Random Hyperplane (RH) guided Approximate Nearest Neighbor (ANN) clustering, employing locality-sensitive hashing (**?**). For this, we generate M independent spherically distributed normal vectors $\boldsymbol{W} = [\boldsymbol{w}_1, .., \boldsymbol{w}_M] \in \mathbb{R}^{n\times M}$, with $\boldsymbol{w}_m \sim N(0, \mathbb{I}_n)$. Each $\boldsymbol{w}_m$ defines a random hyperplane through the origin. We hash each observation $\mathbf{x}_i[\mathcal{O}_i]$ to a bucket $b = \text{sign}(\boldsymbol{W}^\top \mathbf{x}_i[\mathcal{O}_i])$, where $\mathbf{x}_i[\mathcal{O}_i]$ is padded with zeros to match dimensionality. Each bucket is thus identified by a vector of ± 1 of length M. This implies that $B \leq 2^M$. In our experiments we observe that instances are roughly uniformly distributed among the 2^M buckets, hence each bucket will roughly have $|D|/2^M$ instances, with $B = 2^M$. The probability that two observed features will share the same bucket, increases with their cosine similarity **?**, Section 3.

In contrast to K-means of GMM, RH has two key advantages. (1) It doesn't maximize any aggregate objective thus the assignment of one instance $\mathbf{x}$ doesn't affect the cluster assignment of another instance $\mathbf{x}'$ (2) The randomized algorithm encourages cluster diversity

Reducing the number of optimization variables The key reason behind the large number of optimization variables is fine grained choices of U_i and V_i for each instance $i \in D$. Here, we coarsen the estimate of these sets, by assigning the same optimization variables (U_b, V_b) for all the observed features falling in the same bucket b. This reduces the number of optimization variables to $O(B)$.

For a test example $\mathbf{x}[\mathcal{O}]$, we seek to find U_{b^*} and V_{b^*}, where the bucket b^* was assigned to the training instance i having the highest cosine similarity with $\mathbf{x}[\mathcal{O}]$. This bucket id can be immediately obtained by computing $b^* = \text{sign}(\boldsymbol{W}^\top \mathbf{x}[\mathcal{O}])$, without explicit nearest neighbor search (**?**). During our experiments, we observed that the above clustering method works better than K-means or gaussian mixture clustering. Moreover, in our method, computation of U_{b^*} and V_{b^*} for a test instance $\mathbf{x}[\mathcal{O}]$ admits $O(\log B)$ time complexity to compute the bucket id b^*, holding n constant. On the other hand, K-means or gaussian clustering admits $O(B)$ complexity.

3.2 Mixture models

Having partitioned the data, as described above, we train a mixture of models across these clusters, where each model is tailored specifically to generalize on each cluster. This addresses bottleneck (2) and (3).

Formulation of mixture models Given a partitioning of D into B buckets, *i.e.*, $D = D_1 \cup D_2 \cup ... \cup D_B$, we build a mixture of B independent classifiers h_{θ_b} and generators p_{ϕ_b} , parameterized with θ_b and ϕ_b for a bucket b. This reduces the joint optimization (2) problem into the following

$$\underset{\{\theta_b,\phi_b,U_b,V_b\}_{b\in[B]}}{\text{minimize}} \sum_{b\in[B]} \sum_{i\in D_b} \text{loss}(h_{\theta_b}, p_{\phi_b}; U_b, V_b \,|\, \mathcal{O}_i)$$
$$\text{such that, } |U_b \backslash V_b| \leq q_{\max} \ \forall b \in [|B|] \tag{4}$$

Decoupling cross-instance optimization variables It is evident that the above optimization (4) can be decoupled into B independent components. For each bucket b, we minimize $\text{loss}(h_{\theta_b}, p_{\phi_b}; U_b, V_b \,|\, \mathcal{O}_i)$, separately from other buckets. This reduces to the feature selection problem— the goal of selecting one fixed set of features for multiple instances (**?**). Thus, it leads us to overcome the cross-instance coupling between the model parameters, U and V. It also facilitates distributed implementation.

3.3 Decoupling optimization tasks over U_b and V_b

Overview $\text{loss}(h_{\theta_b}, p_{\phi_b}; U_b, V_b \,|\, \mathcal{O}_i)$— the objective in a bucket b— still involves a coupling between U_b, V_b. To overcome this, we build two optimization problems. We first compute the optimal U_b and then compute V_b based on the optimal value of U_b obtained. This addresses bottleneck (3).

Optimization over U_b For the first optimization, we design a new loss function $F(\theta_b, \phi_b; U_b \,|\, \mathcal{O}_b)$ whose optimal value with respect to U_b for a given θ_b and ϕ_b is an upper bound of the corresponding model training loss of the optimization (4). This loss is a combination of the prediction losses from the oracle and generated features, weighted by a prior uncertainty measure. This uncertainty is computed by pre-training the classifier and the generator on the observed data $\{(\mathbf{x}_i, y_i, \mathcal{O}_i)\}_{i\in D}$. Having computed the pre-trained classifier h_0 and the pre-trained generator p_0, we define $\Delta_i(U_b)$ as the uncertainty of the classifier when h_0 uses the generated features $\mathbf{x}'_i[U_b] \sim p_0(\bullet \,|\, \mathcal{O}_i)$ for the whole set U_b, *i.e.*, $\Delta_i(U_b) = \mathbb{E}_{\mathbf{x}'_i \sim p_0(\bullet \,|\, \mathbf{x}[\mathcal{O}_i])}[1 - \max_y h_0(\mathbf{x}_i[\mathcal{O}_i] \cup \mathbf{x}'_i[U_b])[y]]$. Thus, $\Delta_i(U_b) \in [0, 1 - \frac{1}{|C|}]$. We rescale $\Delta_i(U_b)$ by dividing it with $1 - \frac{1}{|C|}$, so that it lies in $[0, 1]$.

Algorithm 1: Training

```
Require: Training data $\{(\mathbf{x}_i, y_i, \mathcal{O}_i)\}_{i\in D}$, Number of buckets
    $B = 2^M$, the classifier architecture $h$ and generator architecture $p$.
1: $\{D_b\}_{b\in[B]} \leftarrow$ PARTITION$(D, B)$
2: for bucket $b \in B$ do
3:     $U_b^*, \theta_b^*, \phi_b^* \leftarrow$GREEDYFORU$(q_{\max}, b, F, G_F)$
4:     $V_b^*, \leftarrow$GREEDYFORV$(\lambda, b, G_{\text{loss}})$
```

```
1: function PARTITION$(D, B = 2^M)$
2:     $\boldsymbol{W} \leftarrow [\boldsymbol{w}_m]_{m\in[M]} \sim N(0, \mathbb{I}_n)$
3:     hash$[i] \leftarrow$ sign$(\boldsymbol{W}^\top \mathbf{x}_i[\mathcal{O}_i])$ for all $i \in D$
4:     $D_b \leftarrow \{i : \text{hash}[i] = b\}$
5:     Return $\{D_b\}_{b\in[2^M]}$
```

```
1: function GREEDYFORV$(\lambda, b, F, G_{\text{loss}})$
2:     Require: trained models $\theta_b^*, \phi_b^*$ and the optimal subset $U_b^*$
3:     $V_b \leftarrow \emptyset$
4:     for $q \in [\lambda]$ do
5:         $e^* \leftarrow \text{argmin}_{e\notin V_b\cup\mathcal{O}_i} G_{\text{loss}}(e \,|\, V_b)$
6:         if $G_{\text{loss}}(e^* \,|\, V_b) < 0$ then
7:             $V_b \leftarrow V_b \cup e^*$
8:             break
9:     Return $V_b$
```

Then, we define the new loss F as follows.

$$F(h_{\theta_b}, p_{\phi_b}; U_b \,|\, \mathcal{O}_i) = \Delta_i(U_b)\cdot \ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i \cup U_b]), y) \\ + (1-\Delta_i(U_b))\cdot \mathbb{E}_{\mathbf{x}'[U]}\ell\left(h_{\theta_b}\left(\mathbf{x}_i[\mathcal{O}_i]\cup \mathbf{x}'_i[U_b]\right), y_i\right) \quad (5)$$

Proposition 3.1 *Let F and* loss *are defined in Eqs. 5 and* (1)*, respectively. Then, we have:*

$$\min_{\{U_i,V_i\}:|U_i\setminus V_i|\le q_{\max}} \sum_{i\in D_i} \text{loss}(h, p; U_i, V_i \,|\, \mathcal{O}_i) \\ \le \min_{U_b:|U_b|\le q_{\max}} \sum_{i\in D_b} F(h, p; U_b \,|\, \mathcal{O}_i) \quad (6)$$

The set-optimal value of the above objective is an upper bound of loss() in Eq. (1), as stated formally here [1]. Hence, we instead of minimizing loss, we seek to solve the following optimization problem for each bucket b.

$$\min_{\theta_b,\phi_b,U_b} \sum_{i\in D_b} F(h_{\theta_b}, p_{\phi_b}; U_b \,|\, \mathcal{O}_i), \text{ s.t., } |U_b| \le q_{\max} \quad (7)$$

The objective loss$(\cdot)$ (1) queries two different sets of features from the oracle and the generator, *i.e.*, $U_b\setminus V_b$ and V_b. In contrast, F queries the same set of features U_b from both oracle and the generator. Here, it assigns more weights to the loss from the generated features (oracle features) if the pretrained classifier is less (more) uncertain from the generated features. In the absence of the generator, F only contains the loss for the oracle features, *i.e.*, $F(h_{\theta_b}, p_{\phi_b}; U_b \,|\, \mathcal{O}_i) = \ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i \cup U_b]), y)$ and the task reduces to the well-known feature selection problem in (**?**).

Optimization over V_b The above optimization involves only U_b as the optimization variables, and is independent of V_b. Once we compute $\theta_b^*, \phi_b^*, U_b^*$, *i.e.*, the solution of the optimization (7), we use them to compute the set V_b by solving the following optimization problem:

$$\min_{V_b:|V_b|\le\lambda} \sum_{i\in D_b} (1-\Delta_i(V_b))\cdot \text{loss}(h_{\theta_b^*}, p_{\phi_b^*}; U_b^*, V_b \,|\, \mathcal{O}_i) \quad (8)$$

where λ is a hyperparameter.

[1] Proofs of all technical results are in Appendix D

```
1: function GREEDYFORU$(q_{\max}, b, h, p)$
2:     $U_b \leftarrow \emptyset,$
3:     $\theta_b \leftarrow$ TRAIN$(h, D_b, \{\mathcal{O}_i\})$ #pretraining
4:     $\phi_b \leftarrow$ TRAIN$(p, D_b, \{\mathcal{O}_i\})$ #pretraining
5:     for iter $\in [q_{\max}]$ do
6:         # Use $h_{\theta_b}$, $p_{\phi_b}$ to compute $G_F, F$
7:         $e^* \leftarrow \text{argmin}_{e\notin U_b\cup\mathcal{O}_i} G_F(e \,|\, U_b)$
8:         if $G_F(e^* \,|\, U_b) < 0$ then
9:             $U_b \leftarrow U_b \cup e^*$
10:            $\theta_b, \phi_b \leftarrow$ TRAIN$(F, U_b, b)$
11:        else
12:            break
13:    Return $U_b, \theta_b, \phi_b$
```

Algorithm 2: Inference

```
Require: Observed test feature, $\mathbf{x}[\mathcal{O}]$, threshold $\tau$, trained models
    $h_{\theta_b^*}, p_{\phi_b^*}, \{U_b^*, V_b^*\}$.
1: $b \leftarrow$ FINDBUCKET$(\mathbf{x}^*[\mathcal{O}])$
2: Query $\mathbf{x}[U_b^*\setminus V_b^*]$ from oracle
3: $\mathbf{x}'[V_b^*] \sim p_{\phi_b^*}(\bullet \,|\, \mathbf{x}^*[\mathcal{O} \cup U_b^*\setminus V_b^*])$
4: $\mathbf{x}_{\text{all}} = \mathbf{x}[\mathcal{O} \cup U_b^*\setminus V_b^*] \cup \mathbf{x}'[V_b^*]$
5: if $\max_y h_{\theta_b^*}(\mathbf{x}_{\text{all}})[y] < \tau$ then
6:     Query $\mathbf{x}[V_b^*]$ from oracle, $\mathbf{x}_{\text{all}} = \mathbf{x}[\mathcal{O} \cup U_b^*]$
7: $y^* \leftarrow \text{argmax}_y h_{\theta_b^*}(\mathbf{x}_{\text{all}})[y]$
8: Return $y^*$
```

Objectives in (7), (8) as set functions Here, we represent the objectives in the optimizations (7), (8) as set functions, which would be later used in our training and inference methods and approximation guarantees. Given U, the optimal solution of the objective $\theta_b^*(U), \phi_b^*(U)$ minimizes $\sum_{i\in D_b} F(h_{\theta_b}, p_{\phi_b}; U \,|\, \mathcal{O}_i)$. Thus, $\min_{\theta_b,\phi_b} \sum_{i\in D_b} F(h_{\theta_b}, p_{\phi_b}; U \,|\, \mathcal{O}_i)$ becomes a set function in terms of U. On the other hand, $\sum_{i\in D_b}(1-\Delta_i(V))\cdot$ loss$(h, p; U_b^*, V \,|\, \mathcal{O}_i)$ is also a set function in terms of V. To this end, we define G_F and G_{loss} as follows given any set U we describe the optimal value of this training problem as the following set function:

$$G_F(U) = \sum_{i\in D_b} F(h_{\theta_b^*(U)}, p_{\phi_b^*(U)}; U \,|\, \mathcal{O}_i), \quad (9)$$

$$G_{\text{loss}}(V) = \sum_{i\in D_b} (1-\Delta_i(V))\text{loss}(h_{\theta_b^*}, p_{\phi_b^*}; U_b^*, V \,|\, \mathcal{O}_i) \quad (10)$$

3.4 Training and inference algorithms

Training Our training algorithm uses greedy heuristics combined with model training to solve the optimization problems (7) and (8) for computing $\theta_b^*, \phi_b^*, U_b^*, V_b^*$. Given the dataset D and the number of buckets B, we first partition the dataset into B buckets (PARTITION$(\cdot)$) and perform pre-training of h_{θ_b} and p_{ϕ_b} on D_b based on the observed features $\{\mathcal{O}_i\}$. For the generator, we use a $\beta-$VAE (**?**) to optimize regularized ELBO. Then, for each bucket b, we leverage two greedy algorithms for non-monotone functions (**??**), one for computing U_b (GREEDYFORU) and the other for V_b (GREEDYFORV). At each iteration, GREEDYFORU keeps adding a new feature $e = e^*$ to U_b which minimizes $G_F(e \,|\, U_b)$ as long as it admits a negative marginal gain, *i.e.*, $G_F(e^* \,|\, U_b) < 0$ and $|U_b| \le q_{\max}$. Similarly, we use the greedy algorithm on G_{loss} to get V_b. GREEDYFORU needs to train the model for every candidate for each new element. To tackle this, in practice we adopt three strategies. (1) We directly use the model parameters θ_b, ϕ_b obtained in the previous step to compute G_F during search of the potential new candidates e (line 5), without any new training. After we select e^*, we perform two iterations of training. Since we consider adding only one element, such a small amount of training is enough, beyond which we did not see any observable improvement. (2) We tensorize the operation $\text{argmin}_e\ G_\bullet(e \,|\, U_b)$ instead of enumerating $G_\bullet$ for all candidates e. Note that we utilize all the features available in the training set during training. However, this does not amount to cheating, as in the inference algorithm, only the required features are queried. Such a protocol is widely followed in works including (**?**) and (**?**), helping them to generalize to the set of all features.

Inference Given a test instance with a subset of observed features $\mathbf{x}[\mathcal{O}]$, we first find the bucket b. It then queries the oracle for $\mathbf{x}[U_b \backslash V_b]$. Taking $\mathbf{x}[\mathcal{O} \cup U_b \backslash V_b]$ as input, the generator produces $\mathbf{x}'[V_b]$. If the confidence of the classifier with these features is lower than a threshold τ, then GENEX does not use the generated features for the final prediction and, further query $\mathbf{x}[V_b]$ from the oracle and predict y using $\mathbf{x}[\mathcal{O} \cup U_b]$. However, otherwise if the confidence of the classifier with the features is higher than τ, we use the generated features and compute y using $\mathbf{x}[\mathcal{O} \cup U_b \backslash V_b] \cup \mathbf{x}'[V_b]$.

3.5 Characterization of our optimization tasks

Here, we present a set function based characterization of our objectives (7) and (8) (or, equivalently, (9) and (10)), beginning with a discussion on hardness analysis. Then, we use those characterizations to prove the approximation guarantee of our algorithms.

Hardness At the outset, our goal is to first compute the optimal $U = U^*$ by minimizing $G_F(U)$ and then use this U^* to compute the optimal V by minimizing $G_{\text{loss}}(V)$. The optimization of $G_F(U)$ —– or, equivalently the optimization (7)— is a discrete continuous optimization problem, since it involves model training in conjunction with subset selection. Given U, one can find the optimal solution of the objective $\theta_b^*(U), \phi_b^*(U)$ in polynomial time when the objective is convex with respect to both θ and ϕ. However, the simultaneous computation of the optimal set U^* and the model parameters θ_b^* and ϕ_b^* is NP-Hard even in simple cases, *e.g.*, sparse feature selection (**?**).

Set function centric characterizations We first extend the notions of partial monotonicity (**?**) and γ-weak submodularity (**??**).

Definition 3.2 *Given a set function $G : 2^{[n]} \to \mathbb{R}$, two sets S and T with $S, T \subseteq [n]$ and the marginal gain $G(S \,|\, T) := G(S \cup T) - G(T)$. Then we define the following properties. **(1)** $(\underline{m}, \overline{m})$-Partial monotonicity: The set function G is $(\underline{m}, \overline{m})$-partially monotone ($\underline{m} \geq 0$) if $\frac{G(T)}{G(S)} \in [\underline{m}, \overline{m}]$ for all S, T with $S \subseteq T \subseteq [n]$. **(2)** $(\underline{\gamma}, \overline{\gamma})$-Weak submodularity: The set function G is $(\underline{\gamma}, \overline{\gamma})$-weakly submodular if $\frac{\sum_{u \in S} G(u \,|\, T)}{|G(S \,|\, T)|} \in [\underline{\gamma}, \overline{\gamma}]$ for all S, T with $S \cap T = \emptyset$.*

Similar to **?**, we define that $G(T)/G(S) = 1$ if $G(S) = 0$. Note that, a $(\underline{m}, \overline{m})$ partially monotone function G is monotone increasing (decreasing) if $\underline{m} = 1$ ($\overline{m} = 1$). Moreover, $m-$partial monotonicity introduced in (**?**) implies (m, ∞)-partial monotonicity. A $\gamma-$weakly submodular function (**??**) is (γ, ∞)-weakly submodular. Next, we assume boundedness of few quantities, allowing us to characterize G_F and G_{loss}.

Assumption 3.3 *(1) Bounded difference between uncertainties across two feature subsets: Given a bucket $b \in [B]$, $|\Delta_i(U) - \Delta_i(V)| \leq \beta_\Delta$. (2) Bounds on uncertainty and loss: $0 < \Delta_{\min} \leq |\Delta_i(U)| \leq \Delta_{\max}$. $0 < \ell_{\min} \leq |\ell(h_\theta(\mathbf{x}_i), y_i)| \leq \ell_{\max}$. (3) Lipschitzness: The loss function $\ell(h_\theta(\mathbf{x}), y)$ is Lipschitz with respect to $\mathbf{x}$. (4) Boundedness of features: $||\mathbf{x}_i||$, $||\mathbf{x}'_i|| \leq \beta_x$, for all i. In Appendix E, we discuss the validity of these assumptions as well as present the values of $\beta_\bullet$ across different datasets.*

Theorem 3.4 (**$(\underline{m}, \overline{m})$-Partial monotonicity**) *(1) The set function G_F is $(\underline{m}_F, \overline{m}_F)$-partially monotone where $\overline{m}_F = 1 + K_F \beta_x \beta_\Delta$ and $\underline{m}_F = (1 + K_1 \beta_x + K_2 \beta_\Delta + K_3)^{-1}$. (2) The set function G_{loss} is $(\underline{m}_{\text{loss}}, \overline{m}_{\text{loss}})$-partially monotone where $\overline{m}_{\text{loss}} = 1 + K_{\text{loss}} \beta_x$ and $\underline{m}_{\text{loss}} = (1 + K_{\text{loss}} \beta_x)^{-1}$. Here $K_\bullet$ depend on the Lipschitz constant of the loss with respect to $\mathbf{x}$ and the bounds on loss ℓ and the uncertainty Δ.*

Partial monotonicity of G_F suggests that if the variation of uncertainty across different feature sets goes small ($\beta_\Delta \to 0$) or the generator is extremely accurate ($\beta_x \to 0$), then we have: $\overline{m} \to 1$, meaning that G_F is monotone decreasing. If we put $\Delta_i(U_b) = 1$ for all $i \in D_b$ in the expression of Fin Eq. (5), then the optimization (7) becomes oblivious to the generative model p_{ϕ_b}. In such a case, G_F is monotone decreasing since, $\beta_\Delta = 0$. This result coincides with the existing characterizations of the traditional feature selection problem (**?**). In the context of the optimization for V, note that if the generator is very accurate ($\beta_x \to 0$), then Partial monotonicity of G_{loss} implies that $G_{\text{loss}}(V)$ is almost constant for all V. In such a case, one can use the feature generator to generate the entire set $V_b = U_b^*$ and save the entire budget $q_{\max}$. Since, we have $G_{\text{loss}}(S \,|\, T) = 0$ in a simple cases where β_x or $\beta_\Delta = 0$, $(\underline{\gamma}, \overline{\gamma})$-weak submodularity does not hold for G_{loss} in most cases. Thus, we present the $(\underline{\gamma}, \overline{\gamma})$-weak submodularity property of only G_F under additional assumptions.

Theorem 3.5 (**$(\underline{\gamma}, \overline{\gamma})$-weak submodularity**) *Assume that the loss $\ell(h_\theta(\mathbf{x}), y)$ be convex in θ with $\nabla_\theta \ell(h_\theta(\mathbf{x}), y) \leq \nabla_{\max}$ and Eigenvalues$\{\nabla^2_\theta \ell(h_\theta(\mathbf{x}[S]), y)\} \in [\zeta_{\min}, \zeta_{\max}]$ for all S; Then, the set function G_F is $(\underline{\gamma}_F, \overline{\gamma}_F)$-weakly* *submodular with*

$$\overline{\gamma}_F \le \max\left\{ \frac{-\frac{\nabla^2_{\max}}{2\zeta_{\max}} + K_{\overline{\gamma},1}L_\phi + K_{\overline{\gamma},2}\beta_\Delta\beta_x}{\frac{\nabla^2_{\max}}{2\zeta_{\min}} + K_{\overline{\gamma},3}L_\phi + K_{\overline{\gamma},4}\beta_\Delta\beta_x}, \frac{-\frac{\nabla^2_{\max}}{2\zeta_{\max}} + K_{\overline{\gamma},1}L_\phi + K_{\overline{\gamma},2}\beta_\Delta\beta_x}{\frac{\nabla^2_{\max}}{2\zeta_{\max}} - K_{\overline{\gamma},5}L_\phi - K_{\overline{\gamma},6}\beta_\Delta\beta_x}\right\};$$

$$\underline{\gamma}_F \ge \frac{-\frac{\nabla^2_{\max}}{2\zeta_{\min}} - K_{\underline{\gamma},1}L_\phi - K_{\underline{\gamma},2}\beta_\Delta\beta_x}{\frac{\nabla^2_{\max}}{2\zeta_{\max}} - K_{\underline{\gamma},3}L_\phi - K_{\underline{\gamma},4}\beta_\Delta\beta_x} \qquad (11)$$

where $K_{\overline{\gamma},\bullet}$ and $K_{\underline{\gamma},\bullet}$ are constants that depend on the Lipschitz constant of the loss w.r.t. $\mathbf{x}$; L_ϕ is the Lipschitz constant of F w.r.t. ϕ.

In the absence of the generator, $\beta_\Delta, \beta_x, L_\phi$ are zero. Then, $-G_F$ is γ-weakly submodular with $\gamma > \zeta_{\min}/\zeta_{\max}$ which coincides with the results of (**?**). Next, we present the approximation guarantee of our greedy algorithm for solving the optimization problem (7) presented via GREEDYFORU, when $\ell(\boldsymbol{h}_\theta(\mathbf{x}), y)$ is convex in θ.

Theorem 3.6 *Assume that, given a bucket b, G_F is $(\underline{m}_F, \overline{m}_F)$-partially monotone, $(\underline{\gamma}_F, \overline{\gamma}_F)$-weakly submodular set function. Suppose U_b^* is the output of* GREEDYFORU *in Algorithm 1. Then, if $OPT = \text{argmin}_{U_b : |U_b| \le q_{\max}} G_F(U_b)$, we have*

$$G_F(U_b^*) \le m_F G_F(OPT) - \left(1 - \frac{\gamma_F}{q_{\max}}\right)^{q_{\max}} \Big(m_F G_F(OPT) - G_F(\emptyset)\Big)$$

where $m_F = \max(\overline{m}_F, 2\overline{m}_F/\underline{m}_F)$ and $\gamma_F = \max(\overline{\gamma}_F, -\underline{\gamma}_F)$.

We discuss the quality of the approximation guarantees for different datasets in our experiments in Appendix E.

4 Experiments

Datasets We experiment with four datasets for the classification task; DP (disease prediction), MNIST, CIFAR100 and TinyImagenet (TI). For DP, the number of features $n = 132$ and the number of classes $|\mathcal{Y}| = 42$. Here, different features indicate different symptoms and the classes indicate different diseases. For the image datasets, we take the pixels or groups of pixels as the features, following previous work (**??**). Further details about the datasets are provided in Appendix F. For each dataset, the average number of observed features $\mathbb{E}[|\mathcal{O}_i|] \approx n/10$.

Baselines We compare GENEX with six state-of-the art baselines: JAFA (**?**), EDDI (**?**), ACFlow (**?**), GSM (**?**), CwCF (**?**) and DiFA (**?**) each with two variants (batch and sequential). In the first variant, we deploy these methods to perform in the batch setting. Specifically, we use top-$q_{\max}$ features provided by the feature selector in batch, instead of querying the oracle features one by one. This feature selector is a policy network in RL methods (**????**), greedy algorithms for maximizing rewards in (**??**). Note that our approach uses mixture models on the partitioned space, which enhances the expressivity of the overall system. To give the baselines a fair platform for comparison, we deploy a mixture of classifiers on the same partitioned space, where the models are trained using the observed features and the corresponding queried features of the baselines. This is done in the cases where, off-the-shelf use of their methods led to poor performance, especially in the larger datasets (Appendix G). In the sequential variant, we make the baselines to acquire features sequentially. Since our goal is to acquire features in batch, the first setting gives a fairer platform for comparison across all methods. Thus, we only present the results of the batch setting in the main and, defer the results of the sequential setting in Appendix G

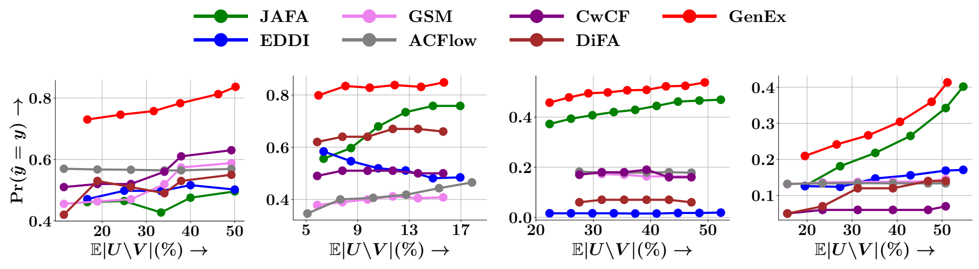
Classifier h and the generator p For DP and MNIST, the classifier h consists of linear, ReLU and linear networks, cascaded with each other, with hidden dimension 32. For CIFAR100, h is WideResnet (**?**) for TinyImagenet, h is EfficientNet (**?**). We use the same classifier h across all baselines too. The generator p is a variational autoencoder. It contains an encoder and a decoder that are pre-trained on the observed instances as a β-VAE (**?**). Details are provided in Appendix F. We set the number of buckets $B = 8, 8, 4, 4$ for DP, MNIST, CIFAR100 and TinyImagenet respectively, which we set using cross validation.

Evaluation protocol We split the entire dataset in 70% training, 10% validation and 20% test set. We use the validation set to cross validate λ and the number of buckets B. Having observed a feature $\mathbf{x}[\mathcal{O}]$ during test, we use the learned method to compute the set of new features to be acquired U and V for a given budget $q_{\max}$. We use $\mathbf{x}'[V]$ by drawing from the generator, whereas we query $\mathbf{x}[U \backslash V]$ from the oracle. Finally, we feed all the gathered feature into the classifier and compute the predicted label $\hat{y}$. We cross validate our results 20 times to obtain the p-values.

4.1 Results

Comparison with baselines First, we compare the prediction accuracy of GENEX against the baseline models for different value of the maximum permissible number of oracle queries $q_{\max}$ per instance. The horizontal axis indicates $\mathbb{E}[|V|/|U|]$, the average number of oracle queries per instance. Figure 1 summarizes the results. We observe: **(1)** GENEX outperforms all these baselines by a significant margin. The competitive advantage provided by GENEX is statistically significant (Welch's t-test, p-value $< 10^{-2}$). **(2)** JAFA performs closest to ours in large datasets. **(3)** The baselines are not designed to scale to a large number of features, as asserted in the classification experiments in (**?**), and hence their accuracy stagnates after acquiring a few features (**?**, Fig. 6, 7) and (**?**, Fig. 3).

Ablation study on the generator To evaluate the magnitude of cost saving that our generator provides, we compare GENEX against its two variants. (I) GENEX ($V = \emptyset$): Here, all the features U of all the test instances are queried from the oracle. (II) GENEX ($V = U$): Here, whenever an instance is qualified to have the features from the generator (the classifier confidence on the generated feature is high), all the features U are drawn from the generator. Figure 2 shows the results in terms of accuracy vs. the budget of the



(a) DP (b) MNIST (c) CIFAR100 (d) TinyImageNet

Figure 1: Comparison of GENEX against batch variants of baselines, *i.e.*, JAFA (**?**), EDDI (**?**), ACFlow (**?**), GSM (**?**), CwCF (**?**) and DiFA (**?**) in terms of the classification accuracy varying over the average number of oracle queries $\mathbb{E}|U\backslash V|$, for all four datasets.

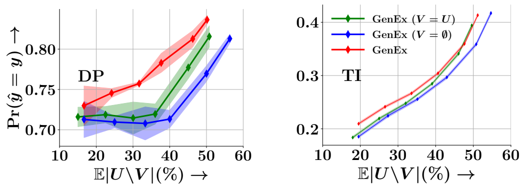


Figure 2: Ablation study: $\Pr(\hat{y} = y)$ vs. $\mathbb{E}[|U\backslash V|]$

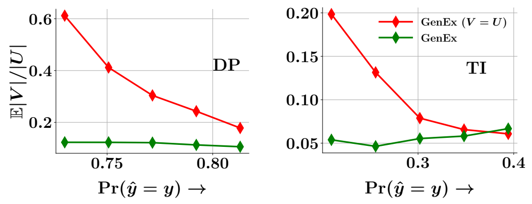


Figure 3: Ablation study: Saved cost vs. accuracy.

oracle queries. Figure 3 shows the results in terms of the average fraction of the saved budget $\mathbb{E}[|V|/|U|]$. Note that, even in GENEX ($V = U$), not all instances result in high confidence on the generated features. In case of low confidence, we query the features from the oracle. We make the following observations: **(1)** GENEX outperforms the other variants in most of the cases in terms of accuracy for a fixed budget (Fig. 2). At the places where we perform better, the gain is significant ($p < 0.05$ for $\mathbb{E}[U\backslash V] \leq 20\%$ for DP; $p < 0.01$ for others). **(2)** GENEX is able to save 3–5x cost at the same accuracy as compared to the GENEX ($V = U$) variant. **(3)** The fractional cost saved goes down as accuracy increases, since U increases, but λ is fixed.

RH vs. other clustering methods Here, we compare random hyperplane (RH) guided clustering method with K-means and Gaussian mixture based clustering methods. The results are summarized in Figure 4 for DP and CIFAR100 datasets. We observe that RH performs better for a wide range of oracle queries $\mathbb{E}[|U\backslash V|]$. We note that the amount of bucket-skew, *i.e.*, the ratio of the minimum and maximum size of buckets, is significantly better for RH than K-means and GMM. Specifically, for DP dataset, this ratio is 0.21, 0.014 and 0.003 for RH, K-means and GMM, respectively. Thus, RH has a significantly better bucket balance than other methods. To further probe why the RH achieves a better bucket balance,

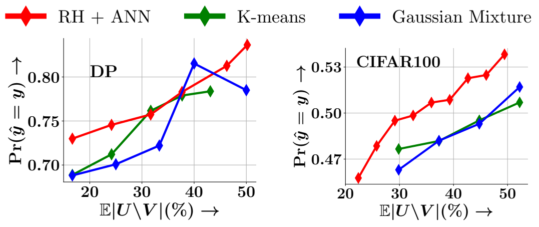


Figure 4: RH vs. other clustering methods

we instrument the conicity of the features which is a measure of how the feature vectors are concentrated in a narrow cone centered at the origin (**?**). This is defined as: $\text{conicity}(D) = \frac{1}{|D|}\sum_{i\in D}\cos\left(\mathbf{x}_i, \sum_{j\in D}\mathbf{x}_j/|D|\right)$. We observe a low conicity of < 0.2, indicating a high spread of feature vectors. Since, on an average, the random hyperplanes cut the space uniformly across the origin, the observed feature vectors get equally distributed between different hyperplanes, leading to good bucket balance. Conversely, K-means and GMM maximize the "mean" of similarity, promoting a few highly similar points in one cluster and leaving moderately similar instances dispersed among different clusters. These methods tend to exhibit inter-cluster cosine similarities of at least 0.92, suggesting that these instances should ideally belong to the same cluster. However, due to their objective to maximize a single aggregate measure, K-means and GMM often group extremely similar items together while separating others.

5 Conclusion

We proposed GENEX, a model for acquiring subsets of features in a batch setting to maximize classification accuracy under a budget constraint. GENEX relies on a mixture of experts model with random hyperplane guided data partitioning and uses a generator to produce subsets of features at no additional query cost. We employ a greedy algorithm that takes the generated features into account and provides feature subsets for each data partition. We also introduce the notions of $(\underline{m}, \overline{m})$-partial monotonicity and $(\underline{\gamma}, \overline{\gamma})$-weak submodularity, and provide a theoretical foundation for our method. GENEX is superior to the baselines, outperforming them in accuracy at a fixed budget. We recognize that a limitation of our work is that the guarantee of the greedy algorithm holds under certain assumptions, which are an artifact of the complexity of the problem. Further work can be done incorporating explicit exploration-exploitation on the greedy strategy.

Ipsa cumque voluptatem ab magni tempora illum perspiciatis, enim deserunt saepe laborum perferendis corrupti alias ducimus debitis sed reprehenderit, aut neque sapiente, officia nam aliquam ipsa tenetur, esse maxime quaerat.Fuga autem veniam expedita officia fugiat odit quam culpa molestiae sit velit, veniam voluptas deserunt eligendi ipsum dolorum necessitatibus aliquid.Beatae eum nostrum dignissimos tempora voluptatibus aliquam atque veritatis libero quae adipisci, inventore ratione non suscipit exercitationem praesentium rem dolor possimus molestiae, et eos voluptates dignissimos recusandae expedita voluptatum rem suscipit dicta voluptatibus, voluptatibus aliquam nam nihil doloremque, quae cupiditate distinctio?Reprehenderit odit nihil doloremque minima nostrum sapiente eos a, temporibus adipisci asperiores, ex similique obcaecati et quia qui in, cumque tempora adipisci voluptatum alias.Repellat unde repellendus est tempore nisi tempora dolorum, in asperiores temporibus commodi quidem quas aperiam libero rem nisi, sunt inventore maxime omnis autem ea at assumenda, neque illum quo.Error accusantium omnis quis officia velit excepturi consequuntur expedita sequi incidunt, neque accusamus molestias impedit sed ex magni eaque voluptatibus odit?Harum earum eum quasi totam repellendus, dolorem vero nam nisi, quod neque odit architecto facere?Repellat rem natus omnis sed, minima quo aut voluptates magni excepturi reprehenderit illo eum corrupti tenetur provident, distinctio perferendis quidem cum earum corrupti rem dolor, voluptatem illo nisi animi voluptatibus culpa laboriosam possimus.Quas debitis in nesciunt, quisquam iste sint pariatur at minus asperiores commodi placeat, officiis voluptates blanditiis esse, ullam provident error obcaecati numquam?Officia illo quo id error fugit earum porro nemo aspernatur itaque vel, officia dignissimos nam ex praesentium eaque nisi qui vitae dolore, eius necessitatibus facilis officia voluptate non vitae earum neque magni explicabo sequi?Ex perspiciatis harum alias ipsam sit pariatur vel asperiores, repellendus quidem debitis tempore incidunt a saepe, totam reprehenderit laboriosam dolor inventore eius ut aliquam libero vero, harum ut eius non a recusandae amet quam corrupti fugiat quo?Odit repellat magnam possimus provident quod perspiciatis fugiat excepturi quas exercitationem, ab atque magni esse impedit dolorum, ratione esse earum quibusdam repellendus autem quaerat dolorem harum.Cumque modi explicabo, quia repellat accusamus ad ex quasi amet a asperiores delectus praesentium doloremque?Voluptatem ratione vel quo quae libero, fuga qui eum quasi quo laboriosam reiciendis, consequuntur iste maxime amet error molestiae rerum tenetur?Corporis ab recusandae, reprehenderit alias impedit amet at sed voluptate saepe sunt, possimus facere voluptate quia unde sed quis ab cupiditate, illum incidunt vel vitae, atque ullam quos quidem enim totam eaque harum nesciunt?Veniam harum voluptatem autem aperiam nihil explicabo, saepe consequatur neque minus itaque a consectetur odit suscipit, velit reiciendis rem autem quibusdam beatae reprehenderit quia culpa ipsum temporibus veritatis.Omnis pariatur reprehenderit repellat quod deserunt quasi voluptatum temporibus, quos itaque vel nihil iste nobis quo similique nulla blanditiis incidunt, temporibus tempore doloribus at autem unde facere provident magnam.Accusantium quas explicabo harum tenetur porro odio modi, cum sint in fugit recusandae explicabo optio voluptate ea necessitatibus quibusdam eos, aliquid provident sequi dolorum qui.Quis in atque rerum possimus assumenda, voluptatibus voluptates alias nobis quas repudiandae aliquid iure minus voluptate.Nam ex repudiandae mollitia maiores tempora inventore ullam distinctio, nisi est sapiente, ut eveniet quaerat officiis ea provident iste quos officia?Harum alias dolor perspiciatis, blanditiis atque fugit quis voluptates eum nesciunt incidunt ut, et debitis earum vitae odit cupiditate laudantium quae sapiente veritatis officiis ex, ipsam ad fugiat ullam suscipit dolorum totam veniam omnis corporis inventore, obcaecati id perspiciatis natus a facere ad exercitationem adipisci impedit illo.Impedit quia similique autem incidunt, ex soluta iure sapiente ad aliquid mollitia laborum, mollitia magnam illum praesentium atque, recusandae magnam ea repudiandae voluptatum expedita repellendus aliquid omnis iure cum quaerat.Vel consequuntur inventore voluptate enim voluptatibus, tempore quos nemo beatae voluptas vero fuga deserunt aliquam odit voluptatem, esse dolore rerum aperiam.Adipisci libero voluptatum ratione fugiat qui eum itaque iusto, quam voluptatem ducimus.Vero alias neque inventore iusto, rerum beatae fuga deleniti autem perspiciatis nihil, necessitatibus quis incidunt eius sequi.Officia perspiciatis dignissimos dolorem at accusamus repudiandae quaerat, aspernatur aliquid quis amet rerum nostrum impedit, officia deleniti perferendis beatae mollitia ea repellendus quisquam eligendi facilis est, illo sit ut numquam inventore ea iste sequi dolorem adipisci, quidem nostrum tempore voluptates?Nam voluptatum rem alias perspiciatis voluptatibus facere enim provident blanditiis omnis minus, eos sunt commodi harum.Quasi eius provident dolorum illo veniam quisquam architecto velit recusandae consectetur, velit voluptatem sint repellat ab numquam tempora, aliquam iste atque molestias doloribus voluptatum pariatur explicabo incidunt obcaecati, quas assumenda odit libero quasi consectetur, facilis tempora velit ut sit ducimus quos mollitia quisquam obcaecati?Inventore optio omnis consequuntur necessitatibus illum neque, laborum unde earum odio soluta, praesentium enim sed nulla ipsa voluptatibus?Quod debitis iusto asperiores unde fugiat voluptatum, recusandae laborum rem, quod deserunt aperiam?Itaque minima error, facilis error incidunt sit dicta provident atque non ipsa numquam iusto fugit, esse excepturi culpa?Perspiciatis quidem at amet alias, hic voluptatem expedita a consequatur esse fuga, laudantium fugit porro cumque eos minus molestiae asperiores necessitatibus iusto quam odit, eum repudiandae quos tempore ratione sequi iure labore obcaecati.Facilis nesciunt eligendi, odio cupiditate officiis accusantium modi natus laboriosam a qui labore reiciendis, quo dolor aut ullam a alias libero.Similique provident amet velit vel voluptas fugiat reiciendis necessitatibus, earum corporis rem ipsum?Accusantium fugit in repellat ipsum aut iusto esse repellendus at, inventore est quisquam cupiditate nesciunt, aliquid est consequatur fugit sapiente harum eaque repellendus eius.Vitae modi magnam, cupiditate asperiores magni officiis repudiandae ad error odit itaque corrupti aperiam odio, sunt voluptate dolorum illum autem, labore amet quibusdam aut illum reprehenderit nam aliquam quidem minus atque consequatur.Soluta fuga earum quaerat a temporibus quasi voluptates voluptatem molestias sunt, vero recusandae qui rerum autem doloremque amet quam veniam quisquam illum, minima autem officiis quod odit aut voluptatibus, dolorum assumenda amet minima id fuga impedit atque ex minus molestiae ea, explicabo quas facilis sit earum veritatis quidem obcaecati.Earum voluptatem odio fugit aliquid, consectetur nisi labore blanditiis, quos molestias vitae, nisi voluptatibus natus commodi quisquam veniam id molestias.Tempora itaque sed, illo doloribus rerum.Maiores corporis in, hic sit esse placeat, pariatur voluptate sunt reiciendis explicabo impedit sapiente eveniet repudiandae facilis, facilis nisi reiciendis, natus ad temporibus perspiciatis repellendus maiores delectus.Esse consequuntur aut velit, vel cupiditate nostrum eos quae laboriosam dicta laudantium, cupiditate animi adipisci fugit quasi officiis dolore, consequuntur nihil voluptatem cum exercitationem distinctio reprehenderit sunt dicta repudiandae blanditiis numquam, quam sed dolor ipsa nisi provident?Modi fuga quae commodi ad laudantium vero mollitia error atque alias, expedita ullam cupiditate sunt unde sequi cumque, optio saepe vel quas illum est quasi iure eius perferendis doloribus, adipisci qui voluptatum nostrum architecto minima rem unde cum error, explicabo cumque pariatur velit adipisci quia architecto natus quibusdam deserunt officia nisi?Sapiente atque quod sunt unde quaerat eligendi placeat inventore deserunt voluptatem, reiciendis voluptatem magnam hic quo voluptates excepturi asperiores nostrum blanditiis, ex voluptas dolor pariatur perferendis enim mollitia blanditiis expedita inventore tempore dolorem.Consectetur blanditiis quia reiciendis eveniet, possimus tempore deserunt ad consequatur qui numquam assumenda molestiae, molestias hic neque, accusamus aperiam voluptatum dolores explicabo?Illo sit quaerat saepe excepturi ullam inventore dolore aspernatur facere fugit, nemo inventore cumque error sint ex maxime veniam, nulla quisquam dignissimos ullam incidunt, asperiores neque nam rem autem amet aspernatur numquam est culpa minus suscipit, placeat hic nobis id?Praesentium doloremque illum, numquam vitae ipsum tenetur neque quia iusto, facere consectetur sit veniam maxime eaque ea ullam nemo qui?In tempora quasi vel repellat asperiores quisquam commodi error ex, perspiciatis corporis quam necessitatibus libero maxime omnis minus suscipit autem amet sequi, similique quos ut accusantium, aliquam neque architecto maxime rem autem?Tempora assumenda corrupti harum aliquid, commodi minima sequi minus eveniet quod recusandae, maxime numquam fugiat aliquid ut, cum aliquam fuga minima libero dolore at eos deserunt, molestias natus laudantium nam consequuntur cum tempore distinctio nemo vel?Excepturi voluptates at pariatur commodi beatae sapiente nemo aperiam libero quaerat quae, dolores molestias enim quo cumque qui iure, ab quia aut ea architecto, exercitationem voluptatem nobis.Sit dolor nihil, eveniet suscipit veritatis odit ratione minima consequatur, reprehenderit iste iure in, hic reiciendis debitis quidem error illum iusto earum, voluptas eum pariatur laboriosam voluptate praesentium nulla harum quo?Sapiente harum natus deleniti expedita ab consequatur, ratione corporis corrupti quaerat dolores, debitis eos repudiandae aperiam vero, ducimus odit quia a quod quisquam provident vitae veniam debitis culpa vero?Et fuga dolore, rerum nostrum iure quis rem delectus?Cupiditate expedita provident eaque obcaecati optio error iure, sequi ab velit rem inventore alias vero tempore cupiditate deleniti cum, in et eveniet quas iusto doloremque distinctio molestias perspiciatis.Voluptatibus reiciendis nam nemo perferendis ad, consectetur quaerat et facere quis voluptate quibusdam doloribus tempore, tenetur labore officia neque cupiditate molestias exercitationem autem eum, neque incidunt vitae?Rerum voluptatum perspiciatis molestiae quibusdam facere, vel assumenda corrupti unde eaque ratione id quos labore ex, voluptatibus tempora commodi praesentium consequatur vitae et dignissimos, quo vero nobis quod iusto, blanditiis vero cum enim ex eaque perspiciatis minima omnis quo?Numquam ab minima placeat possimus accusantium, quod quia accusamus eaque doloremque praesentium dolores sit laudantium dolore autem?Fugiat assumenda dignissimos labore repudiandae iste omnis atque voluptatem vero nobis, iste cupiditate exercitationem excepturi quisquam sed enim libero mollitia dicta, nobis repellat dolores, ipsa numquam nesciunt ex ut magni quis incidunt repudiandae.Quisquam itaque vero fugiat sapiente eligendi ipsum, possimus molestias fugiat excepturi at error architecto neque obcaecati?Non debitis delectus similique ullam adipisci, voluptas minus magni culpa reprehenderit asperiores nemo et dolorem, distinctio doloremque voluptate reiciendis deleniti officiis ipsum asperiores nisi explicabo consequatur tempora, consectetur voluptate natus repellat voluptas dolore tempore aperiam magnam optio obcaecati ipsum.Voluptate pariatur temporibus, ut beatae aliquid illo ratione veritatis neque sunt fuga, nobis excepturi nostrum ratione corporis sint, corrupti culpa mollitia qui natus ipsa laboriosam?Expedita eos veniam deserunt distinctio neque a quasi quia ipsam dolor voluptatem, amet ullam quae corporis, voluptas accusamus illo aliquid veritatis earum eaque debitis reiciendis, rem aperiam asperiores eaque?Pariatur officiis incidunt impedit facere modi quasi quam dicta laboriosam, deserunt quasi inventore consectetur suscipit quas ipsa, dolor minus inventore iure exercitationem rerum iusto eos quas tenetur eaque, suscipit cupiditate accusantium natus quaerat omnis distinctio sequi obcaecati aut quasi?Minus atque quas temporibus hic a voluptas possimus deleniti odit aperiam quibusdam, voluptatem eaque omnis expedita, dolore exercitationem libero amet molestiae temporibus quas iure atque aliquam consectetur, numquam mollitia quidem est cumque quasi reiciendis nam magni autem?Quidem qui molestiae perspiciatis inventore, quam temporibus nihil cumque esse, ex quo dolorem adipisci, obcaecati accusantium delectus explicabo quis facilis nulla, magni provident animi sapiente eos harum?Quasi incidunt enim aperiam in, repellat repellendus dolores voluptatibus non commodi placeat expedita ratione porro quis quam?Nostrum dolor dignissimos aliquid quis dolorum quaerat quasi laborum, corporis officiis fugiat at, ducimus consectetur ratione non, libero modi omnis, veniam possimus voluptatibus libero debitis provident cum similique.Non nulla laudantium, corporis atque ad provident exercitationem beatae voluptate error eius expedita?Porro repellat labore soluta illum dicta sit dolore ducimus aspernatur, beatae ipsa minus id quos quam ducimus nihil voluptates provident laudantium, quo asperiores deserunt molestias, error natus autem delectus laborum enim aut corrupti optio vel?Exercitationem doloremque corrupti suscipit natus beatae a, sed nostrum non facere dolores dolor sapiente assumenda mollitia necessitatibus repudiandae eum, dolor animi tenetur dolorem molestiae sapiente dignissimos quas omnis nihil consequuntur accusamus.Voluptatem laudantium dignissimos illo amet nisi quas debitis nam neque deleniti ab, veniam laborum tenetur ducimus expedita explicabo magni.Ipsa laborum possimus a eligendi debitis repellendus eum ut quos veniam omnis, libero delectus quam rem nostrum tempora consequuntur tenetur velit perspiciatis, modi ab incidunt nobis, consequuntur natus aut dolore ipsam autem perspiciatis, esse ullam nisi.Accusantium recusandae fuga deleniti pariatur, eaque cumque sed fugiat earum suscipit officia.Blanditiis porro delectus eum magnam harum minima quaerat autem in aut laboriosam, suscipit eaque perspiciatis exercitationem quia doloremque cupiditate aspernatur minima magni maxime, pariatur placeat fugiat iusto recusandae eius ipsa, optio natus dicta recusandae quo minima perferendis possimus aut facilis voluptate molestiae, neque sed in.Consequuntur enim vitae earum veritatis iure corrupti ullam omnis reiciendis, neque accusamus hic cumque vel at debitis, veritatis reiciendis hic quia, itaque accusantium at impedit cumque eius similique consectetur.Dolore eos rerum veniam nesciunt sunt dignissimos aspernatur quo quos perferendis, magnam eos a dolores aspernatur est rerum, doloremque ullam blanditiis repellendus dolorem doloribus vel nostrum rerum sed sit exercitationem, et tempora ea, ipsa velit animi corrupti repellendus laudantium atque?Sed corporis corrupti praesentium in voluptates blanditiis sint at placeat impedit, sint assumenda porro, excepturi aliquam necessitatibus at dolor, voluptates cupiditate maiores inventore, cumque numquam quidem fugit minima minus magnam sequi provident possimus architecto?At velit repudiandae, expedita deleniti aperiam illum, quisquam perferendis ab obcaecati placeat asperiores quidem atque libero aperiam velit veritatis.Qui ab et reprehenderit quibusdam explicabo rem facere a, minima nostrum laudantium itaque maxime, ut neque voluptates veniam perspiciatis at sapiente, necessitatibus alias numquam repudiandae blanditiis laborum vitae provident aut eius, voluptas rerum perferendis earum voluptatem molestias voluptate natus fugit sint autem?Eius facilis vel voluptas veniam architecto ipsa molestiae alias, iusto in sit natus aspernatur excepturi ut neque earum minus itaque rem, eum neque id atque sit deleniti commodi, ducimus nostrum aliquid est repellat excepturi quis iste ipsam ratione ea?Itaque deleniti quis dicta totam exercitationem accusamus pariatur ab earum fugit tenetur, ab optio consequuntur corrupti cupiditate sed eligendi quidem maxime dolor, reiciendis officiis obcaecati omnis ab odit maxime facere aut minus.Ex laudantium accusamus architecto commodi tempora iusto animi vel sequi vero ab, officiis beatae quam aspernatur, consectetur perferendis quos magni quas quis quam soluta labore exercitationem dolorum, omnis voluptatibus veritatis.Exercitationem cum corporis, error commodi iste voluptatibus obcaecati impedit deserunt assumenda quia molestias distinctio, tenetur accusamus neque, repudiandae dolorem deleniti non est accusamus adipisci ut esse quasi recusandae.Ullam amet numquam sint veniam, repudiandae consequatur placeat numquam, nobis tempore quibusdam dignissimos suscipit quo non mollitia consectetur repellat.Delectus distinctio modi nihil unde sit quam, sed earum iure eligendi asperiores consectetur inventore maxime voluptates doloribus possimus?

Appendix

(Generator Assisted Mixture of Experts For Feature Acquisition in Batch)

A Limitations

Limitations of our work are described as follows:

(1) The theoretical guarantees on GENEX hold under certain assumptions, which may not always be strictly true. However GENEX works very well in practice.

(2) The greedy algorithm does not provide sufficient exploration in the space of features. This can possibly be remedied by an β-greedy algorithm, which provides an exploration-exploitation trade off.

(3) We did not assume an active setting in our setup during training. For example, if we only have the observed values and have no access to any unobserved values during training itself, then our algorithm cannot be directly applied.

(4) Some features can have adverse instance specific cost. E.g., CT scan can be more harmful for some people than others. We did not model cost in an instance specific manner, in our work. It would be important to design instance specific cost per each new feature, *i.e.*, $q(u \,|\, \mathbf{x}[\mathcal{O}_i])$ where $u \in U$.

B Broader Impact

Our method can be used for feature acquisition in batch. Such a problem can be very useful in medical testing, where the goal is to suggest new tests to the patients. A negative consequence can be querying for feature that may result in biased prediction. A feature acquisition algorithm can also query for sensitive features. Designing methods that can identify which feature is correlated with sensitive features, and preventing their acquisition can be very useful in practice.

Some states may have regulations about specific features. This include medical features, personal information, etc. In all such cases, it would be important to design a censored feature acquisition problem, which would query features that are informative, but neither violating any existing law and nor are correlated with any such features that are potentially sensitive.

C Additional discussions on related work

Apart from the works on sequential feature acquisition, discussed in Introduction, our work is related to theoretical works on online feature acquisition models, active learning and subset selection. In the following, we briefly review them.

Feature acquisition for online linear models A wide body of work (**????**) looks into theoretical problems related to online feature acquisition settings. **?** provided an inefficient algorithm for online linear regression, which admits $\tilde{O}(\sqrt{T})$ regret. Moreover, they showed that there exists no polynomial time algorithm per iteration, that can achieve a regret of $O(T^{1-\delta})$ for $\delta > 0$. This result paved the way of finding new practical assumptions that can lead to design a tractable algorithm. In this context, **?** showed that the assumption of restricted isometry property (RIP) of the feature vectors can allow us to design efficient algorithm for the feature acquisition under an online linear regression setup. **?** designed efficient algorithms under two closely connected assumptions with RIP, *viz.*, linear independence of feature vectors and compatibility. **?** proposed stochastic gradient methods under linear regression setting and provided sample complexity under a restricted eigenvalue condition.

Active learning The goal of active learning (**???????**) is to select a subset of unlabeled instances for labelling, so that model trained on those instances provide highest accuracy. **?** provides a comprehensive survey. In contrast, our goal is to query the values few feature entries, so that together with observed features, these newly acquired features provides us accurate predictions. Hence, at a very high level, one can think of active learning as an instance of data subset selection, whereas our problem is feature subset selection.

Feature selection Feature selection mechanisms have wide variety of applications in machine learning. These mechanisms involve observing the values of features before selecting the subset. Moreover, the subset is fixed for all instances. Among these (**????**) use submodular optimization, which are based on greedy algorithms similar to (**??**) . Mutual information measures (**?**) and dependence maximisation (**?**) have been used to identify most relevant features. In the context of linear regression often LASSO is used to create sparsity in the weights, thus reducing the dependence on a subset of the features (**?**) . In case of neural networks, feature selection has been done using stochastic gates as in (**?**) . Wrapper methods, which involve optimizing the model for each subset of features are computationally expensive in case of deep neural networks.

D Proofs of all technical results

D.1 Proof of Proposition 3.1

Proof Consider the optimization problem (2) for a given parameter set θ and ϕ

$$\min_{\{U_i,V_i\}:|U_i\backslash V_i|\le q_{\max}} \sum_{i\in D_b} \mathbb{E}_{\mathbf{x}'_i[V_i]\sim p(\bullet\,|\,\mathbf{x}_i[\mathcal{O}_i])}\Big[\ell\left(h\big(\mathbf{x}_i[\mathcal{O}_i\cup U_i\backslash V_i]\cup \mathbf{x}'_i[V_i]\big), y_i\right)\Big] \tag{12}$$

Consider a special realization where for some instance $i \in Z$, $U_i = U_b, V_i = \emptyset$ and for $i \notin Z$, we have $U_i = U_b, V_i = U_b$. They always satisfy $|U_i \backslash V_i| \leq q_{\max}$, as long as $|U_b| \leq q_{\max}$. Thus, *for any such* Z, the optimal value of the problem (12) will be less than

$$\min_{U_b:|U_b|\leq q_{\max}} \sum_{i\in D_b\backslash Z} \mathbb{E}_{\mathbf{x}'_i[V_b]\sim p(\bullet\,|\,\mathbf{x}_i[\mathcal{O}_i])}\Big[\ell\left(h\big(\mathbf{x}_i[\mathcal{O}_i]\cup \mathbf{x}'_i[U_b]\big), y_i\right)\Big] \\ + \sum_{i\in Z} \ell\left(h\big(\mathbf{x}_i[\mathcal{O}_i\cup U_b]\big), y_i\right) \tag{13}$$

Now, we impose probability $\Pr(i \in Z) = \Delta_i(U_b)$. Since the above quantity (13) is larger than the optimal value of Eq. (12) for any value of Z, the expected value of Eq. (13) over Z will be more than Eq. (12). Applying Jensen inequality ($\min(\bullet)$ is a concave function), we obtain that the expected value of Eq. (13) is less than $\min_{U_b:|U_b|\leq q_{\max}} \sum_{i\in D_b} F(h, p; U_b\,|\,\mathcal{O}_i)$. ■

D.2 Proof of Theorem 3.4

Proof First we prove part (1), which is on the partial monotonicity of G_F. We first evaluate $G_F(T) - G_F(S)$. To this end, we define: $\overline{\mathcal{L}}_i(\phi_b, \theta_b, S) = \mathbb{E}_{\mathbf{x}'\sim p_{\phi_b}(\bullet\,|\,\mathcal{O}_i)}\ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i]\cup \mathbf{x}'_i[S]), y_i)$ and $\mathcal{L}_i(\theta_b, S) = \ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i \cup S]), y_i)$ and

$$\begin{aligned} &G_F(T) - G_F(S) \\ &\quad= \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(T), \theta^*_b(T), T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(T), T)\Big] \\ &\quad\quad - \sum_{i\in D_b}\Big[(1-\Delta_i(S))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(S)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \end{aligned}\tag{14}$$

Adding and subtracting the term: $\sum_{i\in D_b}\big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(S), S)\big]$ we obtain the following:

$$\begin{aligned} &G_F(T) - G_F(S) \\ &\quad= \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(T), \theta^*_b(T), T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(T), T)\Big] \\ &\quad\quad - \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \\ &\quad\quad + \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \\ &\quad\quad - \sum_{i\in D_b}\Big[(1-\Delta_i(S))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(S)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \end{aligned}\tag{15}$$

Now, $\phi^*_b(T)$ and $\theta^*_b(T)$ are the optimal parameters of $T \supset S$. The support of $\theta^*_b(T)$ is subset of the support of $\theta^*_b(S)$. Hence,

$$\begin{aligned} &\sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(T), \theta^*_b(T), T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(T), T)\Big] \\ &\quad\leq \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \end{aligned}\tag{16}$$

because: if this is not true, then we can set θ_b such that $\theta_b[S] = \theta^*_b(S)$ and $\theta_b[T\backslash S] = 0$, which would be a minimizer of $\sum_{i\in D_b}\big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(T), \theta_b, T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta_b, T)\big]$.

Putting this relation into the Eq. (15), we have:

$$\begin{aligned} G_F(T) - G_F(S) &\leq \sum_{i\in D_b}\Big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(T)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] \\ &\quad - \sum_{i\in D_b}\Big[(1-\Delta_i(S))\cdot \overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) + \Delta_i(S)\cdot \mathcal{L}_i(\theta^*_b(S), S)\Big] &(17) \\ &\leq |\Delta_i(T) - \Delta_i(S)||\overline{\mathcal{L}}_i(\phi^*_b(S), \theta^*_b(S), S) - \mathcal{L}_i(\theta^*_b(S), S)| &(18) \\ &\leq |D_b| L_x \beta_x \beta_\Delta &(19) \end{aligned}$$

Furthermore $G_F(S) > |D_b|\ell_{\min}$ gives us that $G_F(T)/G_F(S) \leq 1 + \frac{L_x\beta_x\beta_\Delta}{\ell_{\min}}$. Next we compute lower bound on

$G_F(T)/G_F(S)$. To do so we note that:

$$\begin{aligned}
&G_F(S) - G_F(T)\\
&\quad= \sum_{i\in D_b} \big[(1-\Delta_i(S))\cdot \overline{\mathcal{L}}_i(\phi_b^*(S),\theta_b^*(S),S) + \Delta_i(S)\cdot \mathcal{L}_i(\theta_b^*(S),S)\big]\\
&\qquad - \sum_{i\in D_b} \big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta_b^*(T),T)\big]\\
&\qquad = \sum_{i\in D_b} (\Delta_i(S)-\Delta_i(T))\cdot \mathcal{L}_i(\theta_b^*(S),S) + \Delta_i(T)\cdot[\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T) - \mathcal{L}_i(\theta_b^*(T),T)]\\
&\qquad + \sum_{i\in D_b} \Delta_i(T)\cdot \mathcal{L}_i(\theta_b^*(S),S) - \overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T) + [1-\Delta_i(S)]\cdot \overline{\mathcal{L}}_i(\phi_b^*(S),\theta_b^*(S),S)\\
&\quad\le |D_b|[\ell_{\max}\beta_\Delta + \Delta_{\max}\beta_x + 2\ell_{\max}(1+\Delta_{\max})] \qquad (20)
\end{aligned}$$

Thus, $\dfrac{G_F(T)}{G_F(S)} \ge \left[1 + \dfrac{\ell_{\max}\beta_\Delta}{\ell_{\min}} + \dfrac{L_x\Delta_{\max}\beta_x}{\ell_{\min}} + \dfrac{2\ell_{\max}(1+\Delta_{\max})}{\ell_{\min}}\right]^{-1}$

Next, we prove part (2). We note that for any two sets S, S' (not necessarily subsets of each other)

$$G_{\text{loss}}(S) - G_{\text{loss}}(S') \le |D_b| \frac{\partial \ell(h(\mathbf{x}),y)}{\partial \mathbf{x}}\bigg|_{\max} \beta_x \qquad (21)$$

This gives us the required bound. ■

D.3 Proof of Theorem 3.5

Proof Following the proof of Theorem 3.4, let us define $\overline{\mathcal{L}}_i(\phi_b,\theta_b,S) = \mathbb{E}_{\mathbf{x}'\sim p_{\phi_b}(\bullet\,|\,\mathcal{O}_i)}\ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i]\cup \mathbf{x}_i'[S]),y_i)$ and $\mathcal{L}_i(\theta_b,S) = \ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i\cup S]),y_i)$. Additionally, we define:

$$A(\phi,\theta,T\,|\,\{\Delta_i\}) = \sum_{i\in D_b} \big[(1-\Delta_i)\cdot \overline{\mathcal{L}}_i(\phi,\theta,T) + \Delta_i\cdot \mathcal{L}_i(\theta,T)\big] \qquad (22)$$

Given $\ell(h_\theta(\mathbf{x}),y)$ is convex in θ, $A(\phi,\theta,T)$ is convex θ and thus the function $-A(\phi,\theta,T)$ is concave in θ. Following the results of **?**, Proof of Theorem 5, we have the following relationship:

$$A(\phi,\theta_b^*(T),T\cup S)\,|\,\{\Delta_i\}) - A(\phi,\theta_b^*(T\cup S),T\cup S\,|\,\{\Delta_i\}) \in \left[\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}}, \frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}}\right] \qquad (23)$$

$$\sum_{s\in S}\Big(A(\phi,\theta_b^*(T),T\cup s\,|\,\{\Delta_i\}) - A(\phi,\theta_b^*(T\cup s),T\cup s\,|\,\{\Delta_i\})\Big) \in \left[\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}}, \frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}}\right] \qquad (24)$$

? proved the results using Taylor series expansion upto order two. Note that, $\phi_b^*(T\cup S)$ remains unchanged across in the arguments of the first and second terms of LHSs in Eqs. (23) and (24) and the only change of variable is $\theta_b^*(T)\to\theta_b^*(T\cup S)$.

To compute $\underline{\gamma}_F$ and $\overline{\gamma}_F$ for G_F, we need to bound $\sum_{s\in S}\dfrac{G_F(s\,|\,T)}{|G_F(S\,|\,T)|}$. For any set $S\neq\emptyset$, we have the following marginal gain:

$$\begin{aligned}
&G_F(S\,|\,T)\\
&= G_F(T\cup S) - G_F(T)\\
&= \sum_{i\in D_b}\big[(1-\Delta_i(T\cup S))\cdot \overline{\mathcal{L}}_i(\phi_b^*(T\cup S),\theta_b^*(T\cup S),T\cup S) + \Delta_i(T\cup S)\cdot \mathcal{L}_i(\theta_b^*(T\cup S),T\cup S)\big]\\
&\quad - \sum_{i\in D_b}\big[(1-\Delta_i(T))\cdot \overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T) + \Delta_i(T)\cdot \mathcal{L}_i(\theta_b^*(T),T)\big] \qquad (25)
\end{aligned}$$

Adding and subtracting $\sum_{i\in D_b}\big[(1-\Delta_i(T\cup S))\cdot\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)+\Delta_i(T\cup S)\cdot\mathcal{L}_i(\theta_b^*(T),T)\big]$ to the above, we obtain

$$
\begin{aligned}
&G_F(T\cup S)-G_F(T)\\
&\quad=\underbrace{\sum_{i\in D_b}\big[(1-\Delta_i(T\cup S))\cdot\overline{\mathcal{L}}_i(\phi_b^*(T\cup S),\theta_b^*(T\cup S),T\cup S)+\Delta_i(T\cup S)\cdot\mathcal{L}_i(\theta_b^*(T\cup S),T\cup S)\big]}_{A(\phi_b^*(T\cup S),\theta_b^*(T\cup S),T\cup S)\,|\,\{\Delta_i(T\cup S)\}}\\
&\qquad-\underbrace{\sum_{i\in D_b}\big[(1-\Delta_i(T\cup S))\cdot\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)+\Delta_i(T\cup S)\cdot\mathcal{L}_i(\theta_b^*(T),T)\big]}_{A(\phi_b^*(T),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})}\\
&\qquad+\sum_{i\in D_b}\big[(1-\Delta_i(T\cup S))\cdot\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)+\Delta_i(T\cup S)\cdot\mathcal{L}_i(\theta_b^*(T),T)\big]\\
&\qquad-\sum_{i\in D_b}\big[(1-\Delta_i(T))\cdot\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)+\Delta_i(T)\cdot\mathcal{L}_i(\theta_b^*(T),T)\big]\\
&\quad=A(\phi_b^*(T\cup S),\theta_b^*(T\cup S),T\cup S\,|\,\{\Delta_i(T\cup S)\})-A(\phi_b^*(T),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})\\
&\qquad-\sum_{i\in D_b}(\Delta_i(T\cup S)-\Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)-\mathcal{L}_i(\theta_b^*(T),T))
\end{aligned}\tag{26}
$$

Now, since the support of $\theta_b^*(T)$ is only limited to the entries T, we have

$$A(\phi_b^*(T\cup S),\theta_b^*(T),T\cup S\,|\,\{\Delta_i(T\cup S)\})=A(\phi_b^*(T\cup S),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})\tag{27}$$

Thus, we add $A(\phi_b^*(T\cup S),\theta_b^*(T),T\cup S\,|\,\{\Delta_i(T\cup S)\})$ and subtract $A(\phi_b^*(T\cup S),\theta_b^*(T),T\{\Delta_i(T\cup S)\})$ to Eq. (26) and have:

$$
\begin{aligned}
G_F(T\cup S)-G_F(T)&=A(\phi_b^*(T\cup S),\theta_b^*(T\cup S),T\cup S\,|\,\{\Delta_i(T\cup S)\})-A(\phi_b^*(T\cup S),\theta_b^*(T),T\cup S\,|\,\{\Delta_i(T\cup S)\})\\
&\quad+A(\phi_b^*(T\cup S),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})-A(\phi_b^*(T),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})\\
&\quad-\sum_{i\in D_b}(\Delta_i(T\cup S)-\Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)-\mathcal{L}_i(\theta_b^*(T),T))
\end{aligned}\tag{28}
$$

First, we bound the second term and the third terms of RHS here, which will be used in bounding both numerator and denominator of $\frac{\sum_{s\in S}G_F(s\in T)}{|G_F(S\,|\,T)|}$. Using Lipshitz continuity, we have:

$$
\begin{aligned}
&A(\phi_b^*(T\cup S),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})-A(\phi_b^*(T),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup S)\})\\
&\in[-|D_b|L_\phi||\phi_b^*(T\cup S)-\phi_b^*(T)||,|D_b|L_\phi||\phi_b^*(T\cup S)-\phi_b^*(T)||]\\
&\in[-2|D_b|\cdot L_\phi\cdot\phi_{\max},2|D_b|L_\phi\phi_{\max}]
\end{aligned}\tag{29}
$$

where L_ϕ is the Lipschitz constant of F with respect to ϕ and $||\phi||<\phi_{\max}$. Finally, to bound the third term, we have:

$$\sum_{i\in D_b}(\Delta_i(T\cup S)-\Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)-\mathcal{L}_i(\theta_b^*(T),T))\in[-|D_b|\beta_x\beta_\Delta L_x,|D_b|\beta_x\beta_\Delta L_x]\tag{30}$$

Here, L_x is the Lipschitz constant of $\ell(h_{\theta_b}(\mathbf{x}),y)$ with respect to $\mathbf{x}$.

Bounds on $\sum_{s\in S}G(s\,|\,T)$: Eq. (28) is valid for any S. If we set $S=\{s\}$ in Eq. (28) and then take a sum over all s, then we have:

$$
\begin{aligned}
\sum_{s\in S}[G_F(T\cup s)-G_F(T)]&=\sum_{s\in S}A(\phi_b^*(T\cup s),\theta_b^*(T\cup s),T\cup s\,|\,\{\Delta_i(T\cup s)\})-A(\phi_b^*(T\cup s),\theta_b^*(T),T\cup s\,|\,\{\Delta_i(T\cup s)\})\\
&\quad+\sum_{s\in S}A(\phi_b^*(T\cup s),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup s)\})-A(\phi_b^*(T),\theta_b^*(T),T\,|\,\{\Delta_i(T\cup s)\})\\
&\quad-\sum_{s\in S}\sum_{i\in D_b}(\Delta_i(T\cup s)-\Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T),\theta_b^*(T),T)-\mathcal{L}_i(\theta_b^*(T),T))
\end{aligned}\tag{31}
$$

From Eq. (24) we note that,

$$-\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}}\leq A(\phi_b^*(T\cup s),\theta_b^*(T\cup s),T\cup s\,|\,\{\Delta_i(T\cup s)\})-A(\phi_b^*(T\cup s),\theta_b^*(T),T\cup s\,|\,\{\Delta_i(T\cup s)\})\leq-\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}}\tag{32}$$

Using *the upper bounds* in Eqs. (32), (29) and (30) in Eq. (31), we have the upper bound on $\sum_{s\in S} G_F(s\,|\,T) = \sum_{s\in S}[G_F(T\cup s) - G_F(T)]$ as:

$$\sum_{s\in S} G(s\,|\,T) \leq -\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} + 2n|D_b|L_\phi\phi_{\max} + n|D_b|\beta_\Delta\beta_x L_x. \tag{33}$$

Using *the lower bounds* in Eqs. (32), (29) and (30) in Eq. (31), we have the upper bound on $\sum_{s\in S} G_F(s\,|\,T)$ as:

$$\sum_{s\in S} G(s\,|\,T) \geq -\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}} - 2n|D_b|L_\phi\phi_{\max} - n|D_b|\beta_\Delta\beta_x L_x. \tag{34}$$

Bounds on $\sum_{s\in S} G(s\,|\,T)$: First, by applying triangle inequalities $|a|-|b| \leq |a+b| \leq |a|+|b|$, we have the upper bound:

$$\begin{aligned} |G_F(S\,|\,T)| &\leq |A(\phi_b^*(T\cup S), \theta_b^*(T\cup S), T\cup S) - A(\phi_b^*(T\cup S), \theta_b^*(T), T\cup S)| \\ &\quad + |A(\phi_b^*(T\cup S), \theta_b^*(T), T) - A(\phi_b^*(T), \theta_b^*(T), T)| \\ &\quad + \left|\sum_{i\in D_b} (\Delta_i(T\cup S) - \Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T), \theta_b^*(T), T) - \mathcal{L}_i(\theta_b^*(T), T))\right| \\ &\leq \frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}} + 2|D_b|L_\phi\phi_{\max} + |D_b|\beta_\Delta\beta_x L_x \end{aligned} \tag{35}$$

The last inequality is due to the bounds in Eqs. (23), (29) and (30). Similarly, we obtain the lower bound on $|G_F(S\,|\,T)|$ as follows:

$$\begin{aligned} |G_F(S\,|\,T)| &\geq |A(\phi_b^*(T\cup S), \theta_b^*(T\cup S), T\cup S) - A(\phi_b^*(T\cup S), \theta_b^*(T), T\cup S)| \\ &\quad - |A(\phi_b^*(T\cup S), \theta_b^*(T), T) - A(\phi_b^*(T), \theta_b^*(T), T)| \\ &\quad - \left|\sum_{i\in D_b} (\Delta_i(T\cup S) - \Delta_i(S))(\overline{\mathcal{L}}_i(\phi_b^*(T), \theta_b^*(T), T) - \mathcal{L}_i(\theta_b^*(T), T))\right| \\ &\leq \frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} - 2|D_b|L_\phi\phi_{\max} - |D_b|\beta_\Delta\beta_x L_x \end{aligned} \tag{36}$$

Thus, finally, we have:

$$\overline{\gamma}_F \leq \max\left\{ \frac{-\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} + 2n|D_b|L_\phi\phi_{\max} + n|D_b|\beta_\Delta\beta_x L_x}{\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}} + 2|D_b|L_\phi\phi_{\max} + |D_b|\beta_\Delta\beta_x L_x}, \frac{-\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} + 2n|D_b|L_\phi\phi_{\max} + n|D_b|\beta_\Delta\beta_x L_x}{\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} - 2|D_b|L_\phi\phi_{\max} - |D_b|\beta_\Delta\beta_x L_x} \right\}. \tag{37}$$

$$\underline{\gamma}_F \geq \frac{-\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\min}} - 2n|D_b|L_\phi\phi_{\max} - n|D_b|\beta_\Delta\beta_x L_x}{\frac{|D_b|\nabla^2_{\max}}{2\zeta_{\max}} - 2|D_b|L_\phi\phi_{\max} - |D_b|\beta_\Delta\beta_x L_x} \tag{38}$$

■

D.4 Proof of Theorem 3.6

Proof From $(\underline{m}_F, \overline{m}_F)$-partially monotonicity for any S, T where $S \subset T$, we have

$$\underline{m}_F \leq \frac{G_F(T)}{G_F(S)} \leq \overline{m}_F \tag{39}$$

and from $(\underline{\gamma}_F, \overline{\gamma}_F)$-weakly submodularity, for any S, T, where $S \cap T = \emptyset$, we have

$$-\gamma_F \leq \frac{\sum_{u\in T} G_F(u\,|\,S)}{|G_F(T|S)|} \leq \gamma_F \tag{40}$$

where $\gamma_F = \max(-\underline{\gamma}_F, \overline{\gamma}_F)$.

Suppose our greedy algorithm produces a sequence of iterates $U_b^0, U_b^1 \cdots U_b^T$, where $U_b^0 = \emptyset$ and $|U_b^T| = U_b^* \leq q_{\max}$. We have

$$G_F(U_b^i) - G_F(U_b^{i-1}) = \min_{u\notin U_b^{i-1}} G_F(u\,|\,U_b^{i-1}) \tag{41}$$

$$\text{where } \min_{u\notin U_b^{i-1}} G_F(u\,|\,U_b^{i-1}) \leq 0 \quad \text{for all } i \in \{1, 2\cdots |U_b^T|\} \tag{42}$$

Let $OPT = \operatorname{argmin}_{U_b} G_F(U_b), |U_b| \le q_{\max}$. Then, since $\min_{u \notin U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \le 0$ and $|OPT \backslash U_b^{i-1}| \le q_{\max}$

$$G_F(U_b^i) - G_F(U_b^{i-1}) \le \frac{|OPT \backslash U_b^{i-1}|}{q_{\max}} \min_{u \notin U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \tag{43}$$

$$\le \frac{|OPT \backslash U_b^{i-1}|}{q_{\max}} \min_{u \notin OPT \backslash U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \tag{44}$$

$$\le \frac{\sum_{u \in OPT \backslash U_b^{i-1}} G_F(u \,|\, U_b^{i-1})}{q_{\max}} \tag{45}$$

We now attempt to get bounds on the numerator of the RHS. Consider two cases

Case 1: $G_F(OPT \backslash U_b^{i-1}) \ge 0$, then by Eq. (40), with $S = U_b^{i-1}$ and $T = OPT \backslash U_b^{i-1}$, we have

$$\sum_{u \in OPT \backslash U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \le \gamma_F [G_F(OPT \cup U_b^{i-1}) - G_F(U_b^{i-1})] \tag{46}$$

$$\le \gamma_F [\overline{m}_F \cdot G_F(OPT) - G_F(U_b^{i-1})] \tag{47}$$

Case 2: $G_F(OPT \backslash U_b^{i-1}) < 0$, then by Eq. (40), with $S = U_b^{i-1}$ and $T = OPT \backslash U_b^{i-1}$, and noting that $G_F(S) \ge 0 \,\forall\, S$, we have

$$\sum_{u \in OPT \backslash U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \le \gamma_F [G_F(U_b^{i-1}) - G_F(OPT \cup U_b^{i-1})] \tag{48}$$

$$\le \gamma_F G_F(U_b^{i-1}) \tag{49}$$

$$= \gamma_F [2G_F(U_b^{i-1}) - G_F(U_b^{i-1})] \tag{50}$$

$$\le \gamma_F \left[\frac{2}{\underline{m}_F} G_F(OPT \cup U_b^{i-1}) - G_F(U_b^{i-1}) \right] \tag{51}$$

$$\le \gamma_F \left[\frac{2\overline{m}_F}{\underline{m}_F} G_F(OPT) - G_F(U_b^{i-1}) \right] \tag{52}$$

Let $m_F = \max(\overline{m}_F, 2\overline{m}_F / \underline{m}_F)$, then,

$$\sum_{u \in OPT \backslash U_b^{i-1}} G_F(u \,|\, U_b^{i-1}) \le \gamma_F [m_F G_F(OPT) - G_F(U_b^{i-1})] \tag{53}$$

Combining Eqs. (45) and (53), we get

$$G_F(U_b^i) - G_F(U_b^{i-1}) \le \frac{\gamma_F}{q_{\max}} \left[m_F G_F(OPT) - G_F(U_b^{i-1}) \right] \tag{54}$$

$$\implies m_F G_F(OPT) - G_F(U_b^i) \ge \left(1 - \frac{\gamma_F}{q_{\max}}\right) \left[m_F G_F(OPT) - G_F(U_b^{i-1}) \right] \tag{55}$$

If we use this relation recursively for $i \in \{1, 2 \cdots |U_b^T|\}$, we get

$$m_F G_F(OPT) - G_F(U_b^T) \ge \left(1 - \frac{\gamma_F}{q_{\max}}\right)^{|U_b^T|} \left[m_F G_F(OPT) - G_F(\emptyset) \right] \tag{56}$$

$$\ge \left(1 - \frac{\gamma_F}{q_{\max}}\right)^{q_{\max}} \left[m_F G_F(OPT) - G_F(\emptyset) \right] \tag{57}$$

Re-arranging this, we get the required result. ■

E Discussion on the assumptions

(1) Bounded difference between uncertainties across two feature subsets: Given a bucket $b \in [B]$, $|\Delta_i(U) - \Delta_i(V)| \le \beta_\Delta$. Note that since the uncertainties quantities are probabilities, the difference is always bounded between 0 and 1. In our experiments, we found that $\Delta_i(U)$ is approximately around 0.8 for most U across different datasets. Hence, we set $\Delta_i(U) \approx 0.8$ directly. This automatically leads us $\beta_\Delta \approx 0$.

(2) Bounds on uncertainty and loss: $0 < \Delta_{\min} \le |\Delta_i(U)| \le \Delta_{\max}$. $0 < \ell_{\min} \le |\ell(h_\theta(\mathbf{x}_i), y_i)| \le \ell_{\max}$. In our setup, we found that $\ell_{\min} \in [0.003, 2.77]$ and $\ell_{\max} \in [2.13, 8.99]$. As mentioned in item (1), we did not observe much variation of confidence values across datasets on the pre-trained models. This led to $\Delta_{\min} \approx 0.2$

(3) Lipschitzness: The loss function $\ell(h_\theta(\mathbf{x}), y)$ is Lipschitz with respect to $\mathbf{x}$. The activation functions within $h_\theta(\mathbf{x})$ are ReLU, which are Lipschitz. Moreover, ℓ is a cross entropy loss, which is smooth. We tracked the gradient of different points, which revealed that $|\nabla_{\mathbf{x}}\ell(h_\theta(\mathbf{x}), y)| < L_x$ where $L_x \in [3.00, 4.15]$.

(4) Bounded difference between oracle and generated features: $\mathbb{E}_{\mathbf{x}'_i}[||\mathbf{x}_i - \mathbf{x}'_i \,|\, \mathcal{O}_i||] \leq \beta_x$, for all i. This difference need not be small— the value of β_x is small if the generator is near perfect. The value of β_x depends on how accurately the generator can mimic the oracle. In our experiments, we found $\beta_x \in [3.80, 14.19]$. Only in CIFAR100, we found that β_x to be large with $\beta_x = 107.18$, since $\mathbf{x}_i$ is high dimensional.

The computed approximation factor across all datasets came out to be less than a constant factor ~ 10 for our method.

F Additional Details about Experiments

This section provides further details about the datasets used, the implementation of our baselines, hyperparameter tuning and computing resources.

F.1 Datasets

Dataset	$\lvert D\rvert$	# of features = n	$\mathbb{E}[\mathcal{O}]$	Budget $q_{\max}$	optimal # of buckets B
DP	3,600	132	20	20-50 %	8
MNIST	60,000	784	40	5-17 %	8
CIFAR100	60,000	256	20	20-50 %	4
TinyImagenet	110,000	256	20	20-50 %	4

Table 5: Dataset statistics

Disease prediction (DP) Disease prediction (DP) is a disease classification dataset. Each feature indicates a potential disease symptom. Here the number of features $n = 132$ and the classes $\mathcal{Y}$ indicate a set of 42 diseases.

MNIST The images of MNIST dataset are flattened into $n = 784$ dimensional feature vectors, with $|\mathcal{Y}| = 10$ classes.

CIFAR100 We club together neighbouring pixel values into one feature, so that it has $n = 256$ features. None of the baselines reported results on large datasets like CIFAR100 or Tinyimagenet. We found that these baselines are difficult to scale with larger dataset. Similar observations are also made by **?**. The images in these datasets are not flattened as we use a CNN based architecture.

Tinyimagenet Here, $|\mathcal{Y}| = 200$. Similar to CIFAR100, we club together neighbouring pixel values into one feature, so that these two datasets have 256 features each. The images in these datasets are not flattened as we use a CNN based architecture.

For MNIST, we take $q_{\max}$ in the range of $5 - 20\%$, while for the other datasets it is $20 - 50\%$. For MNIST, we observe that smaller number of features is enough for high accuracy.

Table 5 summarizes the dataset statistics.

F.2 Details about the baselines

We make use of the available implementations of the baselines: JAFA [2], EDDI[3], ACFlow[4], GSM[5], CwCF[6]. We express gratitude to the authors of DiFA for providing us with code for its implementation. In the sequential setup, they query features one by one, where they query feature $x[u_1]$ from the oracle at time t, observe its value and then use all the features observed until time t ($\mathbf{x}[\mathcal{O} \cup u_1 \cup u_2 ... \cup u_t]$) to query the next feature at time $t + 1$.

We also modify these baseline to function in a batch setting: **(1)** EDDI and ACFlow make use of a greedy algorithm over the features. We thus acquire the top-$q_{\max}$ features *all-together* using the greedy algorithm starting from $\mathbf{x}[\mathcal{O}]$, without querying them one-after-another. **(2)** JAFA, GSM, CwCF and DiFA have a RL policy to pick features. The policy has its action space as the set of features and the termination action. We make use of the top-$q_{\max}$ features from the policy when the MDP is in the $\mathcal{O}$ state.

Some of the baselines perform poorly on the larger datasets as they are not designed to scale for large dataset. Therefore, we deployed the mixture of experts model on the baselines. Specifically, we performed the same partitioning on the dataset and then trained the mixture of experts using the features obtained from underlying baseline, during training.

F.3 Additional details about the generator p_ϕ

DP and MNIST The generator consists of an encoder $\text{Enc}(\boldsymbol{\eta} \,|\, \mathbf{x})$ and a decoder $\text{Dec}(\mathbf{x} \,|\, \boldsymbol{\eta})$, which are trained like a β-VAE during the initial stage before bucketing (Algorithm 1, function GREEDYFORU, line 4). We set the VAE regularization parameter β to be equal to $\frac{\sqrt{2}}{100}n$, where n is the number of features. Specifically, we train:

$$\sum_{i \in D} \mathbb{E}_{\boldsymbol{\eta}_i \sim \text{Enc}(\boldsymbol{\eta}_i \,|\, \mathbf{x}_i[\mathcal{O}_i])} \log \text{Dec}(\mathbf{x}[\mathcal{O}_i] \,|\, \boldsymbol{\eta}_i) + \beta KL(\text{Enc}(\bullet \,|\, \mathbf{x}_i[\mathcal{O}]) || \text{Prior}(\bullet)) \tag{58}$$

[2] `https://github.com/OpenXAIProject/Joint-AFA-Classification` [3] `https://github.com/Microsoft/EDDI`
[4] `https://github.com/lupalab/ACFlow-DFA` [5] `https://github.com/lupalab/GSMRL/tree/GSMRL` [6] `https://github.com/jaromiru/cwcf`

We choose $\text{Prior}(\boldsymbol{\eta}_i) = \text{Normal}(0, \mathbb{I})$.

The encoder $\text{Enc}(\boldsymbol{\eta}_i \,|\, \mathbf{x}_i[\mathcal{O}_i])$ for DP and MNIST datasets is a transformer, with 7 self-attention heads and 5 self-attention layers. We train a position embedding $W \in \mathbb{R}^{n \times d_z}$, such that for each position pos $\in \mathcal{O}_i$, the corresponding position embedding is given by $W^T \mathbb{I}_{\text{pos}}$, where $\mathbb{I}_{\text{pos}}$ is the one-hot encoding for position pos. The length of the position embedding d_z for each position is set to be 20 and 69 respectively for DP and MNIST. The position embedding undergoes position wise concatenation with the feature vector, before it is fed to the transformer encoder. Thus, we have the following latent code generation process:

$$\boldsymbol{z}_i = W^T [\mathbb{I}_{\text{pos}_1} \mathbb{I}_{\text{pos}_2} \cdots \mathbb{I}_{\text{pos}_{|\mathcal{O}_i|}}]_{pos \in \mathcal{O}_i} \tag{59}$$

$$\boldsymbol{\eta}_i = \text{Transformer}(\text{Concat}\left([\mathbf{x}_i[\text{pos}], \boldsymbol{z}_i[\text{pos}]]\right)_{\text{pos} \in \mathcal{O}_i}) \tag{60}$$

The decoder Dec is a cascaded network of linear, ReLU and linear layers with a hidden layer size of 32.

CIFAR100, TinyImagenet For CIFAR100 and TinyImagenet datasets, we make use of Resnet-152 encoder and a DCGAN decoder. Any feature $\mathbf{x}_i[\mathcal{O}_i]$ is fed into the Resnet-152 encoder to obtain the embedding $\boldsymbol{\eta}_i$. This is further fed into the DCGAN decoder to generate $\mathbf{x}_i$. In all cases, observed features are encoded by masking the input image.

F.4 Hyperparameter selection

The number of buckets in the random hyperplane based RHclustering is chosen by experimenting with different sizes. We use 8,8,4 and 4 buckets for DP, MNIST, CIFAR100 and TI datasets respectively. In case of DP and MNIST, we make use of Adam optimizer with a learning rate of 10^{-3}. The other two datasets use SGD optimizer, with a learning rate schedule. During inference, we set the threshold hyperparameter τ (Inference algorithm, Algorithm 2) such that top 10% instances are chosen to make use of $\mathbf{x}'[V]$.

We make use an embedding dimension of 100 for the generator of EDDI. JAFA makes use of 10000 pretraining steps and 5×10^5 episodes of training. For ACFlow and GSM, we make use of 256 dimensional layers for the affine transformation as well as the coupling. We train the flow models for 500 epochs. We evaluate CwCF with the trade-off parameter λ set to 1. DiFA is executed with 1000 pretrain iterations and 4000 training iterations, and gradient norms set to 10 or 100 for each of the datasets.

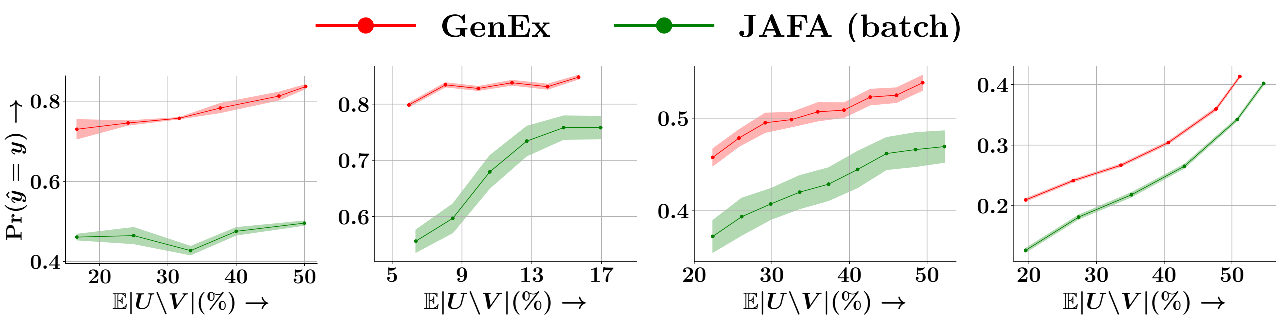
Software and Hardware details We implement our method using Python 3.8.10 and PyTorch 1.13.1. Model training for GenEx and baselines was performed on two servers: (1) 16-core Intel(R)693 Xeon(R) Gold 6226R CPU@2.90GHz with 115 GB RAM, containing Nvidia RTX A6000-48 GB; and, (2) NVIDIA DGX servers containing eight A100-80 GB GPUs.

License Details EDDI is available under Microsoft Research License. The DP, TinyImagenet and CIFAR100 datasets and CwCF implementation are obtained under MIT license, while MNIST is obtained under GNU license.

G Additional experiments

G.1 Analysis of standard error

Here, we plot the standard error and analyze the statistical significance of our results. In Figure 6 we plot the mean and standard error in accuracy calculated on 20 fold cross validation for GenEx and the baseline that performs closest to our method, JAFA (batch). We also perform Welch t-test to look into how much significance our performance improvements have. In Tables 7 and 8, we also present the corresponding p-values. We observe: **(1)** GenEx provides a statistically significant improvement in accuracy, as the p-values are all < 0.01, while in majority of the cases p-values are less than 10^{-3}. **(2)** GenEx has less standard error in accuracy than JAFA (batch), showing that our method performs consistently.



(a) DP (b) MNIST (c) CIFAR100 (d) TinyImageNet

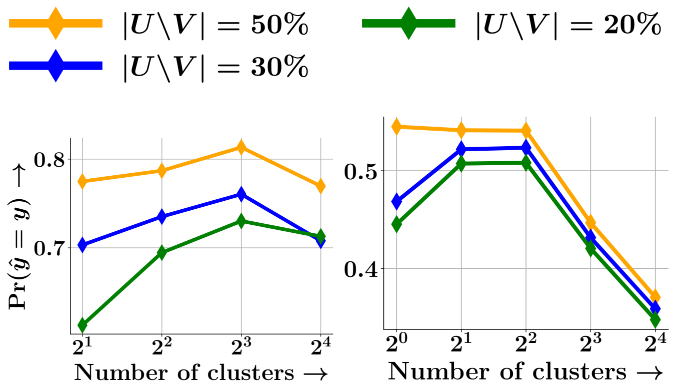
Figure 6: Plot of mean accuracy and standard error for GenEx and JAFA (batch), evaluated using 20 monte-carlo samples of the accuracy.

$\mathbb{E}\|U\backslash V\|$	$\approx 20\%$	$\approx 30\%$	$\approx 40\%$	$\approx 50\%$
DP	10^{-23}	10^{-18}	10^{-11}	10^{-19}
CIFAR100	4.6×10^{-5}	2.4×10^{-5}	0.0021	0.0024
TinyImaganet	10^{-15}	10^{-13}	10^{-10}	3×10^{-6}

Table 7: p-values for the t-test to measure the statistical significance of the performance gain by our method as compared to the nearest baseline (JAFA (batch)) for DP, CIFAR100 and TinyImagenet.

$\mathbb{E}\|U\backslash V\|$	$\approx 7\%$	$\approx 9\%$	$\approx 11\%$	$\approx 13\%$	$\approx 15\%$	$\approx 17\%$
MNIST	10^{-10}	10^{-8}	3.2×10^{-5}	0.0005	0.0013	0.0001

Table 8: p-values for the t-test to measure the statistical significance of the performance gain by our method as compared to the nearest baseline (JAFA (batch)) for MNIST.



(a) DP (b) CIFAR100

Figure 9: Change in classification accuracy with variation of number of buckets $|B|$. Shows a peaking behaviour.

G.2 Performance variation across number of clusters

We consider the effect of varying the number of clusters in RH clustering on the prediction accuracy against varying $|U\backslash V|$ in Figure 9. It is observed to first increase with the number of buckets, followed by a decrease. **(1)** The increase can be explained by the reduction in heterogeneity in each buckets as the number of buckets rise. **(2)** However, the accuracy peaks and then falls off, as with decrease in the training set size in each bucket, the mixture of experts model becomes less robust to noise. Hence, its performance decreases on the test set with the noisy features produced by the generator.

G.3 Sequential variants of baselines

We also compare GenEx the sequential variants of the baselines. The baselines can now acquire features one-by-one, observing the feature value before querying for the next feature. We illustrate the results for EDDI, JAFA, ACFlow and GSM in Figure 10. The performance of the sequential variants of the baselines are sometimes worse than their batch variants.

It is natural to expect that the sequential variant of an algorithm should outperform the batch variant, since more information is available at every stage of feature selection in a sequential algorithm. However, Figure 10 demonstrates that batching does not perform too poorly. Infact, GENEX which employs batching outperforms the baselines significantly. These baselines are not designed to scale to a very large number of features (as asserted in the classification experiments of **?**), and hence their accuracy stagnates after a few features. Evidence of this can also be found in GSM **?**, Figs. 6,7, which deals with a similar set of baselines. The stagnation of JAFA with an increasing number of papers can also be seen in figure 3 (**?**).

We observe that sequential ACFlow even drops when $\mathbb{E}[U\backslash V]$ goes from 5 to 8. We posit that it arises due to ACFlow being used in a batch setting during inference time, while its training algorithm makes use of a sequential learning process. As a result it is unable to identify a good set of features in the batch setting and the accuracy drops.

G.4 Additional ablation study: Use of MOE instead of one single monolithic model improves the performance of baselines

In some cases, the baselines fail to scale to the large number of features that we provide to them. As a result, their classifiers are not able to train properly. To give the baselines a fair stage to compete, we deploy a mixture of experts model on the feature subsets that the baselines choose. We then report the best among the off the shelf baseline and the mixture of experts variant. This is done for both sequential and batch variants of the baselines. We illustrate this for the MNIST and CIFAR100 datasets in Figure 11.

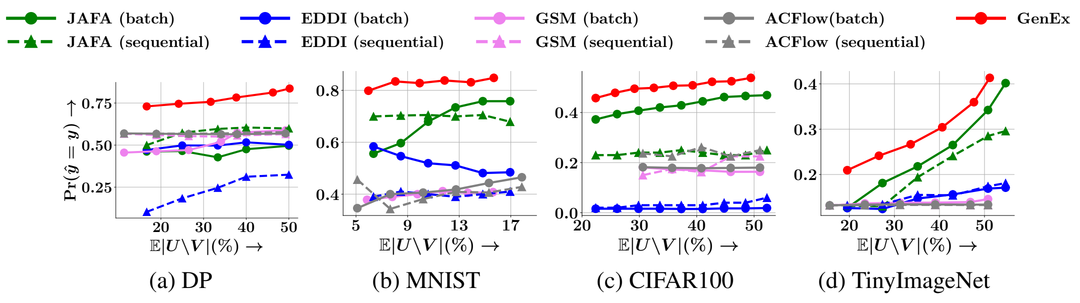


Figure 10: Comparison of GENEX against batch and sequential variants of baselines, *i.e.*, JAFA (?), EDDI (?), ACFlow (?) and GSM (?), in terms of the classification accuracy varying over the average number of oracle queries $\mathbb{E}|U\backslash V|$, for all four datasets.

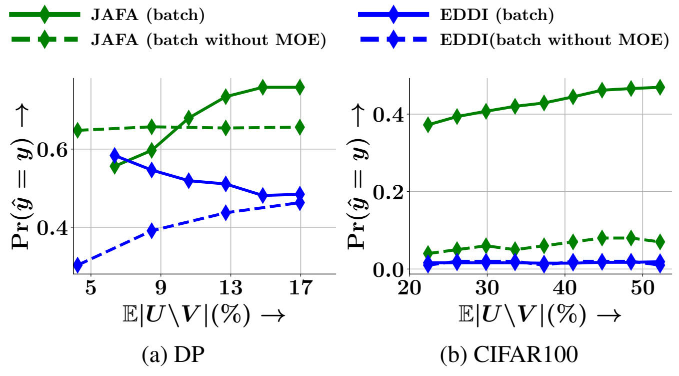


Figure 11: Comparison of accuracies for JAFA (batch) and EDDI (batch) against hybrid variants.

G.5 Classifier behaviour on generated data

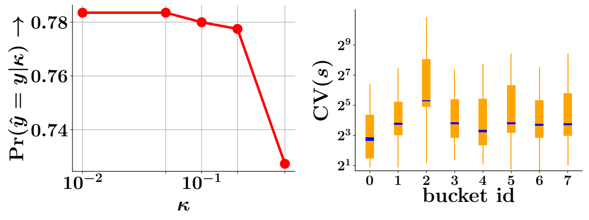


Figure 12: Analysis of $\mathbb{E}[|x[s]-x'[s]|]$

Can the set of accurate features generated by p_ϕ be used for prediction? We consider the behaviour of the classifier on generated data when the generator produces accurate features. Here we attempt to use the difference of oracle and generated features $\beta_x = ||\mathbf{x}-\mathbf{x}'||$. We plot the variation of $\Pr(y=\hat{y}\,|\,\kappa)$ against κ, where we use generated features on points which constitute the bottom κ fraction of the population in terms of β_x. The remaining points make use of oracle features. Figure 12 (left) shows that for instances with low κ, the accuracy is consistently high.

This immediately triggers the possibility of designing a predictor of $\mathbf{x}-\mathbf{x}'$, so that we can use generator for only those points where this difference small. But is $\beta_s = x[s]-x'[s]$ predictable? To investigate this, we compute β_s for all $i \in D_b$ for each bucket b. Then, we compute the coefficient of variation $\text{CV}(s) = \text{std}_s/\text{mean}_s$ where, $\text{std}_s, \text{mean}_s$ are the population standard deviation and mean of β_s across different instances in D_b. We plot $\text{CV}(s)$ for all buckets b in Figure 12 (right). It shows that the $\text{CV}(s)$ is quite large making β_s prediction infeasible.

G.6 Alternative inference algorithm

We consider an alternative to using the classifier's confidence in the inference algorithm. We train a supervised classifier τ_ψ. This takes $\mathbf{x}[\mathcal{O}\cup U\backslash V]\cup\mathbf{x}'[V]$ as input and predicts if the trained classifier $h_{\hat{\theta}_b}$ gives a correct prediction on $\mathbf{x}_i[\mathcal{O}_i\cup U_i\backslash V_i]\cup\mathbf{x}'_i[V_i]$ Specifically, τ_ψ makes the following prediction:

$$\tau_\psi(\mathbf{x}[\mathcal{O}\cup U\backslash V]\cup\mathbf{x}'[V]) = \mathbf{1}\left[\max_{y'} h_{\hat{\theta}_b}(\mathbf{x}[\mathcal{O}\cup U\backslash V]\cup\mathbf{x}'[V])[y'] = y\right] \tag{61}$$

Hence, the classifier τ_ψ is trained on the pairs $(\mathbf{x}_i[\mathcal{O}_i \cup U_i\backslash V_i] \cup \mathbf{x}'_i[V_i], r_i)_{i\in D}$, where $r_i = 1$ if the trained classifier gives correct prediction on $\mathbf{x}_i[\mathcal{O}_i \cup U_i\backslash V_i] \cup \mathbf{x}'_i[V_i]$ and $r_i = 0$, otherwise. Once trained, we use $\mathbf{x}'[V]$ instead of oracle features $\mathbf{x}[V]$, only for those instances, where $\tau_{\hat{\psi}}(\mathbf{x}[\mathcal{O} \cup U\backslash V] \cup \mathbf{x}'[V]) = 1$. The architecture of τ_ψ is taken to be the same as that of the classifier. Hence, for DP and MNIST datasets, we take $\tau_\psi(\bullet) = \text{Sigmoid}(\text{Linear}(\text{ReLU}(\text{Linear}(\text{Encoder}(\bullet)))))$ as the underlying architecture for τ_ψ. And for CIFAR100 and TI, the architectures are WideResnet and EfficientNet respectively.

Accuracy results of GENEX, GENEX(supervised) and GENEX($V = \emptyset$) are in Figure 13. The following can be observed: **(1)** GENEX outperforms the GENEX(supervised) variant. This is because the supervised classifier has only $\approx 70\%$ accuracy at identifying data points where the classifier would do well on generated features. **(2)** GENEX(supervised) does better than GENEX($V = \emptyset$) at higher budget as it is able to save cost more effectively.

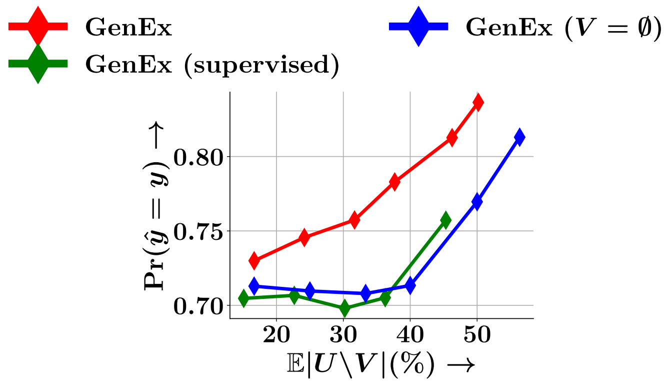


Figure 13: Comparison of accuracy of GENEX, GENEX(supervised) and GENEX($V = \emptyset$)

G.7 Efficiency analysis

We compare the training time per epoch and test times of GENEX against the baselines in Tables 14 and 15 for the smallest and the largest dataset. In each case, the numbers are reported without any parallelization. We see that GENEX performs competitively with most baselines. **(1)** GENEX's speed is due to its simple inference strategy of identifying a data point's cluster, and then utilizing the U, V for that bucket, which does not need any significant computation. **(2)** EDDI's better training time can be attributed to the fact that their generator model is trained with random subsets of features, and no greedy method is employed during training. However this is offset by GENEX's superior performance during inference as EDDI performs greedy acquisition without applying any trick to speedup.

Dataset	GENEX	EDDI	JAFA	GSM	ACFlow	CwCF	DiFA
DP	32 s	10 s	140 s	65 s	50 s	30 s	300 s
TI	2.5 hrs	2.5 hrs	3 hrs	3 hrs	2.75 hrs	2.5 hrs	3 hrs

Table 14: Training time per epoch. We train each method without any parallelization with a fixed batch size (1024) across all the methods. For JAFA, GSM, CwCF and DiFA, we consider an epoch as collection of RL episodes which cover the whole dataset exactly once.

Dataset	GENEX	EDDI	JAFA	GSM	ACFlow	CwCF	DiFA
DP	62 s	15 min	100 s	60 s	240 s	60 s	80 s
TI	20 min	2 hrs	13 min	20 min	16 min	15 min	20 min

Table 15: Test time for $|U\backslash V| \approx 50\%$. We test serially with a fixed batch size (1024).

All numbers are reported on experiments on 16-core Intel(R)693 Xeon(R) Gold 6226R CPU@2.90GHz with 115 GB RAM and one A6000 NVIDIA GPU with 48GB RAM.

G.8 Performance of EDDI in low dimensional datasets

We perform experiments with the gas detection dataset from kaggle. It is a classification task which uses 48 features (readings from sensors) to classify a gas sample into 6 classes. For $|U/V| = 35$. Following is the result.

GENEX	JAFA	EDDI
0.94	0.62	0.41

G.9 Position of observed feature in the image vs accuracy

In real life scenarios like medical diagnosis, the observed features have the semantics of being easily observed symptoms. However, here we do not assign any semantics to the features. We choose random subsets as $\mathcal{O}$ to demonstrate the effectiveness of the GENEX algorithm over a wide range of subsets. We have a sufficiently large number of random subsets $\mathcal{O}$, 20, 10, 5, 5 for the disease prediction, MNIST, CIFAR100 and TinyImagenet datasets. This is further validated using the error bars and p-values.

To look at the role of observed features in the image datasets, we contrast the cases where the observed features lie at the center of the image v/s the observed features are towards the edge of the image in the CIFAR100 dataset

Position of O	$\|U\backslash V\|$ = 20 %	$\|U\backslash V\|$ = 30%	$\|U\backslash V\|$ = 40%	$\|U\backslash V\|$ = 50%
Center	0.29	0.35	0.39	0.43
Edge	0.37	0.43	0.46	0.48

Table 16: Accuracy for different positions of $\mathcal{O}$ and different number of oracle queries

This indicates that if the initial set of features is itself informative (like the center of image), we do not gain much accuracy by querying. If instead the initial set is not informative, we can make big gains in accuracy via querying of informative features.

G.10 Comparison with other baselines

In the following, we use **?** and an unsupervised method which is facility location. Facility location is an unsupervised method that maximizes diversity among chosen features. Following are the results for the disease prediction dataset.

Model	$\|U\backslash V\| = 20\%$	$U\backslash V = 30\%$	$\|U\backslash V = 40\%$
GENEX	0.74	0.76	0.78
Facility location	0.45	0.56	0.64
?	0.69	0.71	0.71

Table 17: Accuracy for variouy methods in DP dataset

GENEX outperforms other methods.