A Survey of Query Strategies for Active Learning

Avishan Bewtra
Penn State University
State College, PA
aqb6074@psu.edu

Christopher Ye
Penn State University
State College, PA
cfy5119@psu.edu

***Abstract*—Annotating datasets is becoming increasingly expensive, especially as larger models require larger annotated datasets to achieve good performance. Experts are required to annotate certain types of data, such as medical images, making dataset generation even more expensive. In scenarios where data-labeling is high cost, as few samples as possible should be labeled while maintaining sufficient performance. Typically, supervised learning uses a dataset with randomly sampled points (which may need to be annotated), where creating a larger labeled set would involve labeling new troves of randomly sampled points. Active learning, however, provides a framework in which (instead of sampling some random subset of points), we instead label points from an unlabeled pool $\mathcal{U}$ based on their utility. This has the potential to limit the number of costly labels to be made by humans, and in turn, makes dataset generation more pragmatic and economical. In this survey, various query strategies—or the strategy used to select the most valuable points to be annotated—are discussed. These strategies include query by committee, uncertainty sampling, expected model change, expected error reduction, and batch sampling with similarity considerations. The advantages and limitations of each of the query strategies are discussed and performance is compared.**

I. Introduction to Active Learning

As large-scale data and compute power become more widely available, machine learning has become increasingly practical. But the ability to generate useful labeled datasets remains difficult and expensive. This paper is motivated by this problem: the efficient creation of sufficient labeled datasets for a machine learning task. Instead of making the labeling of individual datapoints more efficient, active learning attempts to minimize the overall number of datapoints required to achieve a certain performance. In theory, by carefully selecting better datapoints to be labeled, smaller datasets are needed to achieve equal performance to labeling a set of randomly sampled points.

Consider a data distribution, $\mathcal{D}$, from where an insufficient labeled dataset, $\mathcal{L}$, and an unlabeled pool of datapoints, $\mathcal{U}$, are obtained. The goal is to find a function, $\mathcal{F}$, that maps datapoints from $\mathcal{D}$ to the correct labels.

Furthermore, assume there is an oracle that can provide labels to datapoints from the unlabeled pool, $\mathcal{U}$. Note that if all the samples from the data distribution, $\mathcal{D}$, are unlabeled, a randomly selected subset of the unlabeled pool, $\mathcal{U}$, may be labeled to create the initial labeled pool, $\mathcal{L}$, used in active learning algorithms.

We make the following assumptions about our given conditions:

- This oracle is costly, and only as many samples as absolutely necessary should be labeled.
- The oracle is always correct and does not mislabel datapoints.
- The unlabeled pool, $\mathcal{U}$, is representative of the data distribution, $\mathcal{D}$.
- The unlabeled pool, $\mathcal{U}$, is sufficiently big that it is infeasible to label the entire set.

Traditionally, one might try a supervised (or semi-supervised) approach, where a randomly selected subset of the unlabeled pool with a given size, $\mathcal{D}_{sample} \subset \mathcal{U}$, is labeled using the oracle (this set may be empty). This newly labeled set is combined with the original labeled set, $\mathcal{L}$ to produce a training set, $\mathcal{D}_{sample} \cup \mathcal{L}$, for the supervised (or sem-supervised) approach. Active learning attempts to achieve model performance at least equal to a model trained on $\mathcal{D}_{sample} \cup \mathcal{L}$ while querying fewer points. The general hypothesis is if datapoints are carefully selected based on utility, a fewer number of datapoints will be required to achieve the same performance as a randomly sampled dataset. These selected datapoints sent to the oracle are called queries, and the method in which queries are selected is called the query strategy.

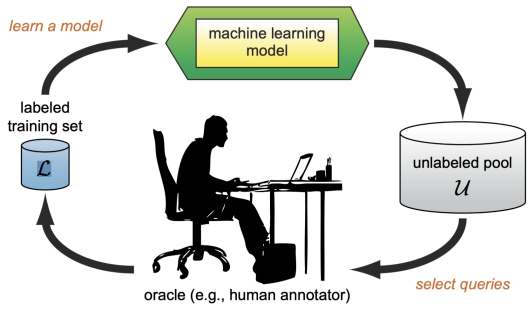

Fig. 1. Flow chart of the active learning framework.

The active learning framework is outlined in figure 1 and consists of four repeated steps: the querying of a datapoint (or datapoints) from an unlabeled pool, $\mathcal{U}$; the annotation of the query (or queries); the updating of the labeled dataset, $\mathcal{L}$; and the re-training of the model, $\mathcal{F}$. This survey is most concerned with query strategies, or the selection of points sent to the oracle.

This is further described in the pseudo-code of Algorithm 1. First we define $\mathcal{D}$, $\mathcal{L}$, $\mathcal{U}$, and a desired performance level, $\mathcal{T}$. Then we continue with the following iteration steps: re-training a model $\mathcal{M}$ based on the given $\mathcal{L}$; querying a point based on $\mathcal{M}$; having the oracle label our queried point, giving us $O(\mathcal{Q})$; and adding our queried point and label to $\mathcal{L}$. The iteration stops when a certain performance, $\mathcal{T}$, is reached.

Algorithm 1 Psuedo-code for the Active Learning Framework

Require: $\mathcal{D}, \mathcal{L}, \mathcal{U}, \mathcal{T} > 0, \mathcal{M}(\theta), \mathcal{O} : x \in D \rightarrow$ label where $\mathcal{M}(\theta)$ is the model parameterized by θ, and $\mathcal{O}$ represents the oracle.

while Performance $< \mathcal{T}$ **do**
  Train $\mathcal{M}(\theta)$ on $\mathcal{L}$
  Use $\mathcal{M}(\theta)$ to select a query point $\mathcal{Q} \in \mathcal{U}$
  $\mathcal{U} \leftarrow \mathcal{U} \backslash \{Q\}$
  Evaluate $O(\mathcal{Q})$.
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{((Q), O(\mathcal{Q})\}$
end while

While ideally, the active learning loop continues until the desired performance is reached, often costs restrict this from occurring and the number of iterations is limited to some constant number.

This survey is broken up into four sections, including an introduction, a detailed overview of each class of query strategies, a comparison of the query strategies, and a conclusion.

We break the literature on query strategies for active learning down based on the different ways that the utility of a datapoint (or datapoints) is measured. Specifically, we've broken the literature down into the following categories: query by committee; uncertainty sampling; expected model change; expected error reduction; and batch sampling with similarity consideration.

II. Query Strategies

A. Query by Committee

Query by committee (QBC) estimates the disagreement between an ensemble of models. First proposed in 1992, Seung et al. summarize query by committee in a line from their paper, "Query by Committee"; "A committee of [student models] is trained on the same dataset. The next query is chosen according to the principle of maximal disagreement" [13].

Two things are required to implement a QBC algorithm:

- One must create a committee of models that represent different regions of the version space.
- One must be able to measure the disagreement between models in the committee.

A model from the version space is one with parameters which allow it to be consistent with the training set. In other words, all of its predicted classifications must be correct for all labeled examples. With certain model architectures, it may be difficult to find parameters which allow to model to be in the version space (in some cases it may be impossible). The details of constructing a model from the version space are out of the scope of this survey.

In addition, as the training set, $\mathcal{L}$, is updated, the version space changes. This presents an implementation challenge, as well as the critical property behind QBC. As dataset updates continue, the version space shrinks, bringing us closer and closer to an optimal solution [10].

There are also many ways to measure disagreement between models in a committee. First, vote entropy (VE) calculates the total disagreement between different models for the label of a given datapoint. Given a datapoint x, $V(y_i)$ represents the number of "votes" that a label y_i receives from the committee's model predictions. C is the number of committee members. A query selected by vote entropy is specified by the following optimization problem:

$$x^*_{VE} = \underset{x}{\operatorname{argmax}}\Big(-\sum_{y_i} \frac{V(y_i)}{C} \log(\frac{V(y_i)}{C})\Big)$$

VE represents the entropy of the votes (or predictions) from the committee models for a datapoint. VE can be calculated with only a predicted classifications from each committee model. For a measure of entropy with respect to predictions which take the form of a probability distribution, one can use Kullback-Leibler (KL) divergence. Consider a model from the committee, $\theta^{(c)}$, and the entire committee, $\mathcal{C}$. A query selected by KL divergence is specified by:

$$x^*_{KL} = \arg\max_x \frac{1}{C} \sum_i^C D(P_{\theta^{(c)}} || P_{\mathcal{C}})$$

$$D(P_{\theta^{(c)}} || P_{\mathcal{C}}) = \sum_i P(y_i|x;\theta^{(c)}) \log \frac{P(y_i|x;\theta^{(c)})}{P(y_i|x;\mathcal{C})}$$

Using the optimization problems specified, we can calculate the datapoint upon which the committee disagrees based on some measure of disagreement. That point is then sent to the oracle to be labeled.

Trivially, in query by committee, each iteration of the active learning loop must reduce the size of version space. All committee members are in the version space, $\mathcal{VS}$, and the selected query point is one for which the committee has maximal disagreement on the classification. If at least one disagreement is present in the committee, then at least one $\mathcal{F} \in \mathcal{VS}$ will no longer be consistent with all the labeled examples after the query is made and the true label is known.

Directly measuring the cardinality of the version space for infinite version spaces is not informative about how much information was gained from a query. A more general measure of the size of the version space is the probability that a function chosen from the set of all candidate functions, $\mathcal{C}$, according to some distribution, is a member of the version space [2]. Let $\mathcal{VS}_{\text{pre-query}}$ be the version space before some query is made, and $\mathcal{VS}_{\text{post-query}}$ be the version space after some query is made. The instantaneous information gained from a query can be measured as [2] :

$$\log\Big(\frac{P(\mathcal{VS}_{\text{pre-query}})}{P(\mathcal{VS}_{\text{post-query}})}\Big)$$

Assuming that the information gain from each query is lower bounded by some constant $c > 0$, then the cumulative information gain, $\log(\frac{1}{P(\mathcal{VS})})$, must increase at least linearly with the number of queries and is upper bounded by the information gained if all the labels were known [2]. With a committee of size two and a stream of unlabeled samples $x_1, x_2, ..., x_\infty$ sampled from $\mathcal{D}$, the expected gap between points with disagreement which are queried increases exponentially [2].

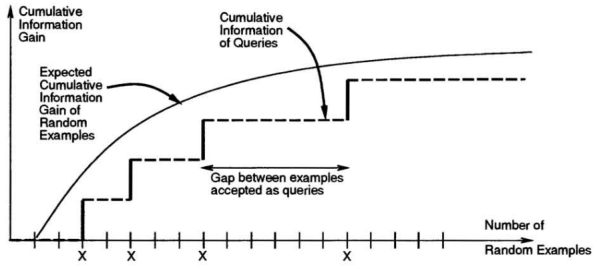

Fig. 2. A graph showing the cumulative information gain for query by committee compared to the cumulative information gain from having access to all the labels. The X indicates that a sample was chosen to be queried. Source: [2]

Formally, derived in [2], the authors present Theorem 1.

Theorem 1: Suppose that a concept class $\mathcal{C}$ has VC-dimension $d < \infty$ and the expected information gained by the two member committee algorithm is bounded by $c > 0$, independent of the query number and of the previous queries. Then the probability that one of the two committee members makes a mistake on a randomly chosen example with respect to a randomly chosen $f \in \mathcal{C}$ is bounded by

$$(3 + O(c^{c_1 n}))\frac{n}{d} exp\Big(-\frac{c}{2(d+1)}n\Big)$$

for some constant $c_1 > 0$, where n is the number of queries asked so far [2].

A direct consequence from this theorem is that under certain assumptions, an active learning algorithm using QBC can only sample $O(\log(n))$ points from a set of n points to achieve similar performance to a model that was trained with labels for the entire set of n points.

Some types of candidate function classes for which this constant information gain assumption holds include candidate functions where functions have the form

$$f_{\vec{w}}(\vec{x}, t) = \left\{ \begin{array}{ll} 1, & \text{if } \vec{w} \cdot \vec{x} > t \\ 0, & \text{if } \vec{w} \cdot \vec{x} \leq t \end{array} \right\}$$

where $\vec{w} \in R^d$ is drawn from some uniform distribution on a convex body [2]. This condition is also holds for candidate classes where functions have the form

$$f_{\vec{w}}(\vec{x}, t) = \left\{ \begin{array}{ll} 1, & \text{if } F(\vec{w}, \vec{x}) > t \\ 0, & \text{if } F(\vec{w}, \vec{x}) \leq t \end{array} \right\}$$

where $\vec{x} \in R^I, \vec{w} \in R^d$ and $\mathcal{F}$ is a smooth function of $\vec{x}, \vec{w}$ [2]. This is true for feed-forward neural networks with smooth transfer functions [2].

Pros:

- QBC is relatively easy to compute, compared to other query strategies described later in this paper. For each model, one must only run inference on each datapoint in order to calculate disagreement across models.
- QBC has also seen strong empirical results since it was first proposed in 1992.
- Because of QBC's nature of ensembling a committee of models, bagging and boosting methods, which are well-explored for traditional supervised learning, have been proposed in conjunction with QBC [1], [3].
- Under certain asssumptions, QBC has strong theoretical bounds on the number of queries required to achieve a certian level of performance [4].

Cons:

- Sampling from the version space is difficult and randomly sampling from the version space may be costly or infeasible or impossible for some model architectures (ex: gradient descent neural networks often do not achieve the 0 training error required to be a member of the version space).
- Variations of QBC may be used where the committee members do not belong to the version space, but the theoretical bounds on performance do not apply. Here, heuristics similar to our theoretical properties may apply, albeit without formal proof.
- Meaningful bounds for generalization error and performance only exist for certain types of candidate classes/model types.

B. Uncertainty Sampling

In the uncertainty sampling query strategy, unlabeled points are ranked by how certain the current candidate function, $\mathcal{F} : x \in \mathcal{D} \rightarrow label$, is in their classification [10]. The point with the highest uncertainty in its classification is queried.

In order to use uncertainty sampling, the function, $\mathcal{F}$, must not only output predicted classification, but some measure of certainty. This often is realized as the model outputting a probability distribution on the possible classifications. This is true for many types of models including neural networks, probabilistic models, and nearest neighbor classifiers [8]. For example, a neural network which uses soft-max in its output layer directly outputs a probability distribution.

Uncertainty sampling can be viewed as a heuristic for query by committee. The point for which the classifier is most uncertain is often also a point for which there are multiple possible classifications in the version space [4]. Unlike query by committee, uncertainty sampling does not require the ability to randomly draw multiple candidate functions from the version space. Additionally, the one candidate function used in uncertainty sampling does not need to be a member of the version space. This allows uncertainty sampling to be used in situations where it is difficult or impossible to implement query by committee.

In uncertainty sampling, where the model outputs a probability distribution, there are two main methods to select the

most uncertain point. One approach is to select the point whose classification has the highest Shannon entropy:

$$x^* = \arg\max_x \Big(-\sum_i P(y_i|x;\theta) log(P(y_i|x;\theta)) \Big)$$

.

Shannon entropy is a measure for the amount of information or choice in some signal. The higher the entropy is, the more information/choice is present in the signal [5]. Because $\mathcal{F}$ is designed to predict the class of some input, and inputs only belong to one class, higher choice/information in the predicted class indicates high levels of uncertainty.

Another approach is to pick the point for which the highest probability class is the lowest [10]:

$$x^* = \arg\min_x \Big(P(\arg\max_y (P(y|x;\theta))) \Big)$$

. The highest probability class, $\arg\max_y P(y|x;\theta)$, is often used as the predicted classification and a lower probability for the highest probability class indicates that the function, $\mathcal{F}$, is uncertain about the class for the given input.

Uncertainty sampling may also be extended to models that do not output probability distributions. For example, with a support vector machine the point closest to the decision boundary could be selected as the most uncertain point [10].

Pros:

- Uncertainty sampling is commonly applied because it only requires the ability to produce some candidate function and to measure the uncertainty of that function's predictions. This allows uncertainty sampling to be applied to many different types of models.
- Empirically, uncertainty sampling has proven effective in reducing the number of required labeled samples to achieve some performance [4].

Cons:

- Inference needs to be done on every point $\in \mathcal{U}$ to select one point, x, to be queried.
- Uncertain regions may also be less dense regions, causing classifiers to be trained on non-representative data [8].
- No performance bounds like those for query by committee exist.

C. Expected Model Change

Another way to calculate the datapoint with the most utility is to measure which one, if labeled, is expected to change the model the most. Generally, this is done for models optimized with some form of gradient descent. In this method, proposed by Settles et al., the "change" of the model is calculated by measuring the expected value of the length of the training gradient [11]. In essence, the queried datapoint, x, should be the one with with the largest magnitude training gradient if added to $\mathcal{L}$. Consider a model with parameters, θ, a loss function, ℓ, and our labeled training dataset, $\mathcal{L}$. Let $\nabla\ell(\mathcal{L};\theta)$ be the gradient of the loss function, ℓ, with respect to the model parameters, θ. Additionally, let $\nabla\ell(\mathcal{L} \cup \langle x, y\rangle;\theta)$ be the new gradient obtained by adding the datapoint, $\langle x, y\rangle$, to the dataset, $\mathcal{L}$.

Of course, the label, y, for a datapoint, x, is unknown before it is sent oracle, so the gradient is calculated for all potential values of y. The expected value of the norm of this gradient with respect to the potential labels is given by:

$$\mathbb{E}\big(||\nabla\ell(\mathcal{L} \cup \langle x, y\rangle;\theta)||\big) = \sum_i P(y_i|x;\theta)||\nabla\ell(\mathcal{L} \cup \langle x, y_i\rangle;\theta)||$$

The goal in expected model change is, as stated, to maximize the expected value of the training gradient with the addition of a new datapoint. Therefore, the query is given by solving the following optimization problem:

$$x^*_{EMC} = \arg\max_x \mathbb{E}\big(||\nabla\ell(\mathcal{L} \cup \langle x, y\rangle;\theta)||\big)$$

Pros:

- Expected model change is designed to work with gradient descent trained models unlike other query strategies. By calculating the expected model change, we know exactly which datapoints are most likely to move the model most during training.
- Outlined by Settles et al., there are strong empirical metrics supporting success [11].

Cons:

- Expected model change is a very compute-heavy query strategy. Calculating the gradient of the model is much more expensive than, for example, running inference for many types of models. As a result, it can take an overwhelming amount of resources to do this computation for each potential label for every datapoint.
- Expected model change is limited in that it does not have any theoretical bounds confirming a certain level of performance.

D. Expected Error Reduction

The optimal machine learning model achieves the lowest expected error on inputs from the data distribution $\mathcal{D}$. The expected error reduction query strategy picks the query point, $\mathcal{Q}$, such that the expected error of the model trained on $\mathcal{L} \cup \{\mathcal{Q}\}$ is minimized. Although this solution produces a model with minimum generalization error if only one query is performed, this greedy approach may not be optimal when the active learning loop is repeated multiple times and multiple queries are performed [11].

The expected future error of a model where the input space is not finite is given by:

$$\int_{x \in \mathcal{D}} E[Err(\hat{y}, y)|x]p(x)dx$$

where y is the true label for x, $\hat{y}$ is the predicted label for x, and $Err(\hat{y}, y)$ is some error function (This can be modified where y and $\hat{y}$ are probability distributions on the possible classifications and $Err(\hat{y}, y)$ is some error function on probability distributions). [6].

For finite input spaces the integral can be replaced by a summation giving the equation:

$$\sum_{x \in \mathcal{D}} E[Err(\hat{y}, y)|x]p(x)$$

.

Unfortunately, directly evaluating the expected error of a model is difficult or in many cases impossible. Evaluating the expected error requires knowing the data distribution, $\mathcal{D}$, and knowing the true labels for every point in $\mathcal{D}$. Thus the expected error needs to be estimated.

To estimate the expected error multiple approaches have been suggested. One approach is use the unlabeled pool, $\mathcal{U}$ to model the data distribution, $\mathcal{D}$, and to use the current model, to estimate the true class of some $x \in \mathcal{U}$. For models that output probability distributions (a model that outputs only a classification is a probability distribution where the predicted output has probability 1), using the log-loss function as the error function produces the following equation for estimated error:

$$-\sum_{y_i \in \mathcal{C}} P(y_i|x_i) \log(P(y_i|x_i))$$

where $P(y|x)$ gives the predicted probability of x_i having class y_i using the current model, and C is the set of possible classifications [7]. Under these assumptions the estimated error for some $x \in \mathcal{D}$ is calculated as the entropy of the classification distribution given by some predictor $\mathcal{P}$ [7].

Using the unlabeled pool as a representative model of the data distribution, $\mathcal{D}$, the estimated expected error of the model trained on $L^+ = L \cup (x, y)$ is:

$$-\sum_{x_i \in \mathcal{U}} \sum_{y_i \in \mathcal{C}} P_{\mathcal{L}^+}(y_i|x_i) \log(P_{\mathcal{L}^+}(y_i|x_i))$$

where $P_{\mathcal{L}^+}(y|x)$ is the predicted probability of x_i having class y_i with a model trained on $\mathcal{L}^+$.

Of course, the label, y, for a datapoint, x, is unknown before it is sent oracle, so the estimated error reduction is calculated for all potential values of $y \in \mathcal{C}$. The the estimated error reduction for labeling a single point is [7]:

$$\sum_{y \in C} P_{\mathcal{L}}(y|x)\Big(-\sum_{x_i \in \mathcal{U}} \sum_{y_i \in \mathcal{C}} P_{\mathcal{L}^+}(y_i|x_i) \log(P_{\mathcal{L}^+}(y_i|x_i))\Big)$$

The point with minimum expected error is picked yielding the following equation to select a query point:

$$x^* = \arg\min_{x \in \mathcal{U}} \sum_{y \in C} P_{\mathcal{L}}(y|x)\Big(-\sum_{x_i \in \mathcal{U}} \sum_{y_i \in \mathcal{C}} P_{\mathcal{L}^+}(y_i|x_i) \log(P_{\mathcal{L}^+}(y_i|x_i)\Big) \quad (1)$$

where $P_{\mathcal{L}}(y|x)$ is the predicted probability of class y given input x for a classifier trained on $\mathcal{L}$, $\mathcal{L}^+ = L \cup (x, y)$, $P_{\mathcal{L}^+}(y|x)$ is the predicted probability of class y given input x for a classifier trained on $\mathcal{L}^+$, and C is the set of all possible classifications [15]. Entropy is used as a measure of uncertainty, so this query selection's process is equivalent to picking the point, $\mathcal{Q}$, such that the model trained on the set, $\mathcal{L} \cup \mathcal{Q}$, produces predictions where the uncertainty of the model's predictions on the unlabeled set, $\mathcal{U}$ is minimized.

Unfortunately, the predictions from the current model, $P_L(y|x)$, may not be accurate leading to an inaccurate estimation of the expected error of the model. To address this other equations can be used to estimate the expected error of the model. Due to the variety of equations that can be used, discussing specific alternatives is beyond the scope of this survey.

Pros:

- Expected error reduction produces an optimal solution if only a single query can be made (in terms of optimal query point to produce a model with the lowest generalization error), and when expected error reduction is repeated, it is often close to the optimal set of points [11].
- Empirical results show reductions in the number of required labeled datapoints [15].

Cons:

- It is impossible to generally calculate the expected error because it requires knowledge about the data distribution, $\mathcal{D}$, and information about the true classifications for all $x \in \mathcal{D}$ [7].
- Estimates of the expected error are very computationally expensive as they often require computing the potential updated model for possible classifications for all points in the unlabeled pool, $\mathcal{U}$. Additionally inference may need to be done using that updated model for all unlabeled points. This makes it often impractical to calculate estimates of the expected error of the model [10].

E. Batch Sampling with Similarity Consideration

Batch sampling with similarity consideration, while not a query strategy itself, is a technique that builds upon other query strategies that are capable of sampling multiple points (a batch) in a single iteration of the active learning training loop. In other words, most of the above query strategies can be combined with batch sampling with similarity consideration to produce new query strategies. This is worth mentioning because training a model on an updated labeled pool, $\mathcal{L}$, is often a time consuming and expensive process. It is often costly or infeasible to infrequently send only one sample to the oracle which is often a human [12]. There may be high setup costs required to perform an initial query but low costs for additional queries. It may also be infeasible to maintain some experimental setup for long periods of time (such as the periods required to train a model on an updated labeled set). It is common to instead send a batch of samples to be labeled [12].

Many techniques such as uncertainty sampling and expected model change produce a ranking of points based on some measure of utility. A naive approach to produce a batch is to pick the k-highest ranked points to be queried, but performance often decreases as k increases [4, 8]. This occurs because the k-highest ranked points may all be very similar and the queried batch contains large amounts of redundant information. Such naive approaches may perform worse than passive learning because of the redundant information in the labeled batch [12].

There are various approaches that attempt to avoid querying multiple similar points. One possible solution is to introduce a density metric along with the ranking produced by some query strategy. This density metric measures how similar two points are.

One measure of density or how similar points are is the difference in their utility rankings. To reduce density, a minimum ranking distance, G, can be introduced. A batch of size n is produced as follows $\{r_1, r_{1+G}, r_{1+2G}, r_{1+3G}, r_{1+(n-1)G}\}$, where r_i is the i^{th} ranked point [14].

Clustering can also be used to group similar points together, and the highest ranked point from each group can be selected to be queried. This clustering can be performed on meta-features of unlabeled points or on a subset of features of each point as performing clustering on the often high dimensional data that composes a datapoint is costly and time consuming [12], [14].

Other approaches to increase diversity of selected points generalize the utility metric used to rank single samples to a metric that functions on a set of points. For example, expected model change can be calculated for a set of points and potential labels. Unfortunately, because the number of sets of a given size k picked from a set of size n grows grows exponentially in k, it is often computationally intractable to compute a utility metric on all possible subsets directly [12]. A subset of all possible sets of size k picked from the unlabeled pool can instead be evaluated.

Randomized batched is a computationally inexpensive approach to reducing the similarity of selected points in a batch. Some number of randomized batches of size k can be created and the batch with the highest average utility is sent to the oracle [12].

Pros:

- Increases the efficiency of the use of labeling resources [12].
- May reduce costs required to reach some performance level because more points are labeled in each iteration of the active learning loop [12].

Cons:

- Without careful design, performance can be worse than passive learning [12].
- Generally increases the amount of points that need to be labeled to achieve some performance [12], [14].

III. Comparison of Query Strategies

A. Efficiency

The computational efficiency of different query strategies depends on the type of machine learning model being trained. The computational cost for various query strategies for a specific model type, $\mathcal{M}$, will be given in terms of the inference cost, Inference$(\mathcal{M}, x \in \mathcal{D})$, the model update cost when some point is added to the labeled set, Update$(\mathcal{M})$, the number of possible classifications $|\mathcal{C}|$, and the size of the unlabeled set, $|\mathcal{U}|$.

Table 1: Computational cost to select a point to query under different query strategies

Method	Complexity
QBC	$O(\lvert\mathcal{U}\rvert * \text{Inference}(\mathcal{M}, x))$
US	$O(\lvert\mathcal{U}\rvert * (\text{Inference}(\mathcal{M}, x)$ $+\text{Cost to compute uncertainty}))$
EMC	$O(\lvert\mathcal{U}\rvert * \lvert\mathcal{C}\rvert * (\text{Update}(\mathcal{M})$ $+\text{Cost to compute norm of Gradient}))$
EER*	$O(\lvert\mathcal{U}\rvert * \lvert\mathcal{C}\rvert * \text{Update}(\mathcal{M})$ $+\lvert\mathcal{U}\rvert^2 * \lvert\mathcal{C}\rvert * (\text{Inference}(\mathcal{M}, x)$ $+\text{Cost to compute error function})$

*The computational cost for expected error reduction is the cost to compute an estimate of the expected error reduction using equation 1 found in the expected error reduction section. Actually computing the expected error reduction is often infeasible.

Query by committee (QBC) and uncertainty sampling (US) are similar in complexity, as one must run inference on each datapoint in order to determine the query. Expected model change (EMC) has a larger complexity, due to, as noted above, the more difficult calculation of an update relative to running inference. Above all, expected error reduction (EER) has a far larger time complexity. This is due to the complex optimization problem required to select a query.

A benchmarking-paper by Yang and Loog in 2018 compares computation-cost across multiple query strategies for classification tasks across 44 datasets [15]. The authors discussion confirms these theoretical complexities. They cite US as the simplest and cheapest method to implement, followed by expected model change. They emphasize that approaches like EER have far larger computational requirements than other query strategies.

B. Performance

A bench-marking paper by Yang and Loog in 2018 compares performance results across multiple query strategies and classification tasks [15]. Across 44 datasets, multiple active learning methods are compared, the most notable are the uncertainty sampling and expected model change query strategies, which perform best. Below, we've compiled the methods included in this survey as well as a randomly sampled dataset for a baseline.

They begin with a small $\mathcal{L}$, with only two labeled points per class, and continue querying points from $\mathcal{U}$ and adding them to $\mathcal{L}$, until $\mathcal{L}$ reaches the same size as the randomly sampled benchmark. Then they compare results.

Performance Across Selected Query Strategies				
	RS	US	EMC	EER
Avg. Acc.	0.79	**0.81**	0.81	0.80
Avg. Rank	6.86	**3.89**	4.36	5.09
Win Count	2	**14**	8	13
Rank	10	**1**	4	2

The above table includes the following methods: randomly sampling a dataset (RS); uncertainty sampling (US); expected model change (EMC); expected error reduction (EER). The paper used the following high level metrics to compare each

method: mean accuracy across all 44 datasets (Avg. Acc.); avgerage rank in accuracy across datasets (Avg. Rank); number of datasets as the highest performing method (Win Counts); and the rank by highest win count (Rank).

In this survey, uncertainty sampling consistently outperformed all other methods, including ones not listed in the table above. The authors mention how, in spite of its simple criteria, it still outperforms more complex, computationally expensive methods in selecting queries. Since uncertainty sampling was proposed in 1994, they suggest that, perhaps, little progress has really been made since then.

It is also reassuring that *all* selected active learning methods in the original benchmark paper outperform labeling a randomly sampled a dataset. This reaffirms that actively labeling a dataset based on the model at the current time step is better than passively learning with a given dataset.

It is also interesting to note that most active learning techniques performed very similarly. If we just look at the query strategies included in this paper, they all perform within a percentage point of each other. This encourages us to believe a query strategy might be selected largely off of it's efficiency, instead of off it's relative performance.

Note that this comparison did not include query by committee. Query committee is not easily utilized with a committee of neural networks because neural networks are often not in the version space [10], and therefore was out of scope of the benchmarking paper, which used neural networks for logistic regression problems. However, QBC is widely adopted, and, since its proposal in 1992, has one of the largest pools of literature developed of any query strategy covered in this survey.

IV. Conclusion

Active learning can dramatically reduce the amount of labeling needed to produce a model with the desired performance. This can mitigate the costs required to implement a machine learning solution. If the budget is available to label a consant number of points, an active learning approach can improve performance compared to passive learning. In this way, by effectively using query strategies for active learning, instead of passive learning, more can be gotten for one's money.

For most problems (finite VC dimension), almost any passive learning algorithm—where a dataset is labeled based on random sampling—is dominated by some active learning algorithm in terms of labeling complexity. However the correct query strategy must be selected to achieve this dominance. Through a comparison of surveyed query strategies, we've found that uncertainty sampling consistently performs best. However, we also found that many active learning methods perform very similarly, which encourages practitioners to select a query strategy based on efficiency and computational complexity. As a result, the benefits of query by committee and uncertainty sampling are even more attractive to the practitioner.

There are many open areas of research pertaining to query strategies to be explored. Many of these are centered on making active learning more efficient, so more comprehensive query selection can be done without requiring outrageous amounts of resources.

References

[1] L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.

[2] Y. Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. "Information, prediction, and query by committee," Neural Information Processing Systems, 1992.

[3] Y. Freund and R.E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, 1997.

[4] S. Hanneke, "Theoretical foundations of active learning," Carnegie Mellon University, 2009.

[5] D. Lewis, "A sequential algorithm for training text classifiers: Corrigendum and additional data," Acm Sigir Forum, 1995.

[6] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," Twenty-first international conference on Machine learning - ICML '04, 2004.

[7] Roy, Nicholas and Mccallum, Andrew. Toward optimal active learning through sampling estimation of error reduction. In In Proc. 18th International Conf. on Machine Learning, pp. 441–448, 2001.

[8] C.E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27:379–423,623–656, 1948

[9] Kim, Seokhwan, et al., "Mmr-based active machine learning for bio named entity recognition," Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers. 2006.

[10] B. Settles, "Active Learning Literature Survey," University of Wisconsin Madison. [Online]. Available: https://research.cs.wisc.edu/techreports/2009/TR1648.pdf.

[11] B. Settles, M. Craven, and S. Ray, "Multiple-Instance Active Learning," Neural Information Processing Systems, vol. 20, 2007.

[12] B. Settles, "From theories to queries: Active learning in practice," Active learning and experimental design workshop in conjunction with AISTATS 2010. JMLR Workshop and Conference Proceedings, 2011.

[13] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," Proceedings of the fifth annual workshop on Computational learning theory, 1992.

[14] Z. Xu, R. Akella, and Y. Zhang, "Incorporating diversity and density in active learning for relevance feedback," Lecture Notes in Computer Science, pp. 246–257, Apr. 2007.

[15] Y. Yang and M. Loog. "A benchmark and comparison of active learning for logistic regression." Pattern Recognition, 2018.