# Bellwether Analysis: Predicting Global Aggregates from Local Regions

Bee-Chung Chen[1], Raghu Ramakrishnan[1,2], Jude W. Shavlik[1], Pradeep Tamma[1]
[1] University of Wisconsin – Madison, USA
[2] Yahoo! Research, Santa Clara, CA, USA
{beechung, shavlik, pradeep}@cs.wisc.edu ramakris@yahoo-inc.com

## ABSTRACT

Massive datasets are becoming commonplace in a wide range of domains, and mining them is recognized as a challenging problem with great potential value. Motivated by this challenge, much effort has been concentrated on developing scalable versions of machine learning algorithms. An often overlooked issue is that large datasets are rarely labeled with the outputs that we wish to learn to predict, due to the human labor required. We make the key observation that analysts can often use queries to define labels for cases, which leads to the problem of learning to predict such query-produced labels. Of course, if a dataset is available in its entirety, we can simply run the query again to compute labels. The interesting scenarios are those where, after the predictive model is trained, new data is gathered at significant incremental cost and, perhaps, over time. The challenge is to accurately predict the query-labels for the projected completion of new datasets, based only on certain *cost-effective* subsets, which we call *bellwethers*.

## 1. INTRODUCTION

Mining large datasets is recognized as a challenging problem with great potential value, and much effort has been concentrated on developing scalable versions of machine learning algorithms. However, large datasets are rarely labeled with the outputs that we wish to learn to predict, due to the human labor that is typically required. This severely limits our ability to apply supervised learning techniques.

We make the key observation that for a large class of practically motivated problems, conventional database queries can be used to "tag" cases with the attribute values that we wish to predict, thereby mitigating the labeling difficulty. The "cases" are themselves the result of aggregating many database records. Consider a company that wants to predict the 1st year worldwide profit of a new item. After selling this item worldwide for one year, the company will know the exact profit. However, if the company can accurately predict the annual worldwide profit using features (e.g., regional profit, etc.) collected in a much shorter time (e.g., 1st week sales) and a much smaller area (e.g., only sales in Wisconsin), it has gained valuable business insight.

In this example, each item for which we have historical data is a case, and the information relating to this item is dispersed across all sales records for the item. We can create additional per-item features, and compute the desired label (worldwide annual profit for the item), by using conventional OLAP-style queries. In fact we can create training datasets by summarizing historical per-item sales for each region of interest (e.g., by state and month, or by county and week). We can then use each per-region training dataset to train a predictive data model. When a new item is introduced, if we collect sales data for a given region and aggregate this as before to create a case for the new item, the predictive model for the region can be used to estimate the desired label, which, in our example, is worldwide annual profit.

The question, then, is what is the best region on which to base such a predictive model, and whether a good region exists at all. Intuitively, gathering sales data for a new item in the region must be within an acceptable cost; cost could reflect real-world marketing expenses, for example. Further, the predictive model for the region must have high accuracy and low variance. We call such regions *bellwethers*, and the problem considered in this paper is how to identify bellwether regions.

In this paper, we make the following contributions: (1) We introduce bellwether analysis, a novel framework that allows us to apply predictive models to massive datasets without human labor for labeling the training examples. (2) We formalize many challenges raised by this framework, showing the richness of the problem and many opportunities for future research. (3) We develop several efficient, scalable algorithms to find bellwether regions, and evaluate their performance. (4) Using real-life datasets, we demonstrate the value of bellwether analysis.

The rest of this paper is organized as follows. After reviewing predictive models in Section 2, we introduce bellwether analysis in Section 3. We define the basic bellwether analysis problem, and an important variation, finding item-centric bellwethers. Intuitively, the basic approach finds a single region to serve as bellwether for all items, while the latter recognizes that different regions may be appropriate for different items or types of items. We present a scalable algorithm for basic bellwether analysis in Section 4, and two algorithms for item-centric bellwethers in Sections 5 and 6. In Section 7, we present a detailed experimental evaluation using both real and synthetic datasets, measuring both the quality of the bellwethers found and the efficiency of our algorithms. We discuss related work and conclude in Section 8.

## 2. BACKGROUND

Before formally introducing bellwether analysis, we first review some basics of predictive models [13]. Let **D** be a data table with attributes $X_1, \ldots, X_p, Y$, where $X_1, \ldots, X_p$ are called *features*, $Y$ is called the *target*, and each row in the table is called an *example*. A predictive model learns the relationship between $X_1, \ldots, X_p$ and $Y$ from **D** and predicts the value of $Y$ given a new example based on its $X_1, \ldots, X_p$ values. **D** is called the training set. We use $h$ to denote a predictive model, and $h(\boldsymbol{x})$ returns the target value of example $\boldsymbol{x}$. If the target $Y$ is a numeric value, $h$ is called a


*VLDB '06*, September 12–15, 2006, Seoul, Korea.

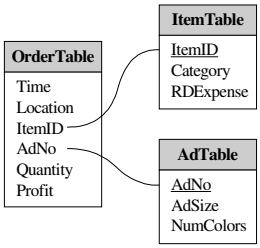


**Figure 1. Data schema of the motivating example**

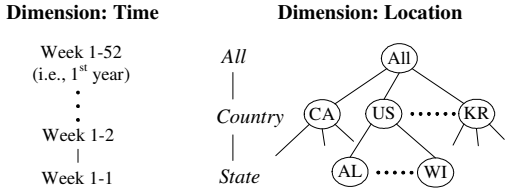


**Figure 2. Dimensions of the motivating example**

*regression model*. If $Y$ is a categorical value, $h$ is called a *classification model*. Decision trees, support vector machines, neural networks and linear regression models are examples of predictive models.

The quality of a predictive model is usually measured by the error (or equivalently, accuracy) of the model, which is the expected discrepancy between the true target value and the predicted value for a new example. For classification models, the misclassification rate (i.e., the expectation of making an incorrect prediction) is a commonly used error measure, while for regression models, the mean squared error (MSE) and root mean squared error (RMSE) are commonly used. MSE is the expected value of the squared difference between the true $Y$ value and the predicted one, and RMSE is the square root of MSE. However, in reality, the true distribution of $X_1$, …, $X_p$, $Y$ is generally unknown. Thus, the error of a model cannot be computed exactly, but needs to be estimated from the given data **D**. We consider two commonly used error estimates: cross-validation error and training-set error.

**Cross-validation error:** To compute cross-validation error, we first partition **D** into $n$ non-overlapping subsets of examples: $\mathbf{D}_1$, …, $\mathbf{D}_n$. For $i$ from 1 to $n$, we train a model on $\cup_{j \neq i} \mathbf{D}_j$ and test the model on $\mathbf{D}_i$ to obtain an error value. Then, the cross-validation error is the mean of the $n$ error values. Based on some distribution assumptions, the confidence interval of the cross-validation error can be obtained based on the variance of the $n$ error values. A commonly used $n$ is 10.

**Training-set error:** Another way to estimate the error of a model is to train the model on **D**, and then test it also on **D** to obtain the error value, which is called the training-set error. Usually training-set error is overly optimistic. However, for simple models, e.g., linear regression models, training-set error can approximate the true error. Note that the overhead of computing cross-validation error is approximately $n$ times that of computing training-set error.

# 3. PROBLEM DEFINITION

We first introduce a motivating example, and then formally define the problem of bellwether analysis. Intuitively, we want to use historical data to find a *region* (e.g., [1st week, Wisconsin]) with a small *cost* such that we can accurately predict the *target value* (e.g., the 1st year worldwide sales) of an *item* (e.g., a product) based on the *features* (e.g., the 1st week sales in Wisconsin) of that item collected from that region. As will be seen, this problem is significantly different from ordinary machine-learning problems in that both features and target values are generated by queries over the historical database.

## 3.1 Motivating Example

Consider a company that wants to predict the 1st year worldwide profit of a new item. After selling this item worldwide for one year, the company will know the exact profit. However, if the company can accurately predict this target value using features (e.g., regional profit, etc.) collected in a much shorter time (e.g., 1st week) and a much smaller area (e.g., only focus on Wisconsin), then it can quickly adapt its business strategy to minimize the loss or even maximize the profit. [1st week, Wisconsin] is an example of such a bellwether region. Our goal is to find such regions. Note that, in this example, we denote a region by a pair of time interval and location values.

To find such a bellwether region, the company can exploit its historical sales database, which contains three tables as shown in Figure 1. Each record in OrderTable represents a transaction of an item (identified by ItemID) at a specific time and location, which includes the quantity and the profit earned from that transaction. Item information is stored in ItemTable, which records the category and R&D expense for each item. Advertisement information is stored in AdTable, which contains the size and number of colors for each advertisement (identified by AdNo). Using the foreign keys, we can obtain the item and advertisement information for each transaction.

Let us first consider a straightforward data-mining approach. We can aggregate OrderTable to obtain the target value (i.e., the 1st year worldwide profit) for each historical item. Thus, a training set can be created by associating the features (Category and RDexpense) of each item, called the **item-table features**, with its target value. Then, we can train a predictive model (e.g., a linear regression model) on the training set, and use the model to predict the target value of a new item based on its features. If this model is very accurate, then no bellwether analysis is needed. However, since the item-table features are usually not sufficiently predictive, the accuracy of the model is usually not acceptable.

To improve the accuracy of the predictive model, adding more informative features is necessary. Note that we have not yet used the information provided by OrderTable and AdTable as features to help predict the target value. However, collecting such features for a new item incurs a cost. At one extreme, if we sell the new item worldwide for a year, we know the worldwide profit exactly. There is no need for prediction, but this incurs a very high cost. At another extreme, if we are not willing to pay anything, then we only have the item table information and no other features can be used. *The goal of bellwether analysis is to find a cost-effective* "region," *such that using new features collected from that region can best improve the accuracy of the model*.

In this example, a time interval and a location together define a region for data acquisition. Figure 2 shows the dimension structures. Any combination of an interval in the time dimension and a place in the location dimension is a candidate region. E.g., [1-1, WI], [1-2, US], and [1-52, All] are regions at different levels. Based on the company's experience, the cost of collecting data for each region can be defined.

For a given region [1-$t$, $loc$], new features of item $i$ can be generated by queries over the database, such as: