An Implementation of Hierarchical Sampling for Active Learning

Daniel L Chen
University of Michigan
`dalchen@umich.edu`

Louis Goedker
University of Michigan
`lgoedker@umich.edu`

Juyoung Jung
University of Michigan
`juyoungj@umich.edu`

Chung Hoon Hong
University of Michigan
`datahong@umich.edu`

Abstract

In many semi-supervised learning problems, the labels of a portion of unlabeled data need to be revealed in order to augment the existing model that uses the labeled training data. Active learning selects the most informative unlabeled samples that best help construct the learning model. However, active learning has a sampling bias caused by randomness when choosing samples. Hierarchical sampling reduces this problem by exploiting cluster structure in the data. It saves the cost of revealing too many labels and obtains better accuracy in classification. We implemented the hierarchical sampling for active learning algorithm and applied that to different data sets, and compared the points sampled by other baseline sampling methods (random sampling and margin sampling). Our results demonstrate that hierarchical sampling performs better than the other baseline sampling methods when needing to reveal relatively few data points, which aligns with the problem motivation of labeling a few very informative samples.

The public repository for the code implemented in this paper is *here*

1 Problem Statement and Motivation

1.1 Introduction

Active learning is a learning method that is motivated in which a large amount of unlabeled data is available, but it is too costly to obtain their labels. Active learning chooses a few unlabeled points and queries those labels. Since the labels are hidden and can be revealed at a cost, the goal is to obtain an accurate classifier with the least amount of cost by querying just a few points that are the most informative about the decision boundary. Due to the underlying structure of the sample space, active learning intrinsically brings sampling bias problem to the system, and this is most apparent where the data form clusters.

1.2 Sampling Bias Problem

Sampling bias is the essential challenge active learning confronts. In the effort of querying points with the least amount of cost to reveal unlabeled data, active learning methods do not manage to solve the bias introduced during sampling. For example, while sampling proceeds during training, sample points are chosen from its previous information, and the training set diverges from the distribution.

1.3 Motivation

The goal of this paper is to check whether hierarchical sampling solves the sampling bias problem when dealing with clustered data. We implement the hierarchical sampling for active learning algorithm and apply that to different datas ets, and compare the points sampled by other baseline sampling methods (random sampling and margin sampling). The quality of the sampled points is compared by constructing a logistic regression model for the various data sets and comparing the induced error.

2 Literature Review and Related Work

Selective Sampling

Selective Sampling [1] is an elementary version of active learning. The method selectively filters informative queries from arbitrary inputs. It performs significantly better than the random sampling method. The prediction error decreases as the number of queries increase. *Freund et al. 1997*

Agnostic Active Learning

Agnostic Active Learning [2] focuses on the situations where there is the presence of arbitrary forms of noise. It can find the optimal separation between the classes with a sizable conditional noise rate.

Margin Based Active Learning

Margin Based Active Learning [3] presents a framework for margin based active learning of linear separators. It showed that the algorithm and argument extended to the non-realizable case. It also analyzed a "large margin" setting and showed that active learning improved sample complexity. The margin sampling algorithm in section 3 was directly drawn from this paper.

Minimax Bounds for Active Learning

Minimax Bounds for Active Learning [4] achieve limits in active learning for broad classes of distributions characterized by decision boundary regularity and noise conditions. Using minimax analysis techniques, it results in significant gains in active learning.

Hierarchical Sampling for Active Learning

Hierarchical Sampling for Active Learning [5] exploits clusters of data to find the most informative unlabeled samples to label. This paper serves as the basis of our project. Our project is an implementation and augmentation of the algorithm described in this paper.

3 Methodology

In this section, we discuss the implementation details of the hierarchical sampling. We also discuss the implementation of the baseline models that hierarchical sampling is compared to (random sampling and marginal sampling).

3.1 Random Sampling

Random sampling is the simplest method of sampling from the uncertain factor space. In a random sampling, each sample is chosen with equal probability. Thus, randomly selected sample is meant to be an unbiased representation of the total population. There are two different methods of random sampling: one without replacement and one with replacement. In the case of without replacement, sample once chosen randomly cannot be chosen again. On the other hand, in the case of with replacement, sample once chosen can be chosen again.

The probability without replacement:

$$P = 1 - \frac{N-1}{N} \cdot \frac{N-2}{N-1} \cdot \dots \cdot \frac{N-n}{N-\big(n-1\big)} \tag{1}$$

The probability with replacement:

$$P = 1 - \left(1 - \frac{1}{N}\right)^n \tag{2}$$

Our implementation uses random sampling without replacement. This is because if the sample has been selected to reveal its label for, it should not be considered unlabeled anymore.

3.2 Margin Sampling

Margin sampling picks the example with the smallest margin. c and c' represent the most likely categories. It selects the unlabeled data that is most likely to become new support vectors. Those unlabeled data lie within the margin of the current support vector machine. When multiple samples are chosen simultaneously, this method does not consider distribution and structural information in the feature space, making the method only applicable to the case when a single example is selected at each iteration. Thus, choosing several samples simultaneously would decrease the classification accuracy with data redundancy. Since marginal sampling does not consider the distribution and the structural space connectivity among the unlabeled data, selecting multiple data simultaneously may lead to choosing data points that share the same information lying close to the hyperplane, which results in oversampling on dense regions. Equation 3 below calculates the margin.

$$M_n = |Pr(c|x_n) - Pr(c'|x_n)| \tag{3}$$

3.3 Hierarchical Sampling

Intuition

Hierarchical sampling exploits the natural clustering of the data and methodically draws samples to find a sound representation of the clusters of data. Hierarchical sampling relies on a hierarchical clustering procedure that produces a tree structure, **where each leaf of the tree represents a sample**. A sample is drawn by traversing some path down the tree. The path traversed is influenced by a weight associated with each node in the tree, which in turn will influence which sample is chosen. Since hierarchical sampling assumes the existence of clusters, the samples drawn should be from clusters that the hierarchical sampling procedure believes to be of less purity or has less information about the best label for the cluster.

Outline of Hierarchical Sampling Algorithm

The general idea is to select the sample best determined by the pruning: a collection of nodes maintained by the algorithm throughout the lifetime of all samples being selected. The idea of pruning will be explained in the next section. After the sample is selected, the pruning and other internal state of the algorithm is updated. the updating of all internal state of the algorithm relies on metrics described in the "Update Method: Pruning Adjustment" heading in this section. The process of sampling and updating internal state is repeated for as long as the user desires or when all the unlabeled points have been revealed (similar to the other two sampling algorithms). In our implementation, however, we run a "tuning phase" before any sampling is done. This is different from the algorithm proposed by the paper, which will be discussed later, and allows the algorithm to select better unlabeled samples based on the existing hierarchical structure.

Hierarchical Clustering: Tree Construction and Usage of Tree

The hierarchical clustering procedure is conducted via an agglomerate clustering algorithm using scikit-learn. The weight of each node v is defined as $w_v = \frac{L_v}{L_{root}}$ where L_n is the number of leaves in the subtree of some node n. The weights are computed via a reverse topological ordering of the tree, where the children are processed before the parent nodes.

We define a pruning to be a set of nodes in the tree such that their subtrees are disjoint and the union of their subtrees covers all leaf nodes. The hierarchical sampling process selects which node in the pruning to draw a leaf node (sample) from based on the weight of the node in the pruning. The very initial pruning is a set only consisting of the root node, and the largest sized pruning is a set consisting of all the leaf nodes. The pruning is adjusted as more and more unlabeled data get sampled by the hierarchical sampling procedure. Intuitively, the greater the size the pruning, the deeper the tree the pruning lies in, which means the sampling procedure will have more control over which cluster the sample will be drawn from. However, a large pruning may not be required: some clusters may already be very pure, and the chances of selecting that cluster to draw samples from would not improve the resulting model that will be constructed. The algorithm carefully constructs the size of the pruning and the nodes in the pruning such that a balance is achieved.

Sampling

The hierarchical sampling procedure is now able to sample points. Three different methods of selecting the nodes to draw unlabeled samples from were discussed in the paper. We implemented the method where a node was selected in a weighted random fashion based on the node weights. A random unlabeled sample that is within the subtree is selected. However, it may be possible that all of the leaves of the subtree have been sampled already: in this case, a different node must be selected at random again, and a leaf must be randomly chosen again. We rely on the size of the dataset to avoid this edge case from occurring often.

After the leaf is selected based on the node in the pruning, the label of the leaf (sample) is revealed. All nodes in the path from the selected node in the pruning to the leaf have now observed that revealed label. This information needs to be propagated to all nodes along that path, and is achieved through the mapping from label to number of those labels seen for each node along the path.

Update Method: Pruning Adjustment

Before the sample and label get returned, the pruning must be updated. A few terms need to be defined in order to compute the scores which will be used to update the pruning.

Let $p_{v,l}$ be the fraction of label l in node v, which can be computed easily via the mappings from label to number of those labels seen (described in the previous section). However, this is only an estimate of the true fraction. A confidence interval for the true fraction can be computed via a lower bound $p_{v,l}^{LB}$ and an upper bound $p_{v,l}^{UB}$ as $p_{v,l}^{LB} = max(p_{v,l} - \Delta_{v,l}, 0)$ and $p_{v,l}^{UB} = min(p_{v,l} - \Delta_{v,l}, 1)$ where $\Delta_{v,l} = \frac{1}{L_v} + \sqrt{\frac{p_{v,l}(1-p_{v,l})}{L_v}}$.

An admissible set is maintained throughout the entire sampling procedure and continuously grows as more samples are drawn. A node and a label (v, l) is defined to be admissible if we are confident that assigning the label to all leaves in the subtree of v incurs little error. This is formally defined in the paper as

$$A_{v,l} = true \iff (1 - p_{v,l}^{LB}) < \beta \cdot \min_{l' \neq l}(1 - p_{v,l}^{UB}) \tag{4}$$

We define an error of assigning l to all samples in the subtree of v as $\tilde{\epsilon}_{v,l}$:

$$\tilde{\epsilon}_{v,l} = \begin{cases} 1 - p_{v,l} & (v, l) \; in \; admissible \; set \\ 1 & o.w. \end{cases} \tag{5}$$

The scores of the nodes in the pruning need to be computed, since they are used to adjust the pruning itself. The score of a node $s(v)$ is formally defined in the paper, but is much clearer when defined recursively as the minimum of

- $\tilde{\epsilon}_{v,l} \; \forall l$
- $\frac{w_a}{w_v}s(a) + \frac{w_b}{w_v}s(b)$ when (v, l) in admissible set for some l and a and b are children of v

Where w_n is the weight of node n.

To compute the scores, first compute all values for $p_{v,l}$, $p_{v,l}^{LB}$, and $p_{v,l}^{UB}$. Then, compute the admissibility of all (v, l) pairs and add to the admissible set. Then, in a single bottom-up pass of the tree, compute all $\tilde{\epsilon}_{v,l}$ and $s(v)$ values. This can be done via a reverse topological sort again. However, in the process of computing these values, the best pruning associated with each node's subtree P' and the labelings associated with the nodes in those prunings L' should be stored as well.

After the scores, P', and L' have all been computed, the pruning can now be adjusted. The pruning is replaced with the union of P' of each node in the current pruning. The labels associated with each node in each P' are updated as well. This concludes the update step.

Deviation from Original Algorithm: Tuning the Sampling Procedure

Based on the algorithm, after the tree has been constructed, the hierarchical sampling procedure is ready to select unlabeled samples to reveal labels for. However, the sampling procedure also has access to the current samples that already have their labels revealed: the training data. The algorithm in the paper ignores the existing training data and reveals samples from unlabeled data directly without considering the potential structural information provided by the training data.

We modify the algorithm for better predictions as follows: before the hierarchical sampling procedure samples any points from the unlabeled samples, we tune the procedure by "sampling" all training data. This is described as the "tuning phase". The process of sampling, as described above, modifies the weights of the nodes and the nodes in the pruning. The "tuning phase" allows the very first sample queried by the user to be drawn from an already-tuned pruning instead of the initial pruning that only consists of the root node.

A more detailed bird's-eye view of the algorithm is outlined below:

Algorithm 1 Augmented cluster-adaptive active learning

```
$Pruning \leftarrow$ {root: -1} (Initialize pruning with root and majority labeling of root as something arbitrary)
$A \leftarrow \{\}$ (admissible set initially empty)
procedure INITIALIZE ALGORITHM
    for each node, maintain $LabelCounts$ which maps $label \rightarrow count$
    construct tree with AgglomerateClustering
    compute weights for each node in reverse topological order
    run TUNING PHASE
procedure TUNING PHASE
    for sample t in training set do
        run SAMPLING but do not return t
        run UPDATE
    end for
procedure SAMPLING
    $v \leftarrow$ random weighted select node in $Pruning$
    $z \leftarrow$ randomly select unsampled leaf in subtree of $v$
    $l \leftarrow$ revealed label of $z$
    for node $u$ in upward path from $z$ to $v$ do
        $u.LabelCounts[l] += 1$
    end for
    run UPDATE
    return $z$
procedure UPDATE
    compute $p_{v,l}$ for every node using $LabelCounts$
    compute $p_{v,l}^{LB}$ and $p_{v,l}^{UB}$ using $p_{v,l}$ for every node
    for every $(v, l)$ pair do
        add $(v, l)$ to $A$ if $(v, l)$ is admissible
    end for
    for every node $v$ in reverse topological order do
        for every label $l$ do
            compute $\tilde{\epsilon}_{v,l}$
        compute $s(v)$
        save $P'$ and $L'$ associated with $s(v)$
        end for
    end for
    for every node $v$ in $Pruning$ do
        retrieve $P'$ and $L'$ associated with $s(v)$
        replace entry $v$ in $Pruning$ with every entry in mapping of $P' \rightarrow L'$
    end for
```

4 Evaluation

4.1 Data

Sports Article Objectivity Data

We first considered binary classification of the Sports Article Objectivity Data [6]. We used 800 training articles and 200 test articles. The purpose of this data set is to predict the objectivity of sports articles. The data set consists of raw text of 1000 sports articles. The raw text was already extracted into 59 features. The articles were given labels of ’subjective’ or ’objective’ using Amazon Mechanical Turk. Amazon Mechanical Turk is a service where you can pay to crowd-source simple tasks such as humans manually reading articles and labeling them ’subjective’ or ’objective’. We relabeled the data as binary 1 or 0 respectively. We found the data set through the UCI Machine Learning Repository.

Phishing Websites Data

We next considered binary classification of the Phishing Websites data [7]. We used 800 training websites and 1,656 test websites. The purpose of this data set is to predict whether or not a given website is a phishing website. The data set consists of 30 features from 2,456 websites. The websites were classified as either phishing or non-phishing. We relabeled the data as binary 1 or 0 respectively. The dataset was collected mainly from: PhishTank archive, MillerSmiles archive, and Google’s searching operators. We found the data set through the UCI Machine Learning Repository.

4.2 Quantitative Result

Accuracy Performance

Performance was defined in terms of classification accuracy. The goal being to minimize the classification error with as few labeled data points as possible. It is obvious to expect the classification error to decrease as the number of labeled training data points increases, but the performance of a sampling method is determined by how much the error decreases incrementally corresponding to the its chosen sampled data points.

For both data sets, logistic regression models were training on subsets of the data for each method, and the classification error of the test data was recorded. The subsets of data that were used to train the model are different quantities of training data points that have been labeled.

Figure 1 below shows the performance of each sampling method on the Sports Article Objectivity data set for each quantity of labeled training data points.

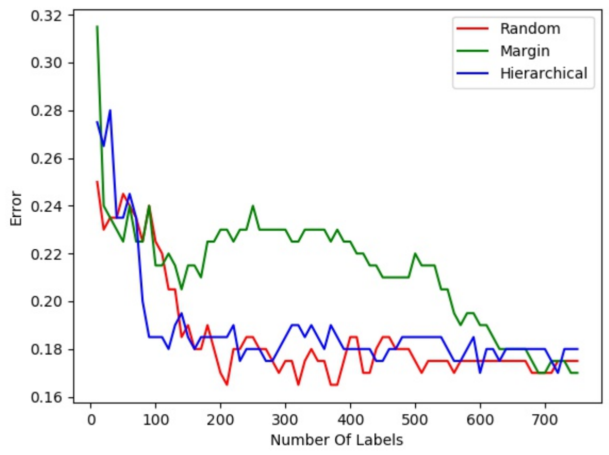

Figure 1: Results on Sports Article Objectivity

Figure 2 below shows the performance of each sampling method on the Phishing Websites data set for each quantity of labeled training data points.

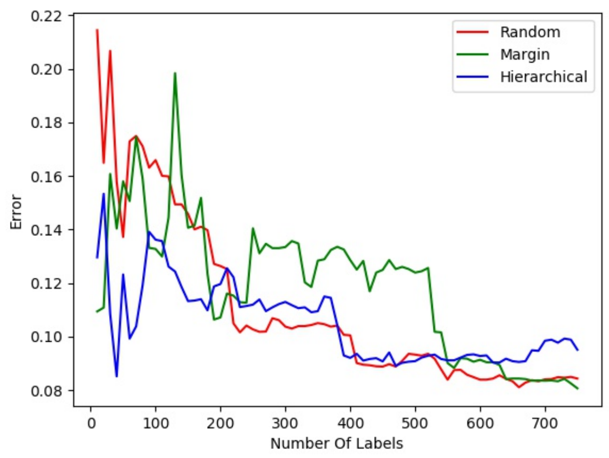

Figure 2: Results on Phishing Websites

Model Selection

To train the classification logistic regression models, a regularization parameter was used. Model selection was performed by testing multiple regularization parameter values. The performance of all sampling methods was evaluated with each regularization parameter value, and the selected regularization parameter was the one that performed roughly the best for all three sampling methods.

4.3 Qualitative Result

In both data set experiments, hierarchical sampling performed best up to around 150 data points, compared to the baseline alternatives of margin sampling and random sampling. Figure 1 and Figure 2 show that classification error is lower for hierarchical sampling compared to the other sampling methods for smaller number of labels.

The results also show the prevalence of sampling bias in margin sampling. In Figure 1 and Figure 2 it is shown that the margin sampling method has poor performance for an intermediate number of labels.

For larger number of labels, the methods converge together in terms of performance. Figure 1 and Figure 2 show that once enough data is gathered, irregardless of sampling method, then the performance can no longer improve.

These results show that when only a few labels can be acquired, hierarchical sampling can be superior to other sampling methods. Tying back to the motivation of active learning where only a few unlabeled samples can be revealed due to an existing cost or budget, hierarchical sampling helps us select the most informative samples to reveal the labels for that will best augment the existing model (which is a logistic regression model in this case).

5 Conclusion

Main Findings

We implemented the hierarchical sampling method which is effective active learning framework for selecting informative data to label. To overcome the difficulty of sampling bias, the hierarchical sampling method utilizes clustering to exploit the inherent structure of the data. The empirical results showed that for fewer number of labeled data points, the hierarchical sampling method performed better than margin and random sampling for two classification data sets.

Successes

For two classification data sets we were able to successfully implement all three sampling methods: Random Sampling, Margin Sampling, and Hierarchical Sampling. Hierarchical sampling is an effective method to overcome sampling bias. This method can be used to greatly reduce the cost of acquiring informative labels.

Weaknesses

We could not test on the large and complex data sets with many features, due to the hierarchical sampling algorithm taking too long in the update procedure. Even with a parallel algorithm, the computation for the results on a larger dataset got timed out on CAEN due to taking over 6 hours.

Possible Implementation Extension

A possible way to achieve even better accuracy using hierarchical sampling would be to augment the existing implementation with marginal sampling when selecting the leaf node associated with a node in the pruning. The current implementation is a purely random selection, but a selection based on estimated posterior probabilities may possibly provide a better selected unlabeled sample. Hierarchical sampling may achieve better accuracy when working with many features or when selecting a large portion of unlabeled data to reveal labels for.

6 Description of Individual Effort

Daniel implemented the random, margin, and hierarchical sampling code, constructed a parallel program to fit and predict the data sets, and wrote the hierarchical sampling section of the report. Chung Hoon pre-processed the data, prepared all the submission on flux for computing, and worked on the paper. Louis implemented the plotting code for the results of each sampling method. He wrote the evaluation section of the report, as well as performed final edits on the report. Juyoung pre-processed the data sets, debugged algorithms and codes worked in Python. He also worked on the writing up of the report and arranged the meeting.

References

[1] Yoav Freund & H. Sebastian Seung & Eli Shamir & Naftali Tishby(1997) Selective Sampling Using the Query by Committee, David Haussler (eds.), Kluwer Academic Publishers

[2] Balcan, M.-F., Beygelzimer, A., & Langford, J. (2009) Agnostic Active Learning. *ICML*

[3] Balcan Maria-Florina &, Broder Andrei. & and Tong Zhang. (2007) Margin Based Active Learning. *COLT*

[4] Castro, R., & Nowak, R. (2007) Minimax bounds for active learning. *COLT*

[5] Dasgupta & Sanjoy & and Daniel Hsu. (2008) Hierarchical Sampling for Active Learning *Proceedings of the 25th International Conference on Machine Learning - ICML* doi:10.1145/1390156.1390183.

[6] Dua, D. and Karra Taniskidou, E. (2017). Sports articles for objectivity analysis Data Set. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science)

[7] Dua, D. and Karra Taniskidou, E. (2017). Phishing Websites Data Set. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science)