Learning Updatable Classifiers from Remote Data

Harris T. Lin
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
htlin@iastate.edu

Neeraj Koul
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
neeraj@iastate.edu

Vasant Honavar
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
honavar@iastate.edu

Abstract

The increasing availability of large data offers an exciting opportunity to use such data to build predictive models using machine learning algorithms. However, most approaches to learning assume direct access to data, and can not efficiently cope with frequent updates to the data. In this paper we show that learning using statistical queries provides a powerful paradigm to address these challenges. We summarize our work and present INDUS, an open source implementation of learning algorithms based on the proposed statistical query paradigm.

1 Introduction

Recent development of high throughput data acquisition technologies coupled with the rise of Semantic web and cloud computing have led to unprecedented opportunities in data-driven knowledge acquisition. Most approaches to learn from such massive data sources assume direct access to data. However, this assumption of direct access to data may not hold in practice due to a variety of reasons such as (i) the massive size of the data being made available and the corresponding memory and bandwidth constraints that prohibit the entire data being shipped to a centralized location and (ii) in certain domains, such as personalized medicine, privacy regulations may explicitly prohibit direct access to data. Further, even in cases where data can be shipped to a centralized location, the local copy of the dataset may quickly become out of date due to frequent updates to the data. Hence, there is a need for algorithms that can learn from massive datasets that exceed the size of the memory available to the learner, do not require access to the entire dataset and are able to cope with frequent updates to the data. We summarize our previous work [1, 3, 2, 6] and present INDUS [5], an open source suite of learning algorithms that address these scenarios.

2 A General Framework for Learning from Remote Data

We consider the following learning problem (Fig. 1).

Problem 1. Let $D$ be a dataset and $L$ be a learner that does not have direct access to $D$. Given a descriptor (or schema) $S$ of $D$, a query interface $Q$ of $D$, a hypothesis class $H$ of $S$, and a performance criterion $P$, the learning algorithm $L$ outputs a classifier $h \in H$ that optimizes $P$, using the set of queries supported by $Q$. The query interface $Q$ is expected to support queries expressed in terms of $S$ that can be used to compute sufficient statistics of $D$ as required by $L$ to output $h$. 
Figure 1: Learning a hypothesis from remote data, where we assume the learner \( L \) does not have direct access to \( D \) (in other words \( D \) is remote from \( L \)).

Figure 2: Updating a previously learned hypothesis, when new data \( D' \) has been added.

Our approach to learning classifiers from remote data adopts the general framework introduced by Caragea et al. [1] for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves decomposing the learning algorithm into two parts: (i) a component that poses the relevant statistical queries\(^1\) to a data source to acquire the information needed by the learner; and (ii) a component that uses the resulting statistics to update or refine a partial model (and if necessary, further invoke the statistical query component).

In a more general setting, \( h \) is constructed by \( L \) by interleaving query and hypothesis generation operations. For example, a decision tree learning algorithm would start with an empty initial hypothesis, obtain the sufficient statistics for the root of the decision tree (a partial hypothesis), and recursively query additional statistics needed to refine the partial hypothesis. A formal treatment of this generalization of hypothesis refinement can be found in [1].

This approach of learning from datasets using only sufficient statistics allows us to cope with (i) massive data size (since in general the statistics are much smaller than the dataset), (ii) no access to the underlying dataset (as long as the data source \( D \) provides a query interface \( Q \) to which queries can be submitted to obtain the sufficient statistics), and (iii) frequent updates (since there is no local copy that can become out of date and often the learning model can be updated by updating the collected statistics). In the next section, we outline a few representative algorithms that can be build from a dataset using only sufficient statistics.

### 3 Representative Algorithms for Learning Classifiers from Remote Data

#### 3.1 k-dimensional Data

The schema of a \( k \)-dimensional dataset is defined as \( S = (A, C, V) \) where \( A = \{a_1, \ldots, a_k\} \) is the set of \( k \) attributes, \( C \in A \) corresponds to the class label, and \( V = \{V_1, \ldots, V_k\} \) is the set of domains for each attribute. The dataset is \( D \subseteq V_1 \times \ldots \times V_k \). We describe below the queries that are sufficient to build from each of the hypothesis class.

**Naive Bayes (NB)**

Given a dataset \( D \) and the corresponding schema \( S \), we need to estimate \( p(C = c) \) and \( p(a_i = v | C = c) \) from \( D \). We require \( Q \) to support the query \( S(D, \ldots) \), which returns the number of instances in \( D \) that satisfy the conditions in the rest of the argument. For example, if \( D \) is stored

---

\(^1\)A statistic is simply a function of a dataset. A statistical query returns a statistic (e.g., the number of instances in the dataset that have a specified value for a specified attribute.)
in a relational database that has an SQL query interface, then \( S(D, C = c, a_i = v) \) can be realized with the SQL query: 
\[
\text{SELECT COUNT(*)} \text{ FROM } D \text{ WHERE } C = c \text{ AND } a_i = v .
\]
Then, it is shown in [3] that the required probability can be estimated by 
\[
p(a_i = v | C = c) = \frac{S(D, C = c, a_i = v)}{S(D, C = c)}.
\]

### Decision Tree

The decision tree learner recursively chooses at each step an attribute that yields the most information gain against the class label. In [3] it is shown that the information gain, and hence the decision tree, can be built using queries of the form \( S(D, C = c, a_1 = v_1, \ldots, a_i = v_i) \). It can also be realized into similar SQL queries as in the case of Naive Bayes.

### Support Vector Machine (SVM)

SVM constructs a binary classifier that corresponds to a separating hyperplane that maximizes the margin of separation between instances belonging to two classes. Because the weight vector that defines the maximal margin hyperplane can be expressed as a weighted sum of a subset of \( D \) (called support vectors), the support vectors and associated weights constitute a sufficient query for \( D \) [1].

#### 3.2 Sequence Data

Rapid advances in bio-sequencing technologies have resulted in terabytes of DNA sequence, protein sequence, and gene expression data being gathered at steadily increasing rates. Here we focus our attention on building the class of models that observe the Markov property: Markov Model (MM), Interpolated Markov Model (IMM) [11], and Probabilistic Suffix Tree (PST) [10]. For a sequence dataset, \( S = (\Sigma, C) \) where \( \Sigma \) is the alphabet from which the sequences in \( D \) are constructed, and \( C \) is the classes to which the sequences in \( D \) are labeled. We have \( D \subseteq \Sigma^* \times C \).

In an Markov model of order \( k - 1 \), the estimate of the probability that a given sequence \( s = \sigma_1\sigma_2\ldots\sigma_n \) belongs to the class \( c \) is given by 
\[
p_{MM(k - 1)}(s, c) = \prod_{i=k}^n p(\sigma_i | \sigma_{i-1}\ldots\sigma_{i-k+1}, C = c).
\]

We assume \( Q \) answers queries of the following form [2]: (i) the query to compute the count of sequences in \( D \) that belong to the class \( c \) and have the subsequence \( s \), denoted by \( S(D, s, C = c) \), and (ii) the query to compute the count of sequences in \( D \) that belong to the class \( c \) and have subsequences of length \( |s| \), denoted by \( S(D, |s|, C = c) \). Then, as shown in [2], each of the above multiplication term can be estimated by 
\[
p(\sigma_i | \sigma_{i-1}\ldots\sigma_{i-k+1}, C = c) = \frac{S(D, \sigma_i | \sigma_{i-1}\ldots\sigma_{i-k+1}, C = c)}{S(D, |s|, C = c)}.
\]

It is shown in [2] that IMM and PST can also be built using similar queries as described above.

#### 3.3 Resource Description Framework (RDF) Data

The rise of Semantic web has resulted an increasing amount of structured and publicly linked data, in the data format known as Resource Description Framework (RDF) ([7] for a primer). RDF represents data in the form of subject-predicate-object triples, also called RDF triples, which describe a directed graph where the directed labeled edges encode binary relations between labeled nodes (also called resources). In [6] the proposed approach is demonstrated on learning from RDF data, using a Relational Bayesian Classifier (RBC) [8].

The dataset \( D \) is this case is simply a set of RDF triples. The schema is \( S = (T, A, C) \) where \( T \) is the target type (specifies a set of resources that are of interest), \( A \) is a tuple of attributes, and \( C \) is the class label. An attribute now specifies a subgraph that are related to a particular resource. In [6] \( D \) is first reduced into a multiset attributed dataset, and then RBC is built from the reduced dataset.

The parameters of RBC can be estimated using several methods such as aggregation, and it is also shown that the parameter estimation can be expressed in terms of SPARQL [9]: a query language designed for RDF.
4 Updatable Predictive Models

In many settings, the dataset $D$ undergoes frequent updates i.e., addition or deletion. In such settings, it is necessary to update the hypothesis to reflect the changes in the data used to build the hypothesis. While in principle, the learning algorithm can be re-executed each time there is an update, it is of interest to explore more efficient solutions for incrementally updating the hypothesis by updating only the relevant statistics (see Fig. 2).

Given a dataset $D$ and a learning algorithm $L$, let $L(D)$ be a predictive model built from the dataset $D$, and let $\theta_L(D)$ be the set of primitive queries required over the dataset $D$ to build $L(D)$.

**Definition 1 (Updatable Model [2]).** Given datasets $D_1$ and $D_2$ such that $D_1 \subseteq D_2$, we say that $L$ is updatable iff we can specify functions $f$ and $g$ such that:

1. $\theta_L(D_2) = f(\theta_L(D_2 - D_1), \theta_L(D_1))$
2. $\theta_L(D_1) = g(\theta_L(D_2), \theta_L(D_2 - D_1))$

Under this definition, NB, MM, and IMM are shown to be updatable [2]. A weaker form of updatable model is defined in [6], where RBC is shown to be not updatable in general, but is updatable under a restricted setting.

5 INDUS: INtelligent Data Understanding System

The presented approach has been implemented as part of INDUS [5], an open source system that learns predictive models from remote data source using sufficient statistics. A high level system architecture is shown in Fig. 3. The current version includes Naive Bayes, Decision Trees, and Relational Bayesian Classifiers. Datasets for Naive Bayes and Decision Trees are assumed to be in a data source that provides an SQL query interface. The Relational Bayesian Classifier is used to learn from an RDF dataset via SPARQL query interface. INDUS has also been extended to learn from multiple distributed data sources. This has been done by introducing an integration framework that presents multiple distributed and semantically disparate data sources as a single data source to the learner [4], which enables the learner to proceed as if it is learning from a single data source. Future work involves adapting other learning models into the statistical query paradigm and implementing them on the INDUS platform.
References


