Generator Assisted Mixture of Experts For Feature Acquisition in Batch

Vedang Asgaonkar, Aditya Jain, Abir De

Indian Institute of Technology 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.

Introduction

Supervised learning algorithms often assume access to a complete set of features $\mathbf{x} \in \mathbb{R}^d$ to model the underlying classifier $\Pr(y \mid \mathbf{x})$. However, in applications like healthcare, information retrieval, *etc.*, a key goal is feature acquisition (Babu and Vijayan 2016; Geng et al. 2007), 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.

Prior Work and Their Limitations

Driven by these motivations, feature acquisition is widely studied in literature. Earlier works used tools from active learning techniques (Melville et al. 2004; Saar-Tsechansky, Melville, and Provost 2009; Huang et al. 2018; Gong et al. 2019; Ma et al. 2018), which optimize measures based on variance, uncertainty or information gain. To improve their performance, a recent line of work explicitly optimizes the prediction accuracy (Shim, Hwang, and Yang 2018; Li and Oliva 2020, 2021; Janisch, Pevnỳ, and Lisỳ 2019, 2020a; Dulac-Arnold et al. 2012; Liyanage, Zois, and Chelmis 2021a,b; Hu et al. 2018; Yu et al. 2016), 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.

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 (Li and Oliva 2021; Ma et al. 2018). 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 (Indyk and Motwani 1999). 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 (Mualem and Feldman 2022) and approximate submodularity (Elenberg et al. 2018; Harshaw et al. 2019; De et al. 2021). 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.

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 $h(\mathbf{x})[y] = \Pr(y \mid \mathbf{x})$ and $\mathcal{H}$ is the hypothesis class. We also employ a generative model for features, denoted as $p(\mathbf{x}[V] \mid \mathbf{x}[\mathcal{O}])$, 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 $\mathbf{x}'[V] \sim p(\bullet \mid \mathbf{x}[\mathcal{O}])$.

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 $\mathbf{x}'_i[V_i] \sim p(\bullet \mid \mathbf{x}[\mathcal{O}_i])$ provide high accuracy on h. In theory, one can encode the above task in the following training loss:

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

and solve the following optimization problem.

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

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

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 (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014).

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.

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.

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) 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 (Indyk and Motwani 1999). 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 Charikar (2002, 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 (Charikar 2002). 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.

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 (Elenberg et al. 2018). Thus, it leads us to overcome the cross-instance coupling between the model parameters, U and V. It also facilitates distributed implementation.

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 h, generator p.

$\{D_b\}_{b\in[B]} \leftarrow$ PARTITION(D, B)
for bucket $b \in B$ **do**
  $U_b^*, \theta_b^*, \phi_b^* \leftarrow$GREEDYFORU$(q_{\max}, b, F, G_F)$
  $V_b^*, \leftarrow$GREEDYFORV$(\lambda, b, G_{\text{loss}})$

function GREEDYFORV$(\lambda, b, F, G_{\text{loss}})$
  Require: trained models θ_b^*, ϕ_b^* and the optimal subset U_b^*
  $V_b \leftarrow \emptyset$
  for $q \in [\lambda]$ **do**
    $e^* \leftarrow \text{argmin}_{e\notin V_b\cup\mathcal{O}_i}\ G_{\text{loss}}(e \mid V_b)$
    if $G_{\text{loss}}(e^* \mid V_b) < 0$ **then**
      $V_b \leftarrow V_b \cup e^*$
      break
  Return V_b

Then, we define the new loss F as follows.

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

Proposition 0.1 *Let F and* loss *are defined in Eqs. 5 and* (1)*, respectively. Then, we have:*

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

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.

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

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 pre-trained 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.*, $F(h_{\theta_b}, p_{\phi_b}; U_b \mid \mathcal{O}_i) = \ell(h_{\theta_b}(\mathbf{x}_i[\mathcal{O}_i \cup U_b]), y)$ and the task reduces to the well-known feature selection problem in (Elenberg et al. 2018).

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:

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

where λ is a hyperparameter.

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

[1] Proofs of all technical results are in (Asgaonkar, Jain, and De 2023)

function GREEDYFORU$(q_{\max}, b, h, p)$
  $U_b \leftarrow \emptyset$,
  $\theta_b \leftarrow$ TRAIN$(h, D_b, \{\mathcal{O}_i\})$ `#pretraining`
  $\phi_b \leftarrow$ TRAIN$(p, D_b, \{\mathcal{O}_i\})$ `#pretraining`
  for iter $\in [q_{\max}]$ **do**
    `# Use` h_{θ_b}`,` p_{ϕ_b} `to compute` G_F, F
    $e^* \leftarrow \text{argmin}_{e\notin U_b\cup\mathcal{O}_i}\ G_F(e \mid U_b)$
    if $G_F(e^* \mid U_b) < 0$ **then**
      $U_b \leftarrow U_b \cup e^*$
      $\theta_b, \phi_b \leftarrow$ TRAIN(F, U_b, b)
    else
      break
  Return U_b, θ_b, ϕ_b

Algorithm 2: Inference

Require: Observed test feature, $\mathbf{x}[\mathcal{O}]$, threshold τ, trained models $h_{\theta_b^*}, p_{\phi_b^*}, \{U_b^*, V_b^*\}$.

$b \leftarrow$ FINDBUCKET$(\mathbf{x}^*[\mathcal{O}])$
Query $\mathbf{x}[U_b^*\setminus V_b^*]$ from oracle
$\mathbf{x}'[V_b^*] \sim p_{\phi_b^*}(\bullet \mid \mathbf{x}^*[\mathcal{O} \cup U_b^*\setminus V_b^*])$
$\mathbf{x}_{\text{all}} = \mathbf{x}[\mathcal{O} \cup U_b^*\setminus V_b^*] \cup \mathbf{x}'[V_b^*]$
if $\max_y h_{\theta_b^*}(\mathbf{x}_{\text{all}})[y] < \tau$ **then**
  Query $\mathbf{x}[V_b^*]$ from oracle, $\mathbf{x}_{\text{all}} = \mathbf{x}[\mathcal{O} \cup U_b^*]$
$y^* \leftarrow \text{argmax}_y\ h_{\theta_b^*}(\mathbf{x}_{\text{all}})[y]$
Return y^*

$\theta_b^*(U), \phi_b^*(U)$ minimizes $\sum_{i\in D_b} F(h_{\theta_b}, p_{\phi_b}; U \mid \mathcal{O}_i)$. Thus, $\min_{\theta_b,\phi_b} \sum_{i\in D_b} F(h_{\theta_b}, p_{\phi_b}; U \mid \mathcal{O}_i)$ becomes a set function in terms of U. On the other hand, $\sum_{i\in D_b}(1 - \Delta_i(V)) \cdot \text{loss}(h, p; U_b^*, V \mid \mathcal{O}_i)$ 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:

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

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

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 (Higgins et al. 2017) to optimize regularized ELBO. Then, for each bucket b, we leverage two greedy algorithms for non-monotone functions (Mualem and Feldman 2022; Harshaw et al. 2019), 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 $G_F(e \mid U_b)$ as long as it admits a negative marginal gain, *i.e.*, $G_F(e^* \mid U_b) < 0$ 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 po-

tential 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 $\text{argmin}_e\ G_\bullet(e \mid U_b)$ 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 (Li and Oliva 2020) and (Ma et al. 2018), 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 $\mathbf{x}[\mathcal{O} \cup U_b \backslash V_b] \cup \mathbf{x}'[V_b]$.

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 (Elenberg et al. 2018).

Set function centric characterizations We first extend the notions of partial monotonicity (Mualem and Feldman 2022) and γ-weak submodularity (Elenberg et al. 2018; Harshaw et al. 2019).

Definition 0.2 *Given a set function $G : 2^{[n]} \to \mathbb{R}$, two sets S and T with $S, T \subseteq [n]$ and the marginal gain $G(S \mid T) := G(S \cup T) - G(T)$. 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 $\frac{\sum_{u \in S} G(u \mid T)}{|G(S \mid T)|} \in [\underline{\gamma}, \overline{\gamma}]$ for all S, T with $S \cap T = \emptyset$.*

Similar to Mualem and Feldman (2022), 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 (Mualem and Feldman 2022) implies (m, ∞)-partial monotonicity. A $\gamma-$weakly submodular function (Elenberg et al. 2018; Harshaw et al. 2019) is (γ, ∞)-weakly submodular. Next, we assume boundedness of few quantities, allowing us to characterize G_F and G_{loss}.

Assumption 0.3 ***(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 extended version, we discuss the validity of these assumptions as well as present the values of $\beta_\bullet$ across different datasets.*

Theorem 0.4 ($(\underline{m}, \overline{m})$-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 (Elenberg et al. 2018). 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 $G_{\text{loss}}(S \mid T) = 0$ 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 0.5 ($(\underline{\gamma}, \overline{\gamma})$-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$\{\nabla_\theta^2 \ell(h_\theta(\mathbf{x}[S]), y)\} \in [\zeta_{\min}, \zeta_{\max}]$ for all S; Then, the set function G_F is $(\underline{\gamma}_F, \overline{\gamma}_F)$-weakly submodular with*

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

$$\underline{\gamma}_F \geq \frac{-\frac{\nabla_{\max}^2}{2\zeta_{\min}} - K_{\underline{\gamma},1} L_\phi - K_{\underline{\gamma},2} \beta_\Delta \beta_x}{\frac{\nabla_{\max}^2}{2\zeta_{\max}} - K_{\underline{\gamma},3} L_\phi - K_{\underline{\gamma},4} \beta_\Delta \beta_x} \tag{11}$$

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 (Elenberg et al. 2018). 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 0.6 *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 $OPT = \operatorname{argmin}_{U_b:|U_b|\leq q_{\max}} G_F(U_b)$, we have*

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

where $m_F = \max\left(\overline{m}_F, 2\overline{m}_F/\underline{m}_F\right)$ 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 extended version.

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 (Li and Oliva 2020, 2021). Further details about the datasets are provided in the extended version. 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 (Shim, Hwang, and Yang 2018), EDDI (Ma et al. 2018), ACFlow (Li and Oliva 2020), GSM (Li and Oliva 2021), CwCF (Janisch, Pevnỳ, and Lisỳ 2020b) and DiFA (Ghosh and Lan 2023) 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 (Li and Oliva 2021; Shim, Hwang, and Yang 2018; Janisch, Pevnỳ, and Lisỳ 2020b; Ghosh and Lan 2023), greedy algorithms for maximizing rewards in (Ma et al. 2018; Li and Oliva 2020). 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. 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. Here, we present the results of the batch setting and, defer the results of the sequential setting in the extended version.

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 (Zagoruyko and Komodakis 2016) for TinyImagenet, h is EfficientNet (Tan and Le 2021). 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 (Higgins et al. 2017). Details are provided in the extended version (Asgaonkar, Jain, and De 2023). We set the number of buckets $B = 8, 8, 4, 4$ for DP, MNIST, CIFAR100 and TinyImagenet using cross validation.

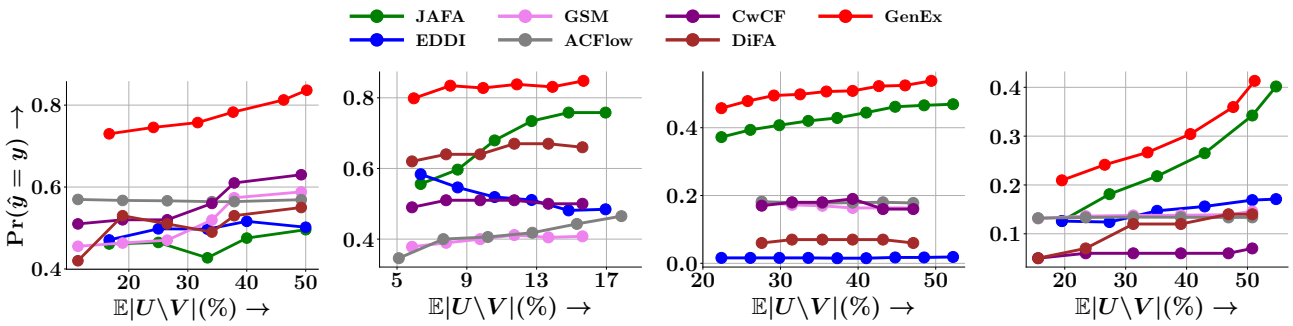
Evaluation protocol We split the entire dataset in 70% training, 10% validation and 20% test set. We use the validation set to cross validate λ and the number of buckets B. Having observed a feature $\mathbf{x}[\mathcal{O}]$ during test, we use the learned method to compute the set of new features to 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. [2]

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 (Li and Oliva 2021), and hence their accuracy stagnates after acquiring a few features (Li and Oliva 2021, Fig. 6, 7) and (Shim, Hwang, and Yang 2018, 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 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

[2] Our code is in https://github.com/VedangAsgaonkar/genex



(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 mean number of oracle queries $\mathbb{E}|U\backslash V|$, for all datasets.

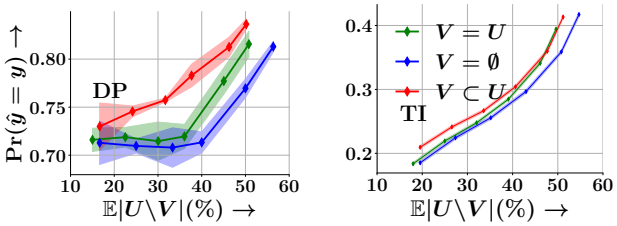


Figure 2: Ablation study: $\Pr(\hat{y} = y)$ vs. $\mathbb{E}[|U\backslash V|]$

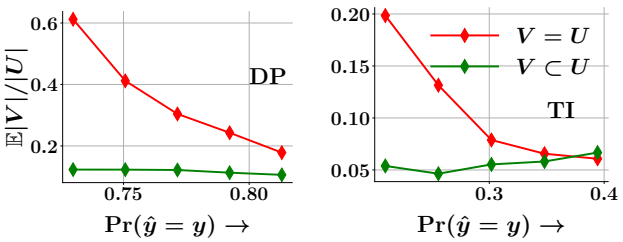


Figure 3: Ablation study: Saved cost vs. accuracy.

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 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 better bucket balance. To further probe why the RH achieves a better bucket balance, we instrument the conic-

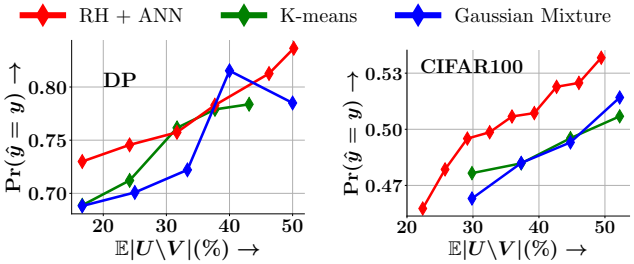


Figure 4: RH vs. other clustering methods

ity of the features which is a measure of how the feature vectors are concentrated in a narrow cone centered at the origin (Sharma, Talukdar et al. 2018). 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, an aggregate objective, promoting a few highly similar points in one cluster and leaving moderately similar instances dispersed among different clusters.

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. Work can be done incorporating exploration-exploitation on the greedy strategy.

References

Asgaonkar, V.; Jain, A.; and De, A. 2023. Generator Assisted Mixture of Experts For Feature Acquisition in Batch (Extended version of current paper). *arXiv preprint arxiv:2312.12574*.

Babu, R. L.; and Vijayan, S. 2016. Wrapper based feature selection in semantic medical information retrieval. *Journal of Medical Imaging and Health Informatics*, 6(3): 802–805.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 380–388.

De, A.; Okati, N.; Zarezade, A.; and Gomez-Rodriguez, M. 2021. Classification Under Human Assistance. *AAAI*.

Dulac-Arnold, G.; Denoyer, L.; Preux, P.; and Gallinari, P. 2012. Sequential approaches for learning datum-wise sparse representations. *Machine learning*, 89: 87–122.

Elenberg, E. R.; Khanna, R.; Dimakis, A. G.; and Negahban, S. 2018. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B): 3539–3568.

Geng, X.; Liu, T.-Y.; Qin, T.; and Li, H. 2007. Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 407–414.

Ghosh, A.; and Lan, A. 2023. Difa: Differentiable feature acquisition.

Gong, W.; Tschiatschek, S.; Nowozin, S.; Turner, R. E.; Hernández-Lobato, J. M.; and Zhang, C. 2019. Icebreaker: Element-wise efficient information acquisition with a bayesian deep latent gaussian model. In *Advances in Neural Information Processing Systems*, 14820–14831.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Harshaw, C.; Feldman, M.; Ward, J.; and Karbasi, A. 2019. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning*, 2634–2643. PMLR.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

Hu, X.; Zhou, P.; Li, P.; Wang, J.; and Wu, X. 2018. A survey on online feature selection with streaming features. *Frontiers of Computer Science*, 12: 479–493.

Huang, S.-J.; Xu, M.; Xie, M.-K.; Sugiyama, M.; Niu, G.; and Chen, S. 2018. Active feature acquisition with supervised matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1571–1579.

Indyk, P.; and Motwani, R. 1999. Approximate Nearest Neighbors: Towards Removing the.

Janisch, J.; Pevnỳ, T.; and Lisỳ, V. 2019. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3959–3966.

Janisch, J.; Pevnỳ, T.; and Lisỳ, V. 2020a. Classification with costly features as a sequential decision-making problem. *Machine Learning*, 109: 1587–1615.

Janisch, J.; Pevnỳ, T.; and Lisỳ, V. 2020b. Classification with costly features as a sequential decision-making problem. *Machine Learning*, 109: 1587–1615.

Li, Y.; and Oliva, J. 2021. Active feature acquisition with generative surrogate models. In *International Conference on Machine Learning*, 6450–6459. PMLR.

Li, Y.; and Oliva, J. B. 2020. Dynamic Feature Acquisition with Arbitrary Conditional Flows. *arXiv preprint arXiv:2006.07701*.

Liyanage, Y. W.; Zois, D.-S.; and Chelmis, C. 2021a. Dynamic instance-wise joint feature selection and classification. *IEEE Transactions on Artificial Intelligence*, 2(2): 169–184.

Liyanage, Y. W.; Zois, D.-S.; and Chelmis, C. 2021b. Dynamic instance-wise joint feature selection and classification. *IEEE Transactions on Artificial Intelligence*, 2(2): 169–184.

Ma, C.; Tschiatschek, S.; Palla, K.; Hernández-Lobato, J. M.; Nowozin, S.; and Zhang, C. 2018. Eddi: Efficient dynamic discovery of high-value information with partial vae. *arXiv preprint arXiv:1809.11142*.

Melville, P.; Saar-Tsechansky, M.; Provost, F.; and Mooney, R. 2004. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, 483–486. IEEE.

Mualem, L.; and Feldman, M. 2022. Using Partial Monotonicity in Submodular Maximization. *arXiv preprint arXiv:2202.03051*.

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

Sharma, A.; Talukdar, P.; et al. 2018. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 122–131.

Shim, H.; Hwang, S. J.; and Yang, E. 2018. Joint active feature acquisition and classification with variable-size set encoding. In *Advances in neural information processing systems*, 1368–1378.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tan, M.; and Le, Q. 2021. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, 10096–10106. PMLR.

Yu, K.; Wu, X.; Ding, W.; and Pei, J. 2016. Scalable and accurate online feature selection for big data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(2): 1–39.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.