Generator Assisted Mixture of Experts For Feature Acquisition in Batch

Vedang Asgaonkar, Aditya Jain, Abir De

Indian Institute of Technology Bombay
{vedang, adityajainjhs, abir}@ece.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 $x \in \mathbb{R}^d$ to model the underlying classifier $Pr(y | x)$. 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 $O \subset \{1, \ldots, d\}$ and the goal is to query for a new subset $U$ from the unobserved set of features: $U \subset \{1, \ldots, d\} \backslash O$. 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, Pevný, and Lisý 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 $O$ 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, \((m, m)\)-partial monotonicity and \((\gamma, \tau)\)-weak submodularity, extending recent notions of partial monotonicity (Muallem 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 \(x \in \mathbb{R}^n\) to represent a feature vector and \(y \in Y\) for the associated label. \(I = [n]\) denotes the set of feature indices. \(O \subseteq I\) represents the indices of observed features, while \(U \subseteq I \setminus O\) represents indices of subset of features to be queried. Given a feature vector \(x\), \(x[i]\) consists of features indexed by \(O\). We denote a singleton feature as \(x[s] \in \mathbb{R}\) for index \(s\).

Our classifier is denoted as \(h \in \mathcal{H}\), where \(h(x)[y] = \Pr(y | x)\) and \(\mathcal{H}\) is the hypothesis class. We also employ a generative model for features, denoted as \(p(x | V) \mid x | O\)\], which generates new features \(x[V]\) conditioned on observed features \(x[O]\). For clear disambiguation from oracle features \(x[O]\), we use \(x^*\) 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 \(\ell(h(x), y)\).

High level objective Given an instance \(x\), we initially observe only a subset of the features \(x[O]\) indexed by \(O\) which varies across the instances. In general, this small subset is not sufficient for accurate prediction. Hence, we would seek to query new features \(x[U]\) subject to a maximum number of oracle queries. Thus, our key goal is to use \(x[O]\) to determine the optimal choice of \(U\) among all such possible subsets, such that \(x[O \cup U]\) 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 \(x[U]\) 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 \(x[V]\), which are drawn the feature generator \(p\) can lead to similar accuracy as the oracle features \(x[V]\).

Problem statement During training, we are given the architectures of a classifier \(h\) and the feature generator \(p\) as well as the training set \(\{(x[i], y_i, O_i)\}_{i \in D}\) and a budget \(q_{\text{max}}\) 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 \(|U_i \cap V_i| \leq q_{\text{max}}\), so that the oracle features \(x_i[O_i \cup U_i \cap V_i]\) and the generated features \(x_i[V_i] \sim p(\cdot | x[O_i])\) provide high accuracy on \(h\). In theory, one can encode the above task in the following training loss:

\[
\text{loss}(h, p; U_i, V_i \mid O_i) = E_{x_i[V_i] \sim p(\cdot | x_i[O_i])}\left(\ell(h(x_i[O_i \cup U_i \cap V_i] \cup x_i[V_i]), y_i)\right);
\]

and solve the following optimization problem.

\[
\begin{align*}
\text{minimize} \quad & \sum_{i \in D} \text{loss}(h, p; U_i, V_i \mid O_i) \\
\text{subject to,} \quad & |U_i \cap V_i| \leq q_{\text{max}} \quad \text{for all } i \in D \tag{3}
\end{align*}
\]

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 \(x^*[V]\) to its gold value \(x[V]\), a small difference \(|x'[V] - x[V]|\) 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 \(O_i\) varies widely across \(i \in D\). Moreover, the observed feature values \(x_i[O]\) for the same \(O\) 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 \(x_i[O_i \cup U_i \cap V_i]\) are very diverse, owing to a large variety of \(O_i, U_i\) 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 higher 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 \( \{x_i|\mathcal{O}_i\}\}_{i \in D} \) 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 \( \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M] \in \mathbb{R}^{n \times M} \), with \( \mathbf{w}_m \sim N(0, \mathbb{I}_n) \). Each \( \mathbf{w}_m \) defines a random hyperplane through the origin. We hash each observation \( x_i|\mathcal{O}_i \) to a bucket \( b = \text{sign}(\mathbf{W}^\top \mathbf{x}_i|\mathcal{O}_i) \), where \( x_i|\mathcal{O}_i \) is padded with zeros to match dimensionality. Each bucket is thus identified by a vector of \( \pm 1 \) 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 \( x \) doesn’t affect the cluster assignment of another instance \( x' \). (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 \( x|\mathcal{O} \), 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 \( x|\mathcal{O} \). This bucket id can be immediately obtained by computing \( b^* = \text{sign}(\mathbf{W}^\top x|\mathcal{O}) \), 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 \( x|\mathcal{O} \) 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., \( D = D_1 \cup D_2 \cup \ldots \cup D_B \), we build a mixture of \( B \) independent classifiers \( \theta_{\theta_b} \) and generators \( p_{\phi_b} \), parameterized with \( \theta_b \) and \( \phi_b \) for a bucket \( b \). This reduces the joint optimization (2) problem into the following

\[
\text{minimize } \sum_{b \in [B]} \sum_{i \in [O]} \text{loss}(\theta_{\theta_b}, \phi_{\phi_b}; U_b, V_b | \mathcal{O}_i)
\]

such that, \( |U_b \setminus V_b| \leq q_{\text{max}} \forall b \in [B] \) (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 \( \text{loss}(\theta_{\theta_b}, \phi_{\phi_b}; U_b, V_b | \mathcal{O}_i), \) 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** \( \text{loss}(\theta_{\theta_b}, \phi_{\phi_b}; U_b, V_b | \mathcal{O}_i) \)— 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 \( F(\theta_{\theta_b}, \phi_{\phi_b}; U_b | \mathcal{O}_i) \) whose optimal value with respect to \( U_b \) for a given \( \theta_b \) and \( \phi_b \) 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 \( \{(x_i, y_i, \mathcal{O}_i)\}_{i \in D} \). Having computed the pre-trained \( \theta_{\theta_b} \) and the pre-trained generator \( \phi_{\phi_b} \), we define \( \Delta_i(U_b) \) as the uncertainty of the classifier when \( U_b \) uses the generated features \( \mathbf{x}'(U_b) \sim \phi_{\phi_b}(\bullet | \mathcal{O}_i) \) for the whole set \( U_b \), i.e., \( \Delta_i(U_b) = \mathbb{E}_{\mathbf{x}' \sim \phi_{\phi_b}(\bullet | \mathcal{O}_i)}[1 - \max_{\mathbf{h}} \theta_{\theta_b}(\mathbf{x}'|\mathcal{O}_i) \cup \mathbf{x}'[U_b][y]] \). Thus, \( \Delta_i(U_b) \in \{0,1\} \). We rescale \( \Delta_i(U_b) \) by dividing it with \( 1 - \frac{1}{|O|} \), so that it lies in \([0,1]\).
Algorithm 1: Training

Require: Training data \{\{(x_i, y_i, O_i)\}_{i \in D}\}. Number of buckets \(B = 2^b\), the classifier \(h\), generator \(p\).

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

1: function GREEDYFORU(\(q_{\max}, b, F, G_F)\)
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: \(e^* \leftarrow \text{argmin}_{e \in V_b \cup O_b} G_{\text{loss}}(e | V_b)\)
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.
\[
F(h_{\theta_b}, p_{\phi_b}; U_b | O_b) = \Delta_l(U_b) \cdot \ell(h_{\theta_b}(x_i | O_i \cup U_b), y) + (1 - \Delta_l(U_b)) \cdot \mathbb{E}_{x_i \in U_b^*} \ell(h_{\theta_b}(x_i | O_i \cup x_i^*}, y) \quad (5)
\]

Proposition 0.1 Let \(F\) and loss are defined in Eqs. 5 and (1), respectively. Then, we have:
\[
\min_{\{U,V\} : U \cap V \subseteq q_{\max}} \sum_{i \in D} \text{loss}(h, p; U_i, V_i | O_i) \leq \min_{U_b : |U_b| \leq q_{\max}} \sum_{i \in D_h} F(h, p; U_b | O_i) \quad (6)
\]

The set-optimal value of the above objective is an upper bound of \(\text{loss}(\cdot)\) in Eq. (1), as stated formally here. Hence, instead of minimizing \(\text{loss}(\cdot)\), we seek to solve the following optimization problem for each bucket \(b\).
\[
\min_{\theta_b, \phi_b, U_b} \sum_{i \in D_h} F(h_{\theta_b}, p_{\phi_b}; U_b | O_i), \text{ s.t., } |U_b| \leq q_{\max} \quad (7)
\]
The objective loss(\(\cdot\)) (1) queries two different sets of features from the oracle and the generator, i.e., \(U_b \backslash V_b\) 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., \(F(h_{\theta_b}, p_{\phi_b}; U_b | O_i) = \ell(h_{\theta_b}(x_i | O_i \cup U_b), y)\) 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:
\[
\min_{\theta_b, \phi_b, U_b} \sum_{i \in D_h} (1 - \Delta_l(U_b)) \cdot \text{loss}(h_{\theta_b}^*, p_{\phi_b}^*; U_b^*; V_b, O_i) \quad (8)
\]
where \(\lambda\) is a hyperparameter.

Objectives in (7), (8) as set functions Here, we represent the objectives in the optimizations (7), (8) as set functions, which will be later used in our training and inference methods and approximation guarantees. Given \(U\), the optimal solution of the objective

\[\text{Proofs of all technical results are in (Asganakar, Jain, and De 2023)}\]
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 $	ext{argmin}_{\gamma} G_*(e | U_b)$ instead of enumerating $G_*$ 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 $x[O]$, we first find the bucket $b$. It then queries the oracle for $x[U_b \setminus V_b]$. Taking $x[O \cup U_b \setminus V_b]$ as input, the generator produces $x'[V_b]$. If the confidence of the classifier with these features is lower than a threshold $\tau$, then GENEX does not use the generated features for the final prediction and, further query $x[V_b]$ from the oracle and predict $y$ using $x[O \cup U_b]$. However, otherwise if the confidence of the classifier with the features is higher than $\tau$, we use the generated features and compute $y$ using $x[O \cup U_b \setminus V_b] \cup x'[V_b]$.

**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 $\theta^*_F(U)$, $\phi^*_F(U)$ in polynomial time when the objective is convex with respect to both $\theta$ and $\phi$. However, the simultaneous computation of the optimal set $U^*$ and the model parameters $\theta^*_F$ and $\phi^*_F$ 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 (Maulen and Feldman 2022) and $\gamma$-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 $G(S | T) := G(S \cup T) - G(T)$. Then we define the following properties.

1. **($m, \overline{m}$)-Partial monotonicity:** The set function $G$ is ($m, \overline{m}$)-partially monotone ($m \geq 0$) if $G(T)/G(S) \in [m, \overline{m}]$ for all $S, T$ with $S \subseteq T \subseteq [n]$.

2. **($\gamma, \tau$)-Weak submodularity:** The set function $G$ is ($\gamma, \tau$)-weakly submodular if $\sum_{S, T \subseteq [n]} G(S | T) / G(S \cup T)$ is $\gamma$-\tau-weakly submodular for all $S, T$ with $S \cap T = \emptyset$.

Similar to Maulen and Feldman (2022), we define that $G(T)/G(S) = 1$ if $G(S) = 0$. Note that, a ($m, \overline{m}$) partially monotone function $G$ is monotone increasing (decreasing) if $m_1 = 1 (\overline{m}_1 = 1)$. Moreover, $m$—partial monotonicity introduced in (Maulen and Feldman 2022) implies $(m, \infty)$—partial monotonicity. A $\gamma$—weakly submodular function (Elenberg et al. 2018; Harshaw et al. 2019) is ($\gamma$, $\infty$)—weakly submodular. Next, we assume boundness of few quantities, allowing us to characterize $G_F$ and $G_{\text{loss}}$.

**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: $0 < \Delta_{\min} < |\Delta_i(U)| \leq |\Delta_{\max}|$ $0 < \ell_{\min} < (\ell(h_\theta(x_i), y_i)) \leq \ell_{\max}$. (3) Lipschitzness: The loss function $\ell(h_\theta(x), y)$ is Lipschitz with respect to $x$. (4) Boundedness of features: $|x_i|$ is bounded for all $i$. In extended version, we discuss the validity of these assumptions as well as present the values of $\beta$ across different datasets.

**Theorem 0.4** ($(m, \overline{m})$)-Partial monotonicity (1) The set function $G_F$ is ($\overline{m}_F, m_F$)-partially monotone where $m_F = 1 + K_F \beta_F \beta_\Delta$ and $m_F = (1 + K_1 \beta_F + K_2 \gamma F + K_3)\beta_\Delta^{-1} - 1$. (2) The set function $G_{\text{loss}}$ is ($\overline{m}_{\text{loss}}, m_{\text{loss}}$)-partially monotone where $m_{\text{loss}} = 1 + K_1 \beta_{\text{loss}} \beta_\Delta$ and $m_{\text{loss}} = (1 + K_1 \beta_{\text{loss}} + K_3)\beta_\Delta^{-1} - 1$. Here $K_F$ depend on the Lipschitz constant of the loss with respect to $x$ and the bounds on loss $\ell$ and the uncertainty $\Delta$.

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 $F$ in Eq. (5), then the optimization (7) becomes oblivious to the generative model $p_{\phi_b}$. 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 $G_{\text{loss}}$ 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 $q_{\text{max}}$. Since, we have $G_{\text{loss}}(S | T) = 0$ in a simple cases where $\beta_\Delta = 0$, ($\gamma, \tau$)-weak submodularity does not hold for $G_{\text{loss}}$ in most cases. Thus, we present the ($\gamma, \tau$)-weak submodularity property of only $G_F$ under additional assumptions.

**Theorem 0.5** ($(\gamma, \tau$)-Weak submodularity) Assume that the loss $\ell(h_\theta(x), y)$ be convex in $\theta$ with $\nabla_\theta \ell(h_\theta(x), y) \leq \nabla_{\text{max}}$ and Eigenvalues $\{\lambda_{\text{max}}, \lambda_{\text{min}}\}$ for all $S$. Then, the set function $G_F$ is ($\gamma_{F}, \tau_F$)-weakly submodular with

$$
\text{\tau}_F \leq \max \left\{ -\frac{\gamma_F^2}{\lambda_{\text{max}}} + K_{\gamma_1} L_\phi + K_{\gamma_2} \beta_\Delta \beta_x, \frac{\gamma_{F}^2}{\lambda_{\text{min}}} + K_{\gamma_3} L_{\phi} + K_{\gamma_4} \beta_\Delta \beta_x - \frac{\gamma_{F}^2}{\lambda_{\text{max}}} + K_{\gamma_1} L_{\phi} + K_{\gamma_2} \beta_\Delta \beta_x \right\};
$$

$$
\text{\gamma}_F \geq \min \left\{ -\frac{\gamma_{F}^2}{\lambda_{\text{min}}} - K_{\gamma_3} L_{\phi} + K_{\gamma_4} \beta_\Delta \beta_x \right\};
$$

(11)
where $K_{\gamma, q}$ and $K_{\gamma, 0}$ are constants that depend on the Lipschitz constant of the loss w.r.t. $x$; $L_{\phi}$ is the Lipschitz constant of $F$ w.r.t. $\phi$.

In the absence of the generator, $\beta_{\Delta}, \beta_{\varepsilon}, L_{\phi}$ are zero. Then, $-G_F$ is $\gamma$-weakly submodular with $\gamma > \zeta_{\min}/\zeta_{\max}$ which coincides with the results of (Ellenberg et al. 2018). Next, we present the approximation guarantee of our greedy algorithm for solving the optimization problem (7) presented via \textsc{GreedyForU}, when $\ell(h_0(x), y)$ is convex in $\theta$.

**Theorem 0.6** Assume that, given a bucket $b$, $G_F$ is $(m_F, \overline{m}_F)$-partially monotone, $(\gamma_F, \overline{\gamma}_F)$-weakly submodular set function. Suppose $U_b^*$ is the output of \textsc{GreedyForU} in Algorithm 1. Then, if $OPT = \arg\min_{U_b} G_F(U_b)$, we have

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

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

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 $|Y| = 42$. Here, different features indicate different symptoms and the classes indicate different diseases. For the image datasets, we take the pixels or image crops as features, following previous work (Li and Oliva 2020, 2021). Further details about the datasets are provided in the extended version. For each dataset, we use cross-validation to select the number of buckets $B$. We set the number of buckets $B = 8, 8, 4, 4$ for DP, MNIST, CIFAR100 and TinyImagenet.

**Evaluation protocol** We split the entire dataset in 70% training, 10% validation and 20% test set. We use the validation set to cross validate $\lambda$ and the number of buckets $B$. Having observed a feature $x[O]$ 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 $x'[V]$ by drawing from the generator, whereas we query $x[U \backslash V]$ 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.

**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 $E[|V|/|U|]$, 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) JAPA 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 $E[|V|/|U|]$.

Note that, even in GenEX ($V = U$), not all instances result in high

---

2 Our code is in https://github.com/VedangAsgaonkar/genex
The thirty-eight AAAI conference on artificial intelligence (AAAI-24)

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 $E[U \setminus V]$, for all datasets.

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

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

Figure 4: RH vs. other clustering methods

**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 $E[U \setminus V]$. 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 conicity 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: $\text{conicity}(D) = \frac{1}{|D|} \sum_{i \in D} \cos \left( \sum_{j \in D} x_i \cdot x_j / |D| \right)$. 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 $(m, m)$-partial monotonicity and $(\gamma, \gamma)$-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


