Practical methods of Active Learning

Sunita Sarawagi
sunita@iitb.ac.in
April 15, 1993

Abstract

In many machine learning applications, labeled data is scarce but unlabeled data plentiful. However, labeling them requires tedious human supervision. The goal of active learning \cite{2, 6, 12, 3, 1, 5} is to seek out from the unlabeled pool those instances which when labeled will help strengthen the classifier at the fastest possible rate. An active learner starts with a limited labeled and a large unlabeled pool of instances. The labeled set forms the training data for an initial preliminary classifier. The active learner then carefully selects a few examples from the unlabeled set and seeks its labels from a user. The newly labeled set is added to the training set and the classifier retrained. The new knowledge is used to select another set of unlabeled instance and this continues in a loop until the user is happy with the learnt classifier.

The number of examples required to be manually labeled can reduced significantly using active learning. This has been both observed in several empirical studies and proved theoretically. \cite{7, 6}. Hence, there is increasing interest in applying this method of labeling data in several practical applications, including, text classification \cite{6, 12, 5}, information extraction \cite{11}, duplicate detection \cite{8, 7}, image labeling and querying \cite{10} and speech recognition \cite{12}.

Starting with some of the early work in \cite{7, 6, 10}, a number of methods have been proposed for active learning. Some of these have been theoretically analyzed, others are specific to particular kinds of classifiers like Support Vector Machines, and developed and tested specifically in particular application settings. We present a taxonomy of the various methods of active learning in Section ??.

In this paper our goal is to consolidate all these myriad approaches, present a detailed empirical evaluation of the various methods of doing active learning, on different classification methods and against multiple different datasets. We hope that this study will provide a practical perspective on the alternatives for active learning. We will therefore only consider those methods that do not require excessive amount of computation in coming up with the instance to be labeled next.

1 Conceptual overview of active learning

We will first give a conceptual overview of the criteria used for active learning without bothering about the efficiency of evaluation. Later, in Section ?? we will present practical alternatives.

The main challenge in active learning is estimating an instance’s impact in improving a classifier without knowing its label. We need to explicitly model the informativeness of an instance by measuring how much including that instance in the training set results in the improvement of the classifier. We will need to address two issues:

- How do we quantify the quality $C$ of a classifier for a given training set $T$?
- Without knowing the label of an instance $x$ how do we estimate the reduction in $C$ possible by augmenting the training set $T$ with $x$ correctly labeled? The best we can do is to measure expected impact as follows. We can calculate expected value of $E_p(Q(L^- < x, y >) + C(T'))$ with the addition of an unlabeled instance $x \in U$. We use the current classifier $C^t$ to find the probability $Pr(y|x)$ that an instance $x$ will get assigned class $y$. We then estimate expected impact $U(x)$ of an instance $x$ as

$$U(x) = |C| - \sum_y Pr(y|x)|C(T \cup (x, y))|$$

(1)

where $|C|$ is the size of the version space before adding the instance and $|C(T \cup (x, y))|$ is the version space size after adding $x$ with label $y$ to the training set $T$. 

1
Existing variants of quantifying $Q$ for the purposes of active learning can be grouped into two types based on whether the unlabeled data $U$ is used or not. The first type of method relies on some property of the structure of the trained classifier, notably the version space size. However, such properties are classifier specific and not easy to find always. The second type of methods rely on some measure of a loss function or confusion on the unlabeled data $U$. A third alternative could be to use a setaside labeled data to evaluate the goodness of the current classifier. However, in practice we may not have such data easily available since active learning is primarily used in situations where labeled data is sparse.

1.1 Data-specific method

A direct method to quantify the quality of a classifier is the expected future error on the available unlabeled instances [7]. There are a number of ways of doing this. If the underlying classifier can output the posterior probability of the class label for an instance $x$, then the error can be approximated to be the log-loss on the posterior distribution ($\sum_x \sum_y \log \Pr(y|x)$) or the 0-1 loss assuming the $y$ is true.

Another way of approximating $C$ is through predictions on the data instances from a committee of classifiers. Instead of training just a single classifier $C$ we sample a committee of $N$ classifiers, $C_1, C_2, \ldots, C_N$ from the set $C$. The different ways in which this sampling can be done is discussed later in Section ???. If $C$ were close to $C^{opt}$ the predictions of all classifiers in the committee would be the same. In contrast if the size of $C$ is large, the $N$ classifiers in the committee are likely to be very different from each other and this will be reflected in the disagreement in the predictions on an unseen instance. We claim that the size of $C$ is proportional to the disagreement in the predictions of the committee member over all unlabeled instances.

1.2 Version-space method

We will present the main intuition behind such an estimation by considering first the case where the two classes are perfectly separable by a chosen method of classification. A version space [7] of a classifier with respect to a training set $T$ is the set $C$ of classifiers all of which are capable of perfectly separating the points in $T$. A training algorithm materializes one member $C^t$ of $C$. In the limit when the training set size approaches infinity, $C$ approaches $C^{opt}$, the optimal set of classifiers. Clearly, $C^{opt}$ is a subset of $C$. When the size of $C$ is large compared to the size of $C^{opt}$, the probability of $C^t$ (a member of $C$) belonging to $C^{opt}$ is small. Thus, larger the size of the version space, greater the error or variance of the classifier. The process of training by adding more and more training instances is to reduce the size of $C$ until it approaches $C^{opt}$.

We can quantify the worth of adding a new instance $x$ to a training set $T$ as how much knowing it will reduce the size of the version space of the classifier. This immediately raises two issues:

- How do we estimate the size of $C$ for a given training set $T$?
- Without knowing the label of $x$ how do we estimate the reduction in $C$ possible by augmenting the training set $T$ with $x$ correctly labeled?

**Estimating the size of $C$** Except for the simplest of classification methods, the size of $C$ cannot be measured exactly. However, several measures have been proposed to approximate this quantity in various ways. **Margin method:** Approximate the size of the version space by the volume of the margin between the given positive and negative instances. Such an estimation is directly possible only in certain discriminating classifiers like Support Vector Machines where the size of the margin is explicitly modeled during the training process. [12] [9] presents a method of active learning that exploits this intuition. Unfortunately, such an approach is not applicable to other classification methods, particularly the ones based on generative models like naive Bayes.

The final algorithm for active learning then is:

**Example:** Consider a very simple learner for separating points from two different classes: positive ($P$) and negative ($N$) on a straight line as shown in Figure ???. Assume that the set of points are separable by a single point on the line. The initial training set consists of one positive point $b$ (star) and one negative point $r$ (circle) picked randomly from the line. The rest of the points(squares) are unlabeled. The size of the version space $C$ is then the distance between $r$ and $b$. Any unlabeled point $x$ to the left of $r$ and the right $b$ will have no effect on the version space. Hence they will not be selected for active learning. Consider points in between $r$ and $b$. For any point $x$ in this region, the probability that it is negative is proportional to its distance from $r$. For simplicity, assume $r$ has a coordinate of 0 and $b$ has a coordinate of 1. Thus, if $x$ has a coordinate of $d$, the probability that its class is negative ($N$) is $Pr(N|x) = d$ and $Pr(P|x) = (1 - d)$. If $x$ were negative, the size of the version space would reduce
by $d$, if it were positive the size would decrease by $(1 - d)$. Hence, the expected reduction in version space size on adding $x$ to the training set is $\Pr(N|x)d + \Pr(P|x)(1 - d) = dd + (1 - d)(1 - d)$. This achieves the maximum value when $d = 0.5$, that is, we choose a point like $m$ exactly in between $r$ and $b$. By including $m$ in the training set, the size of the uncertain region will reduce by half no matter what its label. Any other point say $s$ that is close to the negative boundary but far from the positive boundary could reduce the version space more if its true label is found to be positive but the probability of that is small given the current training data. Thus, the expected reduction in the size of the version space is smaller for this point than for $m$.

2 Practical alternatives of active learning

The above approach trains a classifier for each unlabeled instance and class pair, before it can find the instance most effective for inclusion in the training set. Thus, if we have 3 possible class labels and 10,000 unlabeled examples, we would need to train 30,000 classifier before we can pick one for labeling. This is clearly impractical. A number of methods can be used to make this efficient by employing incremental methods of updating a classifier on the addition of a single example and by subsetting the set of unlabeled examples using sampling. Even so, the computational cost could be prohibitive. In this section we present practical approximations to these methods that have been proposed in the literature. The techniques discussed are a distillation of the various methods of active learning proposed recently.

The main intuition in almost all such methods is to select instances about which the classifier(s) built on the current training set is most uncertain. The unsure instances are those that fall in the classifier’s confusion region. This confusion region is large when the training data is small. The classifier can perhaps reduce its confusion by seeking predictions on these uncertain instances. Instances whose prediction the learner can already make with strong confidence will likely not have much effect on the learner. Theoretical justification for approximating expected reduction in confusion (formally, version space) with prediction uncertainty appear in.

Real-life data is noisy and when picking instances based on uncertainty we need to make sure that we are not picking erroneous or outlying instances. We are more likely to gain from instances that are representative of a large number of unlabeled instances, than an extreme outlying instance. To ensure this, a second criteria that becomes important is the representativeness of an instance.

In the rest of the section we discuss how to quantify the notions of uncertainty (Section 2.1) and representativeness of an instance (Section 2.2).

2.1 Uncertainty score of an instance

There are two major ways of evaluating the uncertainty of the prediction on a instance.

2.1.1 Classifier-specific confidence measure

An intuitive method for measuring uncertainty for separator based classifiers like SVMs and regression is to make it inversely proportional to the distance of the instance from the separator. Similarly for bayesian classifiers, the posterior probabilities of classes can be used as an estimate of certainty. For decision trees, typically uncertainty is derived from the error of the leaf into which the instance falls. For nearest neighbor classifiers, a natural choice is to make the uncertainty proportional to the average distance to the test instance.

The above intuitive notions of certainty do not always work well in practice. For instance, the naive Bayes posterior probabilities has been known to grossly over-estimate the probabilities of the winning class. Similarly, for decision tree classifiers, most leaves have so few instances that the errors at the leaf level are not very meaningful as observed.

The decision tree classifier calculates the uncertainty of each instance as the weighted sum of the following three values

- Error on the training data at the leaf which classifies the unlabeled instance.
- Depth of the leaf which classifies the unlabeled instance.
- Sum of absolute distances from each test condition upto the leaf which classifies the unlabeled instance.
2.1.2 Committee-based approach

A classifier independent way of deriving the uncertainty of an instance is by measuring the disagreement amongst the predictions it gets from a committee of N classifiers. The committee is built so that the N member classifiers are slightly different from each other, yet they all have similar accuracy on the training data.

Theoretically, the members of the committee should all be members of the version space of the classifier with the current training data. The different members thus provide redundant ways of classification. A sure instance would get the same prediction from all members. The uncertain ones will get different labels and by adding them in the training set the disagreement amongst the members will be lowered in the next iteration. Uncertainty of predictions of a committee can be quantified in various ways.

We quantify the disagreement amongst the predictions of the different members of a committee on a given instance $x$ using the entropy function. Let $p_i$ denote the fraction of members that predict the $i$th label for $x$. The uncertainty $U(x)$ associated with $x$ is calculated as $U(x) = -\sum_{i=0}^{k} p_i \log(p_i)$. In our case, the set of class labels can take just two value 0 (for non-duplicate) and 1 (for duplicate). Hence, the uncertainty of an instance is:

$$U(x) = -p_1 \log(p_1) - (1 - p_1) \log(1 - p_1)$$  \hspace{1cm} (2)

where $p_1$ is the fraction of committee members that declare the instance as duplicate. This function will assume the highest value when exactly half the members vote the instance as duplicate.

2.1.3 Methods of creating committees

We next present three different ways of creating committees.

Randomizing model parameters  A common method of creating committees is by making small perturbations on the parameters of the model trained through the given training data [10]. The perturbations are made in the following way.

The first is the version space approach where the members are built to be all within the version space of the current training data yet as far from each other as possible. This is achieved heuristically by randomizing the various decisions that are sometimes taken arbitrarily during the construction of the classifier. An example of this is the committee of Winnow classifiers [5]. We propose a mechanism for randomizing decision tree classifiers. During tree construction, when selecting an attribute for splitting on next, instead of deterministically choosing the attribute with the highest information gain, we randomly pick one with information gain within close range of the best. Secondly, when picking the threshold on which to split a continuous attribute, instead of picking the midpoint of the range within which the information gain remains unchanged, we pick a point uniformly randomly from anywhere in the range. Similar simple ideas have been proposed in [9]. This approach is not applicable to all probabilistic classifiers where the concept of a version space is not well-defined.

The second approach is the random sampling approach where the models are sampled based on the probability of the model given the training data. We explicitly model the distribution of the parameters given the training data. Then sample from these parameters. The form of the distribution varies based on the model and parameter type. Some of the previous approaches show how to perturb the parameters of a Naive Bayes classifier [6] and Hidden Markov Model [1].

The emission probabilities for $s$ follow another multinomial distribution with as many parameters as the size of the vocabulary. Normally, when presented with the training dataset the values of these parameters is estimated using the maximum likelihood as $\frac{n_i}{n}$ where $n$ is the number of training instances and $n_i$ is the number that supports the $i$th outcome of the multinomial. Now, instead of selecting a single parameter value we need to model a distribution around the value so that we can sample from it. For the binomial distribution (special case of the multinomial distribution with just two possibilities) the parameters follow a beta distribution ( with a mean value of $\frac{n_i}{n}$). The variance of the distribution (Beta($\alpha$, $\beta$) $= \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx$, in our case $\alpha = n_i$, $\beta = n - n_i$) is large when the number of training instances is small.

For the multinomial case, the posterior distribution is a Dirichlet distribution. For ease of computation, the Dirichlet distribution can be approximated by a set of normal distributions one for each parameter and the parameters can be later renormalized to sum up to one.

For example, consider the transition probability parameters $a_{ji}$ out of a state $j$. The value of $a_{ji}$ is estimated as $n_{ji}/n$ where $n$ is the total number of transitions out of state $j$ and $n_i$ is the number out of those that go to state $i$. The $a_{ji}$ sum up to 1 over all $i$ and form a multinomial distribution. To sample, from this distribution we first approximate a normal distribution around each parameter $a_{ji}$ with mean $n_i/n$ and variance $n_i(n - n_i)/n^3$. The different sampled $a_{ji}$ are then normalized to sum to 1.
Thus, the procedure for generating the members of the committee is to first train the parameters of the first HMM using maximum likelihood on the training instances. For each subsequent member, at each state’s transitions and emission, model a Dirichlet distribution with counts estimated from the training dataset. Sample from this distribution to determine the parameters of the other states.

**Partitioning training data** A second model-independent method of creating committees is by partitioning the training dataset in various ways, including disjoint partitioning, resampling as in bagging, and N-fold overlapping partitioning. Disjoint partitioning did not work well in our experiments due to the limitation of training data in the early phase of active learning. In the N-fold overlapping partitioning where (for a committee of size N) we first partition a training dataset $D$ into N disjoint sets $D_1, D_2, \ldots, D_N$. Then, train the $i$-th committee with the dataset $D - D_i$. This way each member gets trained on $(1 - \frac{1}{N})$th fraction of the data.

### 2.2 The representativeness of an instance

Real-life data is often noisy. The *most* uncertain instance could often be an outlier. An uncertain instance that is representative of a larger number of unlabeled instances is likely to have a greater impact on the classifier, than an outlying instance. Hence, another important factor is how representative an instance is of the underlying data distribution.

The main challenge in incorporating the representativeness factor is figuring out how to combine it with the uncertainty factor. Two different methods have been proposed for this.

The first approach explicitly measures the representativeness of an instance by estimating the density of points around it after clustering the unlabeled instances [6]. The instances are then scored using a weighted sum of its density and uncertainty value and the $n$ highest scoring instances selected. This method requires us to tune several parameters: distance function for clustering, number of clusters and the weights to tradeoff uncertainty with representativeness.

A second more common approach relies on *sampling* to preserve the underlying data distribution [3-5]. First, each candidate unlabeled instance is weighted by its uncertainty value. Then the $n$ instances for active learning are selected from this using weighted sampling. We chose and experimented with different variations of this approach starting with no-sampling to full-sampling. In no-sampling we simply pick the $n$ highest uncertainty instances. In full-sampling we do weighted sampling on the entire unlabeled set. An intermediate approach is to first pick the top $kn$ ($k \geq 1$) instances based on uncertainty and then do weighted random sampling on them to select $n$ out of these $kn$ instances. $k = 1$ then corresponds to no sampling, $kn = $ total data size corresponds to the full sampling.

### References


