Noname manuscript No.
(will be inserted by the editor)

A Survey of Active Learning Techniques

Tim Menzies, West Virginia University
Kelly D. Cecil, West Virginia University

the date of receipt and acceptance should be inserted later

1 Introduction

The objective of this literature is to review the topic of Active Learning. We wish to give special attention to techniques that minimize the amount of time required for a user to probe sub-samples of the data to accurately classify code modules as being "defective" or "non-defective". We focus our investigation on sampling techniques to minimize the number of queries required by the classifier and classification methods to maximize the accuracy of the classification.

Address(es) of author(s) should be given

2 Active Learning

2.1 What is Active Learning?

There is a large amount of data available in unlimited formats such as software engineering data. These pieces of data are all unlabeled and labeling all of these data points can be expensive. A heuristic to assist in minimizing the quantity of data points to be labeled would be helpful when classifying data. We introduce *active learning* as a promising solution.

Active learning shares similarities with supervised and unsupervised learning. Active learning and supervised learning techniques both attempt to learn a accurate classifier to fit a data set. Labeling points in active learning and unsupervised learning techniques are expensive and should be minimized. The goal of an active learner should be to obtain an accurate classifier with a lower computational cost than a standard supervised learning algorithm by minimizing the number of queries on data points.

Input: A set of unlabeled data D
Find a clustering of the data
for all cluster $c \in D$ **do**
  Label several data points per a sampling strategy
  Assign each cluster a majority label
end for
Build a classifier using the fully labeled data set

Fig. 1: An Active Learning Heuristic suggested by Dasgupta [1]

2.2 Motivation for Active Learning

Dasgupta describes two ways to explain when active learning is helpful [1]. Active learning can provide means for an efficient search through a hypothesis space. Each request for a label may help to shrink the set of plausible classifiers, and active learning techniques can help to select the points that would help to shrink that set as quickly as possible. This goal of selecting minimal points motivates the investigation of effective sampling techniques to aid in the selection of data points. Active learning can also help to exploit a cluster structure in data. A primitive method of constructing classifiers is to find clusters in the data, label selected data points within the cluster, and assign each cluster to the majority class of the labeled points. The classifier is constructed on the results of these clusters.

2.3 Applying Active Learning to Software Engineering

There are considerations that must be faced when applying an active learning technique to software engineering data. Typical sets of software engineering data are very large, so we must be minimize the complexity of our solution. A solution with algorithms of low complexity is desirable to minimize the user must wait before providing input.

This consideration is very important for interactive data mining applications where the feature set is updated. Jakob Nielsen suggests that a second is the limit for the user's flow of thought to continue uninterrupted and ten seconds is the limit for keeping a user's attention focused on dialogue. Nielsen also states that basic advice regarding these response times have been the same for 30 years. [2] A user who must give input on a set of feature sets may have focused their attention on other topics in the time required for the application to finish.

One focus should be to identify clustering and classifier algorithms that minimize their computation complexity while still performing a satisfactory job in their respective domains. We may also investigate methods of *feature subset selection* to remove attributes that add little value to the classification. Previous investigations with feature subset selection have found that parts of the training data can be removed without sacrificing the effectiveness of the classifier [3]. These investigations have also suggested that simple machine learning algorithms such as Naive Bayes can discover information in defect prediction as well or better than more sophisticated classifiers [3].

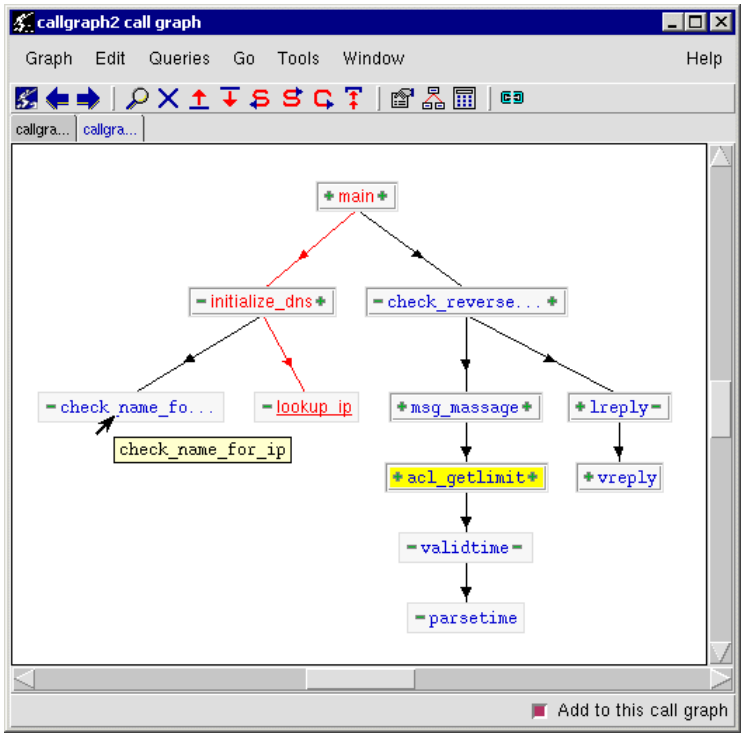

Fig. 2: A function call graph generated in GrammaTech's CodeSurfer source-code analysis tool

We further consider the application of active learning to the typical scientific method: creating a theory, exploration and experimentation of the idea, and formation of a deduction. We may begin with a single hypothesis and apply active learning for

efficient exploration of interesting instances and data points and quickly reduce the hypothesis space. The results of our deduction can further lead to more theories to explore in our domain. Active learning can potentially provide a framework to help us efficiently investigate complex domains including but not limited to software engineering. Consider existing software engineering tools such as GrammaTech's CodeSurfer and CodeSonar that aim to automate source-code analysis. These tools augmented with an active learning framework could allow a user to efficiently investigate source code and interactively update their experiment as their hypothesis evolves.

3 Feature Subset Selection

We now investigate methods of feature subset selection to intelligently reduce attributes in data sets.

3.1 Motivation

We earlier discussed the application of active learning to the field of software engineering. Software engineering data sets are typically large batches being input at once and can be very time consuming to process. Feature subset selection algorithms provide a method to reduce the features of a data set to those that are deemed to be the most valuable to the analysis. This reduction can lead to higher true positive and lower false positive rates while significantly reducing the time required to analyze the data set.

3.2 Principle Component Analysis

Principle Component Analysis is a feature subset selection technique that employees Eigen-vectors to find the most significant features of the data [4].

Principle Component Analysis works as follows [4]. We begin by computing the mean m for each attribute x in the data set D. We subtract m from all values x_i producing a *zero-mean* data set.. We then calculate the covariance matrix of the zero-mean data set. The covariance matrix is then used to calculate the eigen-values and eigen-vectors of the data set. Larger eigen-values indicate strong patterns for a given attribute while smaller eigen-values indicate weaker, less interesting patterns for an attribute.

3.3 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (tf-idf) is a simple feature subset selection technique that traditionally been applied to text document mining but has been applied to search-based software engineering with much success.

```
int distance(point p1, point p2)
{
return sqrt((p1.x − p2.x)^2 + (p1.y − p2.y)^2)
}
```

Fig. 3: An example function [5].

tf-idf begins by extracting terms from source code in our case. Binkley notes that terms are most often extracted from identifiers and comments in the source code. Binkley provides the example function in Figure **??**fig:Function. We extract identifiers **int**, **distance**, **p1**, **p2**, **return**, **sqrt**, **x**, and **y** from the example function.

We begin by calculating the *term frequency (tf)* by counting the frequency of the term t_i in code modules m_j and optionally normalizing by the number of all terms in

m_j to offset a bias in large documents. We calculate the *inverse document frequency (idf)* by dividing the number of modules M by the number of modules containing the t_i and applying a logarithm to the result. We construct the weight vector for our terms by multiplying $tf_{i,j}$ and $idf_{i,j}$.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$
$$idf_i = log \frac{|M|}{\{|m : t_i \in m|\}}$$
$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

Fig. 4: Formulas for tf, idf, and the final tf-idf feature vector.

The weight of a term expresses the importance of the term in a document relevant to the entire collection of documents. A term that occurs rarely in the entire collection of documents but frequently in a module would receive a high weight, while a term that appears frequently in all modules in the collection such as programming language constructs would receive a low weight and could be safely removed from our feature selection.

Table 1 demonstrates Principle Component Analysis and tf-idf with a combination of clustering algorithms. It is worth noting that tf-idf requires less time than Principle Component Analysis for each respective clustering algorithm. tf-idf performs well with K-means and GenIc clustering algorithms when comparing information gain to their Principle Componenent Analysis counterparts.

4 Sampling Techniques

We now take a look at several available techniques to aid the selection of points to label.

4.1 The Problems of Random Sampling

General active learning algorithms call for a random sampling of the data, but random sampling is not an optimal sampling policy for several reasons. Labeling data points is an expensive operation, so it is unwise to label areas where misclassification is not possible. Dasgupta and Hsu state that the number of queries required for a proper analysis when using random queries is $O(\frac{1}{\xi})$ where ξ is the desired error rate. [6] Cohn et al suggests that labeling data points will be most useful in the **region of uncertainty**, an area defined as the region in the sample space where misclassification is still possible based on information already gathered [7]. The cost of labeling in this region is more rewarding than labeling in an area where misclassification is unlikely occur. Our project can benefit from a sampling technique that finds a balance between exploration and exploitation of the dataset.

4.2 Exploration vs. Exploitation

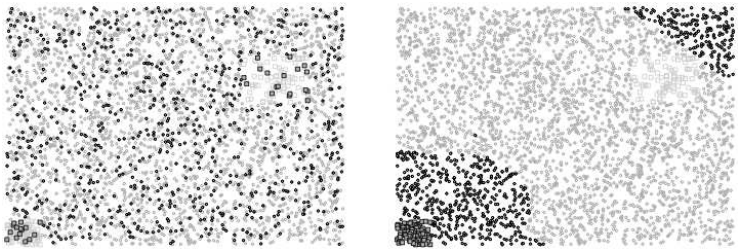

Fig. 5: An random sampling technique favoring exploration (left) vs. a simple sampling algorithm that favors exploitation (right) [8]

An analysis of sampling algorithm will frequently begin to discuss the algorithm's balance between the exploration and exploitation of their selection of data points to be labeled by the expert. Tomanek et. al. describes active learning methods as being a highly exploitative strategy since active learning seeks to thoroughly explore the region of uncertainty defined by Cohn above. The trade off is that the method may not explore regions further away from the region of uncertainty. Wallace and Tomanek both express concerns that clusters may still exist in this unexplored region and refer to this phenomenon as the *missed-cluster effect*. This concern is demonstrated in Figure 5. We see that the random sampling algorithm demonstrated on the left found all regions of interest while requesting labels for many unnecessary points. The simple sampling algorithm demonstrated on the right concentrated requests for labels to the region of

uncertainty but failed to identify a cluster in the unexplored region of the data set. Both authors assert the need for sampling techniques that properly balance exploration and exploitation of a data set.

4.3 Hierarchical Sampling

Dasgupta et. al. present an active learning algorithm shown in Figure6 where they use inherent cluster structures in the data [6]. Their method begins by applying hierarchical clustering on the data. The root of the forming tree is assumed to be the cluster containing all of the instances. Random instances in the tree are queried for their labels and, depending on these labels, statistics for each pruning of the tree to see whether a particular pruning is mixed with different labels or is relatively pure with one class. Labeling can be stopped at any point and the clusters are assigned their labels depending on the majority vote. The algorithm halts when the pruning of the trees is statistically judged to be pure. Their method is statistically consistent and never worse than supervised techniques. When compared to the baseline active learning method of random sampling, it reduces the number of points queried to $O(\sqrt{\frac{1}{\xi}\log\frac{1}{\xi}})$ The success method depends on how well the actual labels align with the cluster structure in the data.

Input: Hierarchical clustering of n unlabeled points; batch size B
$P \leftarrow \{root\}$ (current pruning of the tree)
$L\{root\} \leftarrow 1$ (arbitrary starting label for root)
for time $t = 1$ until the budget runs out **do**
  for $i = 1$ to B **do**
    $v \leftarrow select(p)$
    Pick a random point z from subtree T_v
    Query z's label l
    Update empirical counts and probabilities $(n_u(t), p_{u,l}(t))$ for all nodes u on path from z to v
  end for
  In a bottom-up pass of T, update A and compute scores s(u,t) for all nodes $u \in T$

  for each (selected) $v \in P$ **do**
    Let (P',L') be the pruning and labeling of T_v achieving scores s(v,t)
    $P \leftarrow (P\ \{v\}) \bigcup P'$
    $L(u) \leftarrow L'(u) for all u \in P'$
  end for
end for
for each cluster $v \in P$ **do**
  Assign each point in T_v the label $L(v)$
end for

Fig. 6: Hierarchical Sampling Algorithm [6]

4.4 Greedy Sampling

Dasgupta suggests a greedy approach that makes use of Vapnik-Chervonenkis (VC) theory to selecting data points to be labeled [9]. VC theory attempts to use a statistical point of view to describe learning processes. The greedy strategy selects data points to label by choosing points that are equally weighted as a positive and negative point. Dasgupta formally defines this as follows:

> Let $S \subseteq \hat{H}$ be the current version space. For each unlabeled x_i, let S_i^+ be the hypotheses which label x_i positive and S_i^- the ones which label it negative. Pick the x_i for which these sets are most nearly equal in π-mass, that is, for which $\min\{\pi(S_i^2,\pi(S_i^+)\}$ is largest [9].

The greedy choice made by his approach, to always ask for the label which most evenly divides the space, satisfies the greedy choice property. This does not necessarily minimize the points that must be labeled but typically requires at most O(ln $\mid H \mid$, where H is the hypothesis space. Dasgupta's binary search of the space requires $dlogm$ labels where d is the VC dimension and m is the total number of data points. The VC dimension is defined as a measure of the largest data set of cardinality h that can be shattered (the positive and negative hypotheses are separated) by classification model f.

4.5 Agnostic Sampling

Balcan et. al.'s A^2 agnostic sampling algorithm shown in Figure 7 is the first to be proposed as an active learning algorithm that works in the presence of any form of (limited) noise. The assumption of the algorithm is that samples come drawn from a fixed distribution. A^2 (Agnostic Active) is proposed to be a robust version of the selective sampling proposed by Cohn et. al. The difference of A^2 to selective sampling is the agnostic approach taken by the algorithm that does not eliminate a hypothesis based on the result of a single label query. The authors show that A^2 achieves exponential speedups in several settings previously analyzed without a noise. A^2 performs particularly well for the simple case of learning threshold functions (this holds for arbitrary distributions as well, provided the noise rate is low). The expected number of queries required for the algorithm is $O(\frac{\eta^2 ln\frac{1}{\delta}}{\xi^2})$ where ξ is the desired error rate, η is the noise rate, and ξ is a confidence parameter.

4.6 Query by Committee

The object of the Query By Committee (QBC) is to use quick multiple prediction methods (the committee) to guess a label for each data point. The data point is identified to be labeled when the committee is unable to find a consensus regarding the data points label. The intention of the algorithm is to filter the most informative examples from the set of data points. We discuss Fruend's algorithm for Query By Committee below.

The algorithm begins by calling an oracle **Sample** to retrieve an unlabeled data point x from the set of data points. The algorithm proceeds to call prediction methods

$i = 1$, $D_i = D$, H_i, $S_i = \phi$, and $k = 1$

$DISAGREE_D(H_i) = \mathbf{Pr}_{x\ D}[\exists h_1,\ h_2 \in G : h_1(x) \neq h_2(x)]$

while $DISAGREE_D(H_i)(\ min_{h \in H_i}\ \mathrm{UB}(S_i,\ h,\ \delta_k)$ - $min_{h \in H_i}\ \mathrm{LB}(S_i,h,\delta_k)) > \varepsilon$
do
  Set $S_i = \phi$, $H_i' = H_i$, $k \leftarrow k + 1$
  while $DISAGREE_D(H_i) \geq \frac{1}{2} DISAGREE_D(H_i)$ **do**
    if $DISAGREE_D(H_i')(min_{h \in H_i'}\ \mathrm{UB}(S_i,\ h,\ \delta_k)$- $min_{h \in H]_i}\ \mathrm{LB}(S_i,h,\delta_k))) \leq \varepsilon$
    then
      return h = $argmin_h\ in H_i'\ \mathrm{UB}(S_i,\ h,\ \delta_k)$.
    else
      S_i' = Rejection Sample $2|S_i| + 1$ samples x from D satisfying: $\exists h_1,h_2 \in H_i$: $h_1(x) \neq h_2(x)$
      $S_i \leftarrow S_i \bigcup \{(x,\mathrm{O}(x)) : x \in S_i'\}$; $k \leftarrow k + 1$;
      $H_i' = \{h \in H_i : \mathrm{LB}(S_i,h,\delta_k)) \leq min_{h' \in H_i}\ \mathrm{UB}(S_i,\ h,\ \delta_k)\}$, $k \leftarrow k + 1$.
    end if
  end while
  $H_{i+1} \leftarrow H_i'$;
  $D_{i+1} \leftarrow D_i$ conditioned on the disagreement $\exists h_1,h_2 \in H_i : h_1(x) \neq h_2(x)$;
  $i \leftarrow i + 1$.
end while
return $h = argmin_{h \in H_i}\ \mathrm{UB}(S_i,\ h,\ \delta_k)$

Fig. 7: A^2 Agnostic Sampling algorithm [10]

Note:
$\varepsilon > 0$ - the maximal prediction error
$\delta > 0$ - the desired reliability
Gibbs - an oracle that computes Gibbs predictions
Sample - an oracle that generates unlabeled examples
Label - an oracle (or expert) that generates the correct label of an example

Set: $n = 0$ - the counter of labels to Label
$V_0 = C$ - where V is the version space and C is the complete concept class

repeat
  Call **Sample** to get an unlabeled example $x \in X$ drawn at random
  Call **Gibbs**(V_n,x) twice to get two predictions for the label of x
  if the two predictions are equal **then**
    Reject the example x and return to the beginning of the loop
  else
    Call **Label**(x to get $c(x)$), increase n by 1, and get V_n to be all concepts $c' \in V_{n-1}$ such that $c'(x) = c(x)$
  end if
until More than t_n consecutive samples are rejected
return the Prediction Hypothesis **Gibbs**(V_n,x).

Fig. 8: Query by Committee algorithm for a committee of two [11]

to quickly guess the label of x. Freund defines a prediction method **Gibbs** that randomly chooses a hypothesis h from the possible classes (or labels). The data point is rejected for labeling if the committee of prediction methods agree on a label. The data point is labeled by the oracle **Label** if the committee disagrees on the label.

4.7 Iterative Approach to Selective Sampling

Warmuth applies an iterative approach to selectively sampling previously unlabeled data for finding molecules that will bind with a particular molecule [12]. 5% batches of the data are first chosen at random until positive and negative batches are found. All further batches are chosen using several sampling strategies detailed below. Positive is the goal state for a batch since those compounds will bind. Labels were hidden from learn in order to simulate the experiment and performance is benchmarked by reporting false positives. The sampling strategies were as follows:

- **Nearest Neighbor:** The data is normalized and batches are considered for being closest to the already labeled batches' hyperplanes.
- **Support Vector Machine (SVM):** The batch is selected by choosing the maximum margin hyperplane produced by a support vector machine.
- **Voted Perceptron (VoPerc):** The strategy computes the weight vector of each example and stores these vectors. The vectors recorded in each pass are recorded for each 100 random permutations of the labeled examples. Each weight vector is assigned a $\pm$ vote where the prediction on an example is positive if the resultant vote is greater than zero and negative otherwise. Unlabeled examples are chosen that are closest to zero.

All three sampling strategies performed similarly when comparing true and false positives their respective strategies. The authors declare VoPerc as the best performing technique due to having a smaller variance in true and false positives than the other two techniques.

5 Clustering Methods

5.1 Motivation

The previous section discussed various sampling methods for choosing the most useful queries in an active learning system and their strengths (situations in which they might excel) and weaknesses (queries required for a proper analysis.) We would clearly benefit from using clustering techniques to identify areas of interest and ensure that we further achieve the goal of balancing exploration and exploitation. Our ideal clustering algorithm would have low asymptotic complexity while providing a optimal grouping of similar data points.

5.2 k-means

Input: D - Data points to be clustered, n - number of data points in D, k - number of clusters
Initialize: A list of cluster centroids $C_{1..k}$ to random initial data points, m representing a list of cluster memberships
for each data point $d_i \in D$ **do**
  $m(p_i) = \underset{k \in \{1..n\}}{argmin}\ \text{distance}(p_i, c_k)$
end for
while cluster membership m has changed **do**
  for each $c_i \in C$ **do**
    $c_i = \frac{1}{|m_i|} \sum_{x_j \in m_i} x_j$
  end for
  for each data point $d_i \in D$ **do**
    $m(p_i) = \underset{k \in \{1 to n\}}{argmin}\ \text{distance}(p_i, c_k)$
  end for
end while

Fig. 9: Pseudo-code for K-means Algorithm

K-means is a clustering algorithm that seeks to partition points into k clusters with the nearest mean point. The k-means algorithm has an asymptotic complexity of O(kN) where k is the number of clusters and N is the number of data points in the input set. K-means has an attractive complexity, but it is important to note that the algorithm may have to be executed multiple times to find the best value of k for the given data.

5.3 GenIc

GenIc (GENeralized Incremental Algorithm for Clustering) is a clustering algorithm developed specifically to analyze streaming data. The assumption is that streaming data (data that is arriving in real time) can only be analyzed once, so GenIc has a low time complexity of $O(n)$ where n is the number of input data records [13] GenIc

Input: k - number of centers; m number of initial points, n - size of a generation
for $i = 1tom$ **do**
  Select a point c_i to be an candidate center in the list of centers L
  Assign a weight of w_i to the candidate center
end for
for each data point p in the stream **do**
  Count = Count + 1
  Find the nearest candidate center c_i to the point p
  Move the nearest candidate center using formula: $c_i = \frac{(w_i * c_i + P)}{(w_i + 1)}$
  Increment the weight: $w_i = w_i + 1$
  if $Count \equiv 0 \bmod n$ **then**
    for each center c_i in the list L **do**
      Calculate the probability of survival using the formula: $p_i = \frac{w_i}{\sum_{i=1}^{m} w_i}$
      Select a random number δ uniformly from [0,1].
      if $p_i > \delta$ **then**
        Retain the c_i in L for use in the next generation of n points
      else if $p_i < \delta$ **then**
        Remove c_i from L and choose a new point from the current generation as a replacement
      end if
    end for
  end if
end for
Group the final m centers from list L into the final k centers based on their Euclidean distances

Fig. 10: Pseudo-code for GenIc Algorithm [13]

is also an incremental algorithm meaning that the clusters can be updated as sets of data points referred to as **generations** arrive over time.

GenIc computes the "fitness" of each each center as a function of the number of points assigned to the center. The algorithm computes the probability of the center's survival10 and randomly selects a value between 0 and 1 from a uniform distribution to determine if a center is removed. The fittest generally survive to the next generation, but old centers are occasionally replaced with new centers.

5.4 Canopy

The canopy clustering technique is a technique that addresses situations where there can be a large number of points in the data set, each point can have many features, and there can be many clusters to discover in the data set. The canopy seeks to solve all three problems by dividing the clustering into a two stage process.

The first stage uses some cheap distance or similarity measure to create overlapping subsets called "canopies." Canopies are defined to be a subset of points that are calculated by some estimated distance or similarity measure to be within some distance threshold from a central point [14]. Points may reside in multiple canopies (canopies may overlap), but all points must reside in at least one canopy. An assumption is made that points that are not in the same canopy can not be in the same cluster.

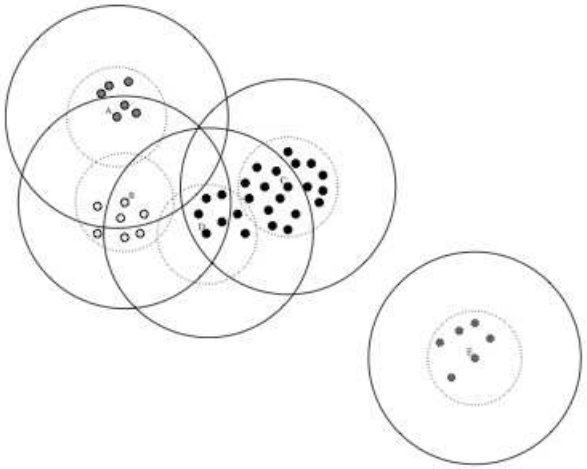

Fig. 11: An example of five canopies [14]

Figure 11 illustrates an example provided by McCullen et. al. [**?**] of five clusters and their canopies. A solid circle indicates the outer thresholds of the canopy. Points labeled A, B, C, D, and E are the center points of the canopies. The doted circles within each canopy designate an area is excluded for consideration to be the center of a new canopy.

The second stage uses a traditional clustering algorithm such as K-means using an accurate distance or similarity measure with the restriction that we only measure distances between points that reside in the same canopy (and assume points of different canopies to have an infinite distance).

McCullen states that the canopy technique requires at most $O(\frac{f^2n^2}{c})$ distance measures where n is the number of data points, f is the average number of canopies in which each data point resides, and c is the number of canopies in use. McCullen also states that f will generally be much less than c. Canopy with K-means as the second stage clustering technique requires $O(\frac{n*k*f^2}{c})$ per iteration [**?**].

5.5 Compass

Compass is a clustering algorithm that works by finding two points with a maximum distance from the other in the data set and build a tree structure by pulling data to each of the two extreme points to form two sets. We then recurse on each set until the variance between the layers is less than a determined threshold. The Compass algorithm is very simple, and thus is very quick. The algorithm's complexity is O(3LN) where L is the number of layers and N is the number of input points. Further research is in progress to investigate the inclusion of early stopping rules in the algorithm to reduce the number of layers of the tree.

Input: d - Set of data points
Initialize: $left$,$right$ as an empty set

Let c be a random point in d
Find the point $left_0 \in d$ that is maximum distance from c
Find the point $right_0 \in d$ that is maximum distance from $left_0$
for each point $p_i \in d$ **do**
  if distance(p_i,$left_0$) < distance(p_i,$right_0$) **then**
    Add p_i to $left$
  else
    Add p_i to $right$
  end if
end for
if $\Delta(d, \{left, right\}) < \Delta_{threshold}$ **then**
  Recurse on $left$ and $right$
else
  Return d as a cluster
end if

Fig. 12: Pseudo-code for Compass Clustering Algorithm

5.6 Comparison of Clustering Techniques

A fast clustering algorithm has several advantages over a slower clustering algorithm such as Greedy Agglomerative Clustering (GAC) which promises a computational complexity of $O(n^2)$. GAC and other clustering algorithms with large computational complexities do not scale well to large data sets. We attempt to investigate clustering algorithms that minimize the computational complexity while still providing an acceptable clustering of the data set.

k-Means can provide a more reasonable time investment for a single execution of the algorithm, but the algorithm may be ran multiple times in order to find the best number of k clusters to fit the data set. The worst case time of O(3LN) and ability to be updated incrementally makes Compass a strong candidate with respect to computational complexity.

Reducer and Clusterer	Time	InterSim	IntraSim	Gain
TF-IDF*K-means	17.52	-0.085	141.73	141.82
TF-IDF*GenIc	3.75	-0.14	141.22	141.36
PCA*K-means	100.0	0.0	100.0	100.0
PCA*Canopy	117.49	0.0	99.87	99.87
PCA*GenIc	11.71	-0.07	99.74	99.81
TF-IDF*Canopy	6.58	5.02	93.42	88.4

Table 1 Results from a text mining experiment comparing combinations of feature subset reduction and clustering algorithms to more sophisticated methods.

We find in our studies that the Compass clustering algorithm performs significantly better in most cases than k-means and bisection methods. 2 and 3 provide a Win-Tie-

Method	Win	Loss	Tie
COMPASS	180	0	0
BestK	100	46	34
K=16	157	20	3
K=8	142	25	13
K=4	115	40	25
K=2	71	70	39
K=1	62	82	36
BISECT4	1	76	103
BISECT6	30	44	106
BISECT8	36	39	105

Table 2 Win-Loss-Tie Results for the COCOMO81 Data Set

Method	Win	Loss	Tie
COMPASS	171	8	1
BestK	109	7	64
K=16	125	1	54
K=8	113	6	61
K=4	77	13	90
K=2	162	0	18
K=1	79	10	91
BISECT4	0	178	2
BISECT6	31	85	64
BISECT8	29	86	65

Table 3 Win-Loss-Tie Results for the Maxwell Data Set

Loss (13) comparison of three clustering methods with varying k clusters based on the magnitude of the relative error (MRE) [15]:

$$MRE = \frac{|actual_i - predicted_i|}{actual_i}$$

Foss et al. has demonstrated that using only MRE can lead to an incorrect evaluation of the results in certain situations [15]. Wilcoxon statistics were applied to ensure that these results do not fall into the limitations of MRE. Compass performs significantly better than K-means and bisecting K-means on both data sets. Compass is a very promising algorithm that can provide a small computational complexity while delivering high quality clusters.

$win_i = 0$,$tie_i = 0$, $loss_i = 0$
$win_j = 0$,$tie_j = 0$, $loss_j = 0$
if WILCOXON($MRE's_i$,$MRE's_j$) says they are the same **then**
  $tie_i = tie_i + 1$
  $tie_j = tie_j + 1$
else
  if median($MRE's_i$) ¡ median($MRE's_j$) **then**
    $win_i = win_i + 1$
    $loss_j = loss_j + 1$
  else
    $win_j = win_j + 1$
    $loss_i = loss_i + 1$
  end if
end if

Fig. 13: Pseudo-code for Win-Tie-Loss calculation between variants i and j

6 Domain Experts

We investigate techniques for machine learning techniques to act as our domain expert.

6.1 Motivation

Compton et al. defines an expert as people whose expertise is scarce and valuable. [16] A human expert can often be difficult to find. Humans are also prone to error and are slow when analyzing large amounts of batch data (i.e. software engineering data). An expert may also possess a bias or may specialize in different strategies than other experts [17]. An active learning scheme may benefit from the input of multiple experts who are capable of processing large amounts of data quickly.

6.2 The Use of Simulated Experts in Evaluating Knowledge Acquisition

A machine learning algorithm may be used to stand in for a domain expert for cases when a human domain expert is unavailable. The machine learning algorithm is not an ideal substitute of a living, breathing expert, but the simulated expert could improve the knowledge acquisition process. It is important to note that the expert is only as good as the machine learning algorithm governing it's "opinion" of the data and using only a single expert may not allow the data to be fully explored. Compton [16] and Shaw [17] both suggest the use of multiple experts that can correspond on a problem and "brain-storm" to fully explore their domain of expertise.

6.3 Comparing Conceptual Structures Between Experts

Our experts must be able to communicate with each other if they are to discover on what topics they agree or disagree. Shaw expresses the need for experts to be able to communicate their knowledge of concepts with each other in order to come

to a consensus [17]. The author establishes four relationships that can occur between experts:

- **Consensus:** Experts use the terminology and concepts in the same manner.
- **Correspondence:** Experts use different terminology to describe the same concepts.
- **Conflict:** Experts use the same terminology to describe different concepts.
- **Contrast:** Experts differ in their knowledge of both terminology and concepts.

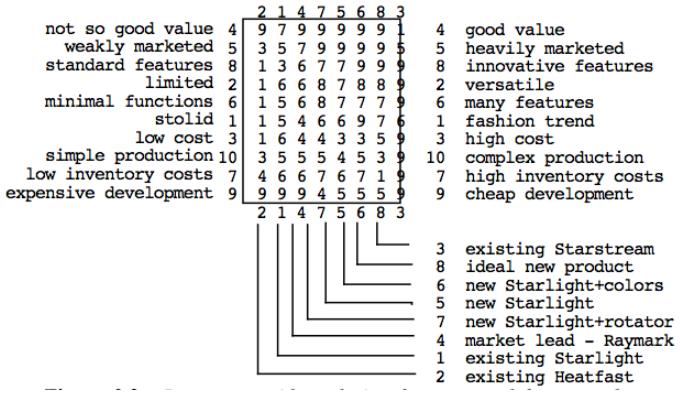

Fig. 14: An example of a reperatory (entity-attribute) grid [17]

We can use a reperatory grid (referred to as an entity-attribute grid in Shaw's literature) as a means of establishing an expert's knowledge on a topic. Figure 14 shows an example of a reperatory grid. Experts can exchange these reperatory grids to develop a difference grid (shown in Figure 15) to easily establish consensus and conflict between the experts. This difference grid populated by scores of 0 (denoted by blank grid locations) to 5 with lower scores indicating a consensus and higher scores indicating conflicts. Our example shows that our experts agree on concepts in the upper right corner of the grid and show conflict on concepts in the lower left corner of the grid.

6.4 ID3 and C4.5

C4.5 is a decision tree machine learning algorithm developed by Ross Quinlan that is an extension of the ID3 algorithm that addresses several shortcomings of ID3.

The ID3 algorithm begins by choosing a subset of the training data set at random. Quinlan refers to this subset of data as the *window* [18]. A decision tree is formed that correctly classifies all objects in the window. The algorithm attempts to classify the rest of the training set using the generated decision tree. If the algorithm correctly classifies the rest of the training set, the algorithm terminates; Otherwise, the algorithm adds a selection of incorrectly classified data to the *window* and regenerates the decision tree. 16 shows an example data set, and Figure 17 shows the resulting decision tree produced by ID3.

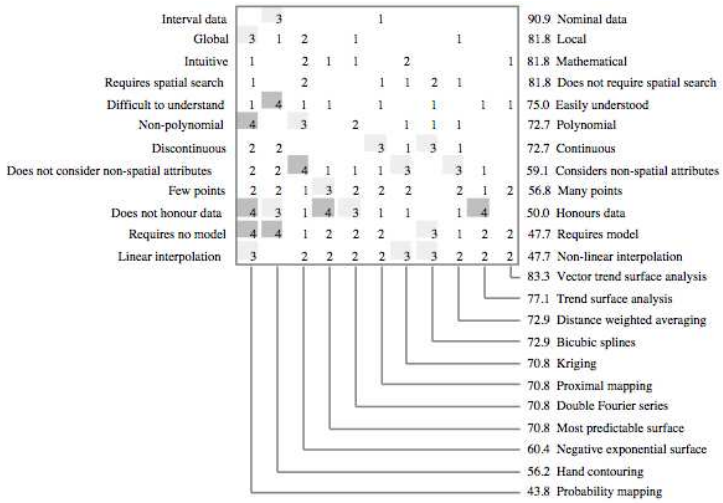

Fig. 15: An example of a difference grid [17]

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Fig. 16: An example data set to be processed by ID3 [18]

C4.5 extends ID3 by adding the ability to deal with missing data and numeric data and provides pruning of the decision tree.

6.5 Naive Bayes

Naive Bayes is a simple statistics-based learning method that is based on Bayes' theorem. Given a class variable C and a list of features $f_1..f_n$):

$p(C|f_1..f_n) = \frac{p(C)*}{p(f_1..f_n|C)}p(f_1..f_n)$

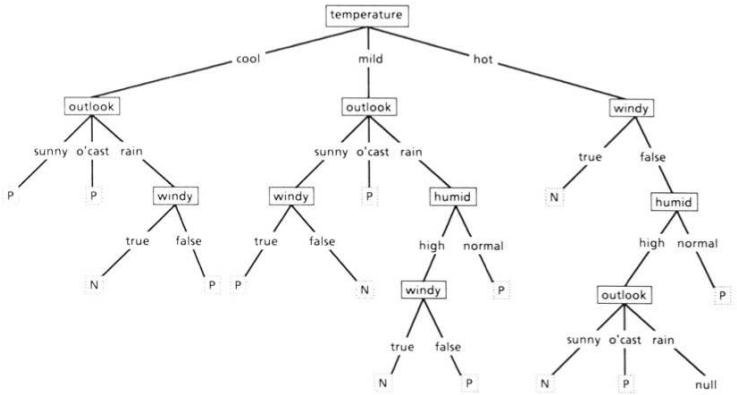

Fig. 17: An example decision tree produced by the ID3 algorithm [18]

The Naive Bayes algorithm is simple to use in practice, very efficient, and robust against noisy data [19]. The algorithm may be used on raw numeric data using a Gaussian probability density function or on discrete data. A limitation of the Naive Bayes classifier is the assumption of feature independence utilized by the classifier is often violated in the real world. The Naive Bayes classifier works very well despite this limitation.

Measure	Bayes	Gauss	C4.5	PEBLS	CN2
No. wins	-	12-7	19-9	16-11	20-8
No. signif. wins	-	6-5	12-8	12-6	16-6
Sign test	-	75.0	96.0	75.0	98.0
Wilcoxon test	-	70.0	96.0	94.0	99.8
Average	79.1	77.8	77.2	77.6	76.2
Rank	2.43	2.75	3.14	3.21	3.46

Fig. 18: Comparison of Naive Bayes to other algorithms [20]

Testing by Pedro Domingos has found that Naive Bayes performs better than other classification algorithms including C4.5 [20]. Figure 18 shows Domingos' results comparing Naive Bayes with discretized data (Bayes), Naive Bayes using a Gaussian probability function for the raw numeric data (Gauss), and several other algorithms. Domingos ranked each classifier using 1 to indicate the most accurate algorithm and increasing as accuracy decreases. Naives Bayes with discrete data performs best while Naive Bayes using the Gaussian probability density function ranked second best.

7 Summary

We have investigated the heuristics of an active learning technique and hypothesized about it's application to complex domains with much focus on software engineering.

Feature subset selection provides us with the means to reduce features in a data set to those that are most interesting to the data set. Principle Component Analysis is a feature subset selection algorithm that uses Eigen-Vectors to find the most significant features in the data. We also present the Term Frequency-Inverse Document Frequency algorithm. The algorithm counts the frequency of a term in a document and scales by the inverse term frequency to produce a weight describing the importance of a term within a document relevent to the entire collection of documents.

Sampling methods are very important to selecting data points. Random sampling is technique that heavily favors exploration and will label points that will add little information to the active learner. We look at techniques that favor exploitation to query points that are in the **region of uncertainty**. Hierarchical sampling applies hierarchical clustering to create a tree structure and applies statistical methods to various pruning of the trees to judge if a tree is pure (containing one class). Greedy sampling applies Vapnik-Chervonenkis theory to choose the most attractive points that are almost equally labeled with positive and negative labels. A^2 agnostic sampling algorithm provides a solution to noise in the data by ensuring that a hypothesis will not be removed from the hypothesis space because of a single point. Query by Committee uses a committee of simple prediction methods to collectively guess the label of a point and send the point to an expert if they disagree on the label. Finally, Warmuth suggests iterative approaches to selective sampling using nearest neighbor, support vector machine, and voted perceptron techniques and found that all three work similarly well with voted perceptron providing slightly less variance in true and false positives than the other two techniques.

Clustering methods allow the active learner to identify areas of interest to further promote exploration and exploitation of the data set. K-means is an general use O(kN) clustering algorithm that updates the mean point of clusters as the cluster membership changes. GenIc is a clustering algorithm that computes the "fitness" of a cluster to determine if the cluster survives to the next generation. GenIc is designed for streaming data and assumes that data can only be analyzed once; GenIc boosts a O(n) complexity to facilitate this streaming data. Canopy is a clustering technique that introduces "canopies" to ensure that traditional clustering techniques only consider points within a distance threshold. Canopy considerably reduces the complexity of clustering algorithms; For example, K-means with canopies requires $O(\frac{n*k*f^2}{c})$ where f is usually much less than c. Compass is a simple O(3LN) algorithm that works by finding two points of maximum distance from each other and constructs a tree by separating data points between the two points based on a distance measure. The algorithm then recurses on the both data sets until the variance between layers is less (uninteresting) or greater (too chaotic) than a variance threshold. We present results that suggest that simple feature subset selection and clustering methods perform as well as or better than more complicated methods. We also present evidence that the Compass clustering algorithm performs considerably better than K-means and Bisecting K-means for various k in several datasets from the PROMISE data repository.

We rely on *Domain Experts* to label interesting data points. Human experts are scarce and are prone to error when analyzing large batch sets of data (such as software

engineering data). We introduce the possibility of using a machine learning algorithm to stand in for a human domain expert and establish a method for experts to communicate their opinions on their domain and terminology with one another. We introduce C4.5 and Naive Bayes machine learning algorithms as possible candidates for experts. C4.5 is a decision tree learning algorithm that trains by continuously adds portions of a training set until the entire training set can be correctly classified. Naives Bayes is a statistical learning algorithm based on Bayes' theorem. We present results by Domingos that suggests that Naive Bayes outperforms other learners such as C4.5 in practice.

References

1. S. Dasgupta and J. Langford, "Tutorial summary: Active learning," in *ICML*, p. 178, 2009.
2. J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.
3. T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang, "Implications of ceiling effects in defect predictors," in *PROMISE '08: Proceedings of the 4th international workshop on Predictor models in software engineering*, (New York, NY, USA), pp. 47–54, ACM, 2008.
4. L. I. Smith, "A tutorial on principal component analysis.".
5. D. Binkley and D. Lawrie, "Information retrieval applications in software maintenance and evolution," in *In Encyclopedia of Software Engineering (P. Laplante, ed.)*, Taylor and Francis LLC, 2010.
6. S. Dasgupta and D. Hsu, "Hierarchical sampling for active learning," in *ICML '08: Proceedings of the 25th international conference on Machine learning*, (New York, NY, USA), pp. 208–215, ACM, 2008.
7. D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.
8. B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos, "Active learning for biomedical citation screening," in *Knowledge Discovery and Data Mining (KDD)*, 2010.
9. S. Dasgupta, "Analysis of a greedy active learning strategy," in *In Advances in Neural Information Processing Systems*, pp. 337–344, MIT Press, 2004.
10. M.-F. Balcan, A. Beygelzimer, and J. Langford, "Agnostic active learning," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, (New York, NY, USA), pp. 65–72, ACM, 2006.
11. Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Mach. Learn.*, vol. 28, no. 2-3, pp. 133–168, 1997.
12. M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen, "Active learning in the drug discovery process," 2002.
13. C. Gupta and R. Grossman, "Genic: A single pass generalized incremental algorithm for clustering," in *In SIAM Int. Conf. on Data Mining*, SIAM, 2004.
14. A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 169–178, ACM, 2000.
15. T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, vol. vol, pp. 29no11pp985–995, 2003.
16. P. Compton, P. Preston, and B. Kang, "The use of simulated experts in evaluating knowledge acquisition," in *University of Calgary*, pp. 12–1, 1995.
17. M. Shaw and B. R. Gaines, "Comparing conceptual structures: Consensus, conflict, correspondence and contrast," *Knowledge Acquisition*, vol. 1, pp. 341–363, 1989.
18. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, March 1986.
19. Y. Yang and G. Webb, "Weighted proprotional k-interval discretization of naive-bayes classifiers," in *PAKADD'03*, pp. 501–512, 2003.
20. P. Domingos and M. J. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

List of Tables

List of Figures