Learning from Aggregate Views

Bee-Chung Chen[*], Lei Chen[*], Raghu Ramakrishnan[*], David R. Musicant[+]
[*] *University of Wisconsin, Madison, WI, USA*
[+] *Carleton College, Northfield, MN, USA*

Abstract

In this paper, we introduce a new class of data mining problems called learning from aggregate views. In contrast to the traditional problem of learning from a single table of training examples, the new goal is to learn from multiple aggregate views of the underlying data, without access to the un-aggregated data. We motivate this new problem, present a general problem framework, develop learning methods for RFA (Restriction-Free Aggregate) views defined using COUNT, SUM, AVG and STDEV, and offer theoretical and experimental results that characterize the proposed methods.

1. Introduction

The standard classification problem is to learn a predictive model, such as a decision tree, from a given table **T** of training examples. In many settings, however, we do not, or sometimes *cannot*, make table **T** available in its entirety to the learning algorithm:

- *Limitations of class-label generation*: Generating a class label for each individual training example may not be feasible. For example, in ATOFMS mass spectrum labeling [7], "ground-truth" labels are too expensive to generate for individual spectra, but filters co-located with the instrument can measure various compounds of interest over a time window. These measurements are aggregations of the class label for a set of training examples.
- *Communication and Storage limitations*: In sensor networks, bandwidth is limited and only aggregated versions of the collected data can be exchanged [24]. In High Energy Physics and network log analysis, the amount of data collected per second is very large, and only aggregated data is stored and available for learning, even though a predictive model at the level of individual examples is of great potential value (e.g., to decide whether to store a future HEP event, or whether to classify a network packet as an intrusion attempt).
- *Privacy preservation and information hiding*: Privacy policies limit what data can be revealed, and data about individuals is especially sensitive. A common approach is to release aggregated views of the data [1]. Aggregation is also useful when two parties wish to collaborate while minimizing the sharing of proprietary data.

Motivated by these scenarios, we introduce the following new class of problems, called **learning from aggregate views**: *Given a set **C** of aggregate views defined (using SQL's `GROUP BY` clause and aggregate operations) over a table **U** of training examples, learn a predictive model for the class label attribute of **U**, given values for other attributes of **U**.* Observe that we consider learning from a *given* set of views, and want to build a model for the underlying table ***U***, which is *not given*.

1.1. Contributions and Future Directions

In this paper, we: (1) introduce *learning from aggregate views*, (2) develop four scalable learning methods for a special case called *learning from projections and counts*, (3) introduce a sampling-based training view transformation to extend our learning methods to *learning from RFA* (*Restriction-Free Aggregate*) *views* defined using count, sum, average and standard deviation, and (4) study our methods theoretically and describe a series of extensive experiments showing that these methods of learning from aggregated examples can usually achieve very high accuracy for individual-level predictions.

We are not aware of any prior research considering how to learn models from a set of views defined by SQL group-by queries. The special case of *learning from projections* (i.e., learning a model from multiple views defined by relational projections) has been considered, but scalable algorithms (see Section 6) are not known.

Important future directions include: (1) the *design of a set of views to publish*, given a set of resource or privacy constraints and data mining objectives, and (2) methods for more general aggregate views such as views defined using selections, in addition to grouping and aggregation.

1.2. Motivating Example

Suppose that a company wants to use Table 1 to predict a new customer's beverage preference. However, its privacy policy disallows the use of individually-identifiable data for marketing. It is well-known that the combination of date-of-birth (we use age for brevity), gender and zip can uniquely identify over 85% of US individuals. Thus, given the privacy requirement of *2-anonymity*, Table 1 cannot be revealed [30]. So, Table 1 is "rolled up" by suppressing the least significant digit for *Zip* and *Age*; let the result be **U**. Two aggregate views (shown in Table 2) are then created using the following

SQL commands, where $\boldsymbol{S}_1$ = {*Age*, *Gender*}, $\boldsymbol{S}_2$ = {*Age*, *Zip*}, $Z_1 = Z_2 =$ *Salary*, and $Y =$ *Beverage*:

$\mathbf{T}_1$ = SELECT $\boldsymbol{S}_1$, Y, COUNT(*), AVG(Z_1) FROM **U** GROUP BY $\boldsymbol{S}_1$;
$\mathbf{T}_2$ = SELECT $\boldsymbol{S}_2$, Y, COUNT(*), AVG(Z_2) FROM **U** GROUP BY $\boldsymbol{S}_2$;

The two aggregate views guarantee 2-anonymity, even though **U** does not. In general, when rolling-up the data table cannot guarantee the desired degree of privacy preservation, or it obscures too much useful information, we can complement it by creating multiple views such that each contains a subset of attributes. Also, even though Table 1 and **U** (the "rolled-up" table, not shown here) are very predictive for beverage preference, each aggregate view individually is not. The challenge we consider is how to build a predictive model for individual customers if we are given these two views but are not allowed to access the original table (Table 1).

An important complementary problem is to determine what aggregate views to publish, i.e., how to arrive at $\mathbf{T}_1$ and $\mathbf{T}_2$. This requires us to verify that publishing these two views is consistent with our privacy policy (2-anonymity, in this example), that together these views offer enough information for the objectives at hand (e.g., predicting beverage preferences, using the algorithms developed in this paper). This is an important direction for future work.

2. Learning from Aggregate Views

Assume that all training data (conceptually) comes from a *universal table* **U** (which is also called *the original table* and contains all the information we need) with a set $\boldsymbol{A} = \boldsymbol{X} \cup \{Y\}$ of attributes, where $\boldsymbol{X}$ is the set of *predictor attributes* and Y is the *class label*. Let Dom(A) denote the domain of attribute A. The problem of **learning from general aggregate views** is defined as follows:

- **Training data:** The training data consists of N *training views*: $\mathbf{T}_1, \ldots, \mathbf{T}_N$, where $\mathbf{T}_i$ is defined as:

  SELECT $\boldsymbol{S}_i$, $f_i(\boldsymbol{Z}_i)$ FROM **U**
  WHERE Ψ_i GROUP BY $\boldsymbol{S}_i$ HAVING Θ_i

  where $\boldsymbol{S}_i \subset \boldsymbol{A}$ are the *group-by attributes*; $\boldsymbol{Z}_i \subset \boldsymbol{A}$ are the *aggregated attributes* ($\boldsymbol{Z}_i - \boldsymbol{S}_i \neq \emptyset$); Ψ_i and Θ_i are selection conditions; f_i is an aggregate function.
- **Goal:** The goal is to learn a classification model $h(\boldsymbol{X})$ such that given a new example $\boldsymbol{x}$ with schema $\boldsymbol{X}$, $h(\boldsymbol{x})$ accurately outputs the class label y of $\boldsymbol{x}$.

Intuitively, we separate **U** into N (possibly overlapping) training views having schemas:

$$[\boldsymbol{S}_1, f_1(\boldsymbol{Z}_1)], [\boldsymbol{S}_2, f_2(\boldsymbol{Z}_2)], \ldots, [\boldsymbol{S}_N, f_N(\boldsymbol{Z}_N)].$$

Our goal is to learn a classifier that predicts Y based on $\boldsymbol{X}$, where each $\boldsymbol{S}_i \subset \boldsymbol{X} \cup \{Y\}$. The WHERE-clause in the view definition allows us to filter out some tuples of **U** prior to the aggregation, e.g., to consider only customers of a given gender. The HAVING-clause allows us to filter out group-level aggregated information, e.g., only allow training views to contain those aggregated results with large enough counts:

SELECT *Age*, *Zip*, *Beverage*, COUNT(*) FROM **U**
GROUP BY *Age*, *Zip*, *Beverage* HAVING COUNT(*) > 10

Table 1. Individual-level customer data table

SSN	Age	Gender	Salary	Zip	Beverage
1	25	M	$41K	55056	Beer
2	26	M	$94K	55057	Beer
3	25	F	$56K	55056	Beer
4	26	M	$67K	55057	Beer
5	37	M	$93K	55056	Beer
6	32	M	$73K	55057	Beer
7	39	F	$80K	55056	Wine
8	31	F	$70K	55057	Wine
9	31	F	$24K	11234	Wine
10	33	F	$77K	11235	Wine
11	31	M	$23K	11235	Wine
12	37	M	$53K	11233	Wine
13	28	F	$83K	11234	Beer
14	22	F	$98K	11235	Beer
15	26	M	$63K	11235	Wine
16	22	M	$99K	11233	Wine

Table 2. Two aggregate views that replace Table 1

Age	Gender	Beverage	Cnt	avgSal
2x	F	Beer	3	$79.0K
2x	M	Beer	3	$67.3K
2x	M	Wine	2	$81.0K
3x	F	Wine	4	$62.8K
3x	M	Beer	2	$83.0K
3x	M	Wine	2	$38.0K

Age	Zip	Beverage	Cnt	avgSal
2x	1123x	Beer	2	$90.5K
2x	1123x	Wine	2	$81.0K
2x	5505x	Beer	4	$64.5K
3x	1123x	Wine	4	$44.3K
3x	5505x	Beer	2	$83.0K
3x	5505x	Wine	2	$75.0K

We require that the learner: (1) has no ability to choose what views to have, (2) cannot access the original table, and (3) and cannot uniquely link tuples across views. Note that these requirements come from the application domains, e.g., learning from privacy-preserved data, and make the proposed problem unique. Also, observe that views are typically *not* independent given the class label (because $\boldsymbol{S}_i$ may overlap with $\boldsymbol{S}_j$), and exploiting this can lead to better predictions.

2.1. Learning from RFA Views

In this paper, we focus on a subclass of the problem of learning from aggregate views, in which the training views are defined without any "restriction" specified by the WHERE-clause or the HAVING-clause. We call this subclass **learning from RFA (Restriction-Free Aggregate) views**, where each training view $\mathbf{T}_i$ is of one of the following four types of **RFA views**:

SELECT $\boldsymbol{S}_i$,Y FROM **U** (**Projection View**)
SELECT $\boldsymbol{S}_i$,Y,COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y (**Count View**)
SELECT $\boldsymbol{S}_i$,AVG(Z_i),Y,COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y
SELECT $\boldsymbol{S}_i$,AVG(Z_i),STDEV(Z_i),Y,COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y

where $\boldsymbol{S}_i \subset \boldsymbol{X}$, $Z_i \in (\boldsymbol{X}–\boldsymbol{S}_i)$, and $(\cup_i \boldsymbol{S}_i) \cup (\cup_i \{Z_i\}) = \boldsymbol{X}$. Note that since we always group by $\boldsymbol{S}_i$ and Y, each view $\mathbf{T}_i$ can be identified by its schema.

This problem formulation also covers views with SUM and views with multiple AVG's (and STDEV's), because: (1) SUM(Z_i) can be transformed to AVG(Z_i) using COUNT(*), and (2) a view containing n AVG's can be projected to n single-AVG (and STDEV) views.

If all the $\mathbf{T}_i$'s are projection views or count views, the problem is called **learning from projections** or **learning from counts**, respectively. Note that these two types of views contain exactly the same information about **U**, and the two problems are essentially the same. Count views are losslessly compressed versions of projection views.

Thus, we only discuss learning from projections. In this case, the training views have the following schemas:

$$[\boldsymbol{S}_1, Y], [\boldsymbol{S}_2, Y], \ldots, [\boldsymbol{S}_N, Y].$$

The goal is to predict Y based on $\cup_i \boldsymbol{S}_i$.

In this paper, we study four methods for learning from projections (Sec 3). Two are adaptations of standard methods (Direct Ensembles and Naïve Bayes), while the others (OBE and SBE) are novel approaches that we show to be (theoretically) optimal under different assumptions. For other RFA views, our approach is to first transform them into projection views, and then learn a model from the transformed views (Sec 4).

2.2. Probabilistic Interpretation of RFA Views

Our methods are derived using a probabilistic interpretation of RFA views. Let $\boldsymbol{X} = \{X_1, \ldots, X_m\}$, and $p(\boldsymbol{X},Y)$ denote the joint probability distribution of all the attributes; i.e., $p(\boldsymbol{X},Y) = p(X_1, ..., X_m, Y)$, where X_i (or Y) is interpreted as the random variable representing the outcome of the i-th (or class label) attribute. Then, projection (and count) views can be interpreted as estimates of marginal probabilities; i.e., $\mathbf{T}_i$ is an estimate of $p(\boldsymbol{S}_i,Y)$. For example, given a projection view [*Age*, *Gender*, *Beverage*] with 3 rows [2x, M, Beer], a naïve way to estimate the probability p(2x, M, Beer) is to divide the count, 3, by the number of tuples, 16, in Table 1.

Definition 1. (Count-Estimability) *The joint probability of a set of attributes $\boldsymbol{S}$ is count-estimable with respect to table* $\mathbf{T}$ *if $p(\boldsymbol{S})$ can be estimated by*

SELECT $\boldsymbol{S}$, COUNT(*)/*grand_total* FROM $\mathbf{T}$ GROUP BY $\boldsymbol{S}$

where grand_total is the total count of table $\mathbf{T}$.

Usually $\boldsymbol{S}_i \cup \{Y\}$ is count-estimable only when $|\boldsymbol{S}_i|$ is small, because otherwise the size of the feature space would typically be much larger than the size of $\mathbf{T}_i$. Note that, except for Naïve Bayes, the methods we developed do not depend on count-estimability. We use classifiers to estimate probabilities, rather than counts or frequencies.

Next, consider a RFA view $\mathbf{T}_i$ that has AVG(Z_i) (and possibly STDEV(Z_i)). It provides extra information given by Z_i; i.e., we can estimate the probability density $p(\boldsymbol{S}_i, Z_i, Y)$ by assuming that $p(Z_i \mid \boldsymbol{S}_i, Y)$ is a normal (or other) distribution with mean AVG(Z_i) (and standard deviation STDEV(Z_i)). By the definition of conditional probability, we have $p(\boldsymbol{S}_i, Z_i, Y) = p(Z_i \mid \boldsymbol{S}_i, Y) \cdot p(\boldsymbol{S}_i,Y)$.

The goal of learning can thus be described in terms of probabilities: learn a model that accurately estimates $Score(Y \mid \boldsymbol{X}) = p(Y \mid \boldsymbol{X})$ from the $p(\boldsymbol{S}_i, Y)$'s and $p(\boldsymbol{S}_i, Z_i, Y)$'s.

Definition 2. (Optimality) *Let p^* be the underlying true distribution. We say that a classifier is optimal if for any test example $\boldsymbol{x}$, the classifier always outputs $y^* = \text{argmax}_{y \in \text{Dom}(Y)} p^*(Y=y \mid \boldsymbol{X}=\boldsymbol{x})$. We say that a classifier is strongly optimal if for any test example $\boldsymbol{x}$, it outputs the true class-probability $p^*(Y=y \mid \boldsymbol{X}=\boldsymbol{x})$.*
(argmax$_i$ $f(i)$ denotes the i that maximizes $f(i)$.)

Note that many classifiers can indeed output class-probabilities, at least heuristically. Methods to convert classifiers to probability estimators include [28, 34].

3. Methods for Learning from Projections

In this section, we describe four methods for learning from projections (and counts) and offer theoretical characterizations. We discuss their applicability to learning from other RFA views in Section 4 and present experimental evaluations in Section 5.

3.1. DE: Direct Ensemble

In Direct Ensemble, we first build a *base classifier* on each training view $\mathbf{T}_i$ using existing machine learning algorithms, e.g., decision trees. At prediction time, given a test example, each base classifier predicts the probability of each class label for this example. Then, the probabilities are combined using uniform weighting to determine the class label of the example:

$$Score_{DE}(Y \mid \boldsymbol{X}) = p(Y \mid \boldsymbol{S}_1) + \cdots + p(Y \mid \boldsymbol{S}_N),$$

where $p(Y \mid \boldsymbol{S}_i)$ denotes the class-probability outputted by the base classifier trained on $\mathbf{T}_i$. The base classifiers analyzed in this paper are decision trees [29], bagged [5] decision trees, random forests [6] and Bayes Net classifiers [8]. We use the default probability estimation methods implemented in Weka [33] to output the class-probabilities for each.

3.2. NB: Naïve Bayes

Naïve Bayes can be naturally applied to learning from projections (and counts). It is based on a set of very strong independence assumptions:

$$\forall\, X_j, X_k \in \boldsymbol{X}, j \neq k \Rightarrow (X_j \perp X_k \mid Y),$$

where $(X_j \perp X_k \mid Y)$ denotes "X_j is independent of X_k given Y". This yields the following scoring function:

$$Score_{NB}(Y \mid \boldsymbol{X}) = p(Y,\boldsymbol{X}) = p(Y) \cdot p(X_1 \mid Y) \cdots p(X_m \mid Y),$$

where $p(X_i \mid Y)$ can be easily estimated from any training view that contains X_i and Y. Although the underlying distribution usually violates these strong assumptions, Naïve Bayes classifiers work surprisingly well in practice [11, 17]. Moreover, in our setting, we have the following nice property: a Naïve Bayes model learned from projection (or count) views is exactly the same as that learned from the original table.

3.3. OBE: Ordered Bayesian Ensemble

NB and DE both have potential weaknesses:

- Naïve Bayes ignores dependencies between attributes.
- Direct Ensemble may perform badly when some training views are not predictive, because each base classifier is given the same weight regardless of whether or not it is trained on a predictive view.

Motivated by these concerns, we develop two new Bayesian ensemble methods extending DE. We describe Ordered Bayesian Ensemble (OBE) next, and Symmetric Bayesian Ensemble (SBE) in Section 3.4. Note that OBE relaxes the independence assumptions more than SBE, but is asymmetric w.r.t. view ordering.

3.3.1. OBE Motivating Example. Consider the "beverage of choice" example again. Let the training views be: $\mathbf{T}_1$ of schema [*Y*, *A*, *G*, COUNT(*)] and $\mathbf{T}_2$ of schema [*Y*, *A*, *Z*, COUNT(*)], where *A*, *G*, *Z* stand for *Age*, *Gender*, *Zip*, and *Y* is the class label *Beverage*. They provide information about $p(Y, A, G)$ and $p(Y, A, Z)$. Our goal is to learn a classifier that predicts *Y* using *A*, *G* and *Z*. From the probabilistic viewpoint, this goal is equivalent to learning the class-probability $p(Y \mid A, G, Z)$. From the definition of conditional probability,

$$p(Y \mid A, G, Z) = p(Y, A, G, Z) / p(A, G, Z).$$

If we are given a test example $\boldsymbol{x} = [a, g, z]$, then $p(A=a, G=g, Z=z)$ is constant for all class labels. Therefore we only need to consider $p(Y, A, G, Z)$, where

$$\begin{aligned} p(Y, A, G, Z) &= p(Y, A, G) \cdot p(Z \mid Y, A, G) \\ &= p(Y, A, G) \cdot p(Z \mid Y, A) \quad \text{(Assume } G \perp Z \mid A, Y\text{)} \\ &= p(A, G) \cdot p(Y \mid A, G) \cdot \frac{p(Y, A, Z)}{p(Y, A)} \\ &= p(A, G) \cdot p(Y \mid A, G) \cdot \frac{p(Y \mid A, Z) \cdot p(A, Z)}{p(Y \mid A) \cdot p(A)} \end{aligned}$$

Similarly, given a test example, $p(A,G)$, $p(A,Z)$ and $p(A)$ are also constants for all class labels. Thus, we can safely eliminate them from the formula, and obtain

$$Score(Y \mid A, G, Z) = p(Y \mid A, G) \cdot \frac{p(Y \mid A, Z)}{p(Y \mid A)}.$$

To compute the score for a test example $\boldsymbol{x} = [a, g, z]$, two base classifiers are sufficient: one trained on $\mathbf{T}_1$ that estimates $p(Y=y \mid A=a, G=g)$, and the other trained on $\mathbf{T}_2$ that estimates $p(Y=y \mid A=a, Z=z)$ and $p(Y=y \mid A=a)$. To estimate $p(Y=y \mid A=a)$, we can just use the classifier trained on $\mathbf{T}_2$ and treat *a* as a missing value; i.e., $p(Y=y \mid A=a)$ is estimated by $p(Y=y \mid A=a, Z=?)$.

Note that in the above derivation, we made an independence assumption: $(G \perp Z \mid A, Y)$. We made this assumption since there is no good way to measure the dependency between *G* and *Z* given *A* and *Y* because *G* and *Z* are in different views. Also note that the scoring function we derive is dependent on the order in which the training views are considered; i.e. if we consider $\mathbf{T}_2$ first, the scoring function will be

$$Score(Y \mid A, G, Z) = p(Y \mid A, Z) \cdot \frac{p(Y \mid A, G)}{p(Y \mid A)}.$$

Although probabilistically the two scoring functions are identical, operationally they are different because in this case we use the classifier trained on $\mathbf{T}_1$ to estimate $p(Y \mid A)$; i.e. $p(Y=y \mid A=a)$ is estimated by $p(Y=y \mid A=a, G=?)$, rather than $p(Y=y \mid A=a, Z=?)$.

3.3.2. OBE Scoring Function. We now describe the scoring mechanism for OBE in detail. Suppose that the training views are ordered as $\mathbf{T}_1$, ..., $\mathbf{T}_N$, such that the following "new attribute introduction" property holds:

$$\boldsymbol{S}_i - \left(\cup_{j=1,\ldots,i-1} \boldsymbol{S}_j\right) \neq \phi \text{, for } i = 2, \ldots, N.$$

That means each training view introduces at least one new attribute. We further define:

$$\boldsymbol{S}_i^{new} = \boldsymbol{S}_i - \left(\cup_{j=1,\ldots,i-1} \boldsymbol{S}_j\right) \text{ and } \boldsymbol{S}_i^{old} = \boldsymbol{S}_i - \boldsymbol{S}_i^{new}.$$

In other words, $\boldsymbol{S}_i^{new}$ is the set of attributes that are in $\boldsymbol{S}_i$, but not in any of the previous training views, $\boldsymbol{S}_j$, for $j = 1, \ldots, i\text{-}1$. $\boldsymbol{S}_i^{old}$ is the set of attributes that are in $\boldsymbol{S}_i$, and also in some of the previous training views.

Our goal is to learn a classifier $h(\boldsymbol{X})$, for $\boldsymbol{X} = \cup_i \boldsymbol{S}_i$. Given the OBE independence assumptions:

$$(\boldsymbol{S}_i^{new} \perp (\cup_{j=1,\ldots,i-1} \boldsymbol{S}_j) - \boldsymbol{S}_i \mid \boldsymbol{S}_i^{old}, Y) \text{, for } i = 2, \ldots, N,$$

the following scoring function gives a optimal classifier. For the proof, see Section 3.5.

$$Score_{OBE}(Y \mid \boldsymbol{X}) = p(Y \mid \boldsymbol{S}_1) \cdot \prod_{i=2,\ldots,N} \frac{p(Y \mid \boldsymbol{S}_i)}{p(Y \mid \boldsymbol{S}_i^{old})}.$$

Note that each $p(Y \mid \boldsymbol{S}_i)$, for $i = 1, \ldots, N$, is estimated by a base classifier, e.g., a decision tree, which is trained on view $\mathbf{T}_i$ and used to output the class-probabilities. Then, $p(Y \mid \boldsymbol{S}_i^{old})$ can be estimated in two different ways:

- We can use the same classifier that predicts $p(Y \mid \boldsymbol{S}_i)$ to estimate $p(Y \mid \boldsymbol{S}_i^{old})$ by treating the values in $\boldsymbol{S}_i^{new} = \boldsymbol{S}_i - \boldsymbol{S}_i^{old}$ as missing values.
- We can train another classifier to predict $p(Y \mid \boldsymbol{S}_i^{old})$ on the same view $\mathbf{T}_i$ based only on $\boldsymbol{S}_i^{old}$.

In our current implementation, we choose the first alternative because it makes the training phase of OBE exactly the same as that of Direct Ensemble.

3.3.3. OBE Algorithm. The two problems left are: (1) how to choose the orderings and (2) how to handle the case where the "new attribute introduction" property does not hold. As noted before, different orderings of training views may result in different OBE scoring functions. We just choose a small random subset of possible orderings of views. At prediction time, we compute the score for each class label for each ordering in this subset. We then compute the average score for each class label, and finally output the class label with the highest average score. Given an ordering ω of the training views, if the "new attribute introduction" property does not hold, $Score_\omega(Y \mid \boldsymbol{X})$ is computed by simply throwing away views that violate the "new attribute introduction" property.

Note that although different orderings may result in different scoring functions, the training process is exactly the same as that of Direct Ensemble. The difference between the two is in the prediction process, where OBE uses a more sophisticated technique for computing the final score. In fact, OBE evaluates the scoring function multiple times, once for each ordering.

3.4. SBE: Symmetric Bayesian Ensemble

While Naïve Bayes has overly strong independence assumptions, the ordering dependency of OBE can also be undesirable. The Symmetric Bayesian Ensemble (SBE) relaxes the strong Naïve Bayes independence assumptions and does not depend on the ordering of the training views, but its independence assumptions are stronger than those for OBE.

Let $\boldsymbol{S}_i^{unique} = \boldsymbol{S}_i - (\cup_{j \neq i} \boldsymbol{S}_j)$ be the set of attributes that appear only in $\boldsymbol{S}_i$, and $\boldsymbol{S}_i^{common} = \boldsymbol{S}_i - \boldsymbol{S}_i^{unique}$ be the set of attributes that appear in $\boldsymbol{S}_i$ and also in some other view(s). Let $\boldsymbol{X}^{common} = (\cup_{i=1,...,N} \boldsymbol{S}_i^{common})$.

Given the following SBE assumptions:

- $\boldsymbol{S}_i^{unique} \neq \varnothing$, for $i = 1, ..., N$,
- $(\boldsymbol{S}_i^{unique} \perp (\boldsymbol{X} - \boldsymbol{S}_i) \mid \boldsymbol{S}_i^{common}, Y)$, for $i = 1, ..., N$,
- $\forall X_j, X_k \in \boldsymbol{X}^{common}, j \neq k \Rightarrow (X_j \perp X_k \mid Y)$,

the following SBE scoring function is optimal.

$$Score_{SBE}(Y \mid \boldsymbol{X}) = p(Y) \cdot \left(\prod_{X_k \in \boldsymbol{X}^{common}} p(X_k \mid Y) \right) \cdot \left(\prod_{i=1}^{N} \frac{p(Y \mid \boldsymbol{S}_i)}{p(Y \mid \boldsymbol{S}_i^{common})} \right),$$

where $p(Y)$ and $p(X_k \mid Y)$ are estimated by a Naïve Bayes model, $p(Y \mid \boldsymbol{S}_i)$ is estimated by a classifier trained on $\mathbf{T}_i$, and $p(Y \mid \boldsymbol{S}_i^{common})$ is also estimated by the same classifier trained on $\mathbf{T}_i$ but by treating $\boldsymbol{S}_i^{unique}$ as missing values.

Note that if a training view does not have any unique attributes, we select several different subsets of training views such that every view in the subset has a unique attribute, and then build an SBE classifier for each subset. On prediction, we combine the scores from these SBE classifiers by uniform weighting.

3.5 Scalability

The four proposed methods are scalable, given the fact that many scalable learning algorithms were developed in the past few years (e.g., for decision trees, scalable algorithms include BOAT [18], VFDT [12]) and those scalable algorithms can be directly used to learn the base classifiers for the ensembles (DE, OBE and SBE). We also developed a cost-based optimization algorithm to improve efficiency of Naïve Bayes learning on disk-resident datasets. For lack of space, we omit the details of the Naïve Bayes optimization and detailed discussion of how to use existing scalable learning algorithms to learn base classifiers.

3.6 Optimality Results

In this section, we show that each of the four methods of learning from projections (and counts) is optimal, but under different assumptions, shedding light on their expected strengths and weaknesses.

Theorem 1. (Hansen, 1990 [20]) *Direct Ensemble asymptotically achieves 100% accuracy if the number of base classifiers is infinite and each of them makes independent mistakes*.

However, in our case, although we can obtain a large number of base classifiers by making the base classifier trained on each view itself an ensemble of many classifiers, these classifiers are highly correlated.

Theorem 2. *If the Naïve Bayes independence assumptions hold on the underlying true distribution, and $\{Y\}$ and $\{X_i, Y\}$ are count-estimable w.r.t. the training views, then the Naïve Bayes model is optimal*.

This result comes directly from the Naïve Bayes scoring function. Several other sufficient conditions exist that guarantee a different version of optimality of Naïve Bayes without requiring the Naïve Bayes independence assumption [11]. Furthermore, experiments have shown that Naïve Bayes is often a better classifier when the sample size is small, but there are other cases in which Naïve Bayes can perform badly, especially when the dependency between predictive attributes is adequately strong [11].

Theorem 3. *If $\mathbf{T}_1$, $\mathbf{T}_2$, …, $\mathbf{T}_N$ are ordered such that the "new attribute introduction" property and OBE independence assumptions hold on the underlying true distribution, and the base classifiers used by an OBE classifier are strongly optimal, then the OBE classifier using this specific ordering of views is optimal*.

Proof: Let p^* be the true underlying distribution.

$$\begin{aligned} p^*(Y, \boldsymbol{X}) &= p^*(Y, \boldsymbol{S}_1) \cdot p^*(\boldsymbol{S}_2^{new} \mid Y, \boldsymbol{S}_1) \cdots p^*(\boldsymbol{S}_N^{new} \mid Y, \cup_{j=1,...,N-1} \boldsymbol{S}_j) \\ &\qquad \text{Given} (\boldsymbol{S}_i^{new} \perp (\cup_{j=1,...,i-1} \boldsymbol{S}_j - \boldsymbol{S}_i) \mid Y, \boldsymbol{S}_i^{old}) \\ &= p^*(\boldsymbol{S}_1) \cdot p^*(Y \mid \boldsymbol{S}_1) \cdot p^*(\boldsymbol{S}_2^{new} \mid Y, \boldsymbol{S}_2^{old}) \cdots p^*(\boldsymbol{S}_N^{new} \mid Y, \boldsymbol{S}_N^{old}) \\ &= p^*(\boldsymbol{S}_1) \cdot p^*(Y \mid \boldsymbol{S}_1) \cdot \prod_{i=2,...,N} \frac{p^*(Y, \boldsymbol{S}_i)}{p^*(Y, \boldsymbol{S}_i^{old})} \\ &= p^*(\boldsymbol{S}_1) \cdot p^*(Y \mid \boldsymbol{S}_1) \cdot \prod_{i=2,...N} \frac{p^*(Y \mid \boldsymbol{S}_i) \cdot p^*(\boldsymbol{S}_i)}{p^*(Y \mid \boldsymbol{S}_i^{old}) \cdot p^*(\boldsymbol{S}_i^{old})} \end{aligned}$$

Given x, y maximizes $p^*(Y=y \mid X=x)$ iff y maximizes $p^*(Y=y, X=x)$. Since $p^*(S_i)$ and $p^*(S_i^{old})$ are constants given x, the y that maximizes $p^*(Y=y \mid X=x)$ is the y that maximizes the following formula.

$$p^*(Y = y \mid \boldsymbol{x}[\boldsymbol{S}_1]) \cdot \prod_{i=2,...N} \frac{p^*(Y = y \mid \boldsymbol{x}[\boldsymbol{S}_i])}{p^*(Y = y \mid \boldsymbol{x}[\boldsymbol{S}_i^{old}])},$$

where $\boldsymbol{x}[\boldsymbol{S}_i]$ denotes the projection of $\boldsymbol{x}$ on a set $\boldsymbol{S}_i$ of attributes. Note that when the OBE classifier has accurate probability estimates, the above formula is exactly the OBE scoring function using the ordering of $\mathbf{T}_1$, $\mathbf{T}_2$, …, $\mathbf{T}_N$. Thus, this OBE classifier is optimal. □

Theorem 4. *If every training view has at least one unique attribute, and the SBE independence assumptions hold on the underlying distribution, and the base classifiers used by an SBE classifier are strongly optimal, then the SBE classifier is optimal*.

The proof is similar to that of OBE, but uses the SBE assumptions. We omit it for lack of space.

Although we showed some sufficient conditions for the optimality of OBE and SBE, there is no easy way in

reality to tell whether the OBE or SBE assumptions hold. Also, it is hard to guarantee the accuracy of probability estimation. Thus, we view them as two extensions of Direct Ensemble based on two different probabilistic models, rather than optimal classifiers.

Finally, we summarize the four methods by reviewing them from the following two viewpoints:

- *From a probabilistic viewpoint*: Except for Direct Ensemble, the independence assumptions of the methods can be described by different types of Bayes Nets [15]. We do not directly apply Bayes Net learning techniques, however, because (1) the dependency structure would need to be learned and Bayes Net structure learning is costly, and (2) when the training data is relatively sparse in the feature space, Bayes Nets usually suffer from the lack of sufficient statistics to estimate the probabilities.
- *From an ensemble viewpoint*: Direct Ensemble, OBE and SBE are all ensembles of base classifiers trained on the views. The difference between them is in how to combine the scores of these base classifiers. In Direct Ensemble, the scores are combined naïvely by a direct summation, whereas in the two Bayesian ensembles the scores are combined based on different probabilistic models. Although ensemble techniques have been widely used in Machine Learning [5, 14], we are not aware of previous ML research on learning from multiple aggregated views.

4. Methods for RFA Views

Having developed four methods for learning from projections and counts, we now discuss how to learn from the other two types of RFA views. The key idea is to use a sampling method to transform the other two types of RFA views into projection or count views.

Suppose training view $\mathbf{T}_i$ is of one of these types:

SELECT $\boldsymbol{S}_i$, AVG(Z_i), Y, COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y;

SELECT $\boldsymbol{S}_i$, AVG(Z_i), STDEV(Z_i), Y, COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y;

Based on our probabilistic interpretation, $\mathbf{T}_i$ provides information about $p(\boldsymbol{S}_i, Z_i, Y)$. Note that Z_i is a predictor attribute. Thus, if we can transform $\mathbf{T}_i$ into:

SELECT $\boldsymbol{S}_i^+$, Y, FROM **U**,

where $\boldsymbol{S}_i^+ = \boldsymbol{S}_i \cup \{Z_i\}$, then we reduce the problem to learning from projections (and counts). Sampling is a simple yet useful transformation method. Suppose $\mathbf{T}_i$ contains STDEV(Z_i). For each tuple $\boldsymbol{t}=[\boldsymbol{x}_i, z_i^{AVG}, z_i^{STDEV}, y, c]$ in $\mathbf{T}_i$, we sample z_i^1, …, z_i^c from $p(Z_i \mid \boldsymbol{S}_i=\boldsymbol{x}_i, Y=y)$ and replace $\boldsymbol{t}$ by $[\boldsymbol{x}_i, z_i^1, y]$, …, $[\boldsymbol{x}_i, z_i^c, y]$, where $p(Z_i \mid \boldsymbol{S}_i=\boldsymbol{x}_i, Y=y)$ has mean z_i^{AVG} and standard deviation z_i^{STDEV}. If we do not have prior knowledge about $p(Z_i \mid \boldsymbol{S}_i=\boldsymbol{x}_i, Y=y)$, we assume that it is a normal distribution. If c is large, we perform low-resolution sampling to control the size expansion by allowing only K distinct Z_i-values. In this case, the resulting view is a count view.

If $\mathbf{T}_i$ does not contain STDEV(Z_i), we can still use the sampling approach, but here we only know the mean of $p(Z_i \mid \boldsymbol{S}_i=\boldsymbol{x}_i, Y=y)$. If we do not have prior knowledge about $p(Z_i \mid \boldsymbol{S}_i=\boldsymbol{x}_i, Y=y)$, we assume that it is a spike at z^{AVG}. In this case, we do not even need sampling. We can just treat $\mathbf{T}_i$ as:

SELECT $\boldsymbol{S}_i$, Z_i, Y, COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_i$,Y;

Using the sampling-based transformations in this section, learning from RFA views can be reduced to learning from projections and counts.

5. Experimental Results

For brevity, we refer to the four methods as **DE**, **NB**, **OBE**, and **SBE**. Because methods of learning from projections are the core of learning from aggregate views, we focus on evaluating them in this paper.[1] We also demonstrate the effectiveness of learning from other RFA views using a real-world mail order dataset.

Some highlights of the results are: (1) surprisingly, we can often use aggregate views, even RFA views, to build models that are as accurate as those from the original table, (2) OBE is not sensitive to its independence assumptions (but SBE is), (3) OBE or DE with random forests or bagged decision trees can be a good practical choice, and (4) the sampling approach to handling RFA views is shown to produce good predictive models with a reasonable increase in the size of the training views.

5.1. Accuracy on UCI Datasets

We experimented with UCI datasets to (1) identify good base classifiers, (2) demonstrate effectiveness of learning from projections and counts, and (3) identify when learning from projections and counts is difficult. Seventeen UCI datasets, summarized in the left half of Table 3, are used to evaluate accuracy. The base classifiers used are:

- **J48**: Weka's [33] C4.5 decision tree [29] implementation.
- **Bag**: Bootstrap aggregating (Bag) meta learner [5] based on 20 unpruned J48 trees (each tree is trained on 70% of data).
- **RF**: Random Forest [6] with 20 random trees. At each split, $\log(M+1)$ randomly chosen attributes are considered to find the best split, where M is the number of attributes.
- **K2**: A Bayes Net classifier with Bayesian scoring function and the K2 [8] search algorithm (each node has at most 5 parents).

Since the above four classifiers support learning from weighted training examples, if a training view contains

[1] We only present a small selection of results; a complete spreadsheet is available upon request.

Table 3. Experimental result on UCI datasets

Dataset	# of Cases	# of Features	# of Classes	% of numerical attributes	% of symbolic attributes	% of cases in largest class	Best accuracy on original data	Naïve Bayes		DE				OBE				SBE			
										Bag		RF		Bag		RF		Bag		RF	
								diff	std	diff	std	diff	std	diff	std	diff	std	diff	std	diff	std
breast-cancer-w	699	9	2	100	0	66	97.20	**0.00**	1.71	0.44	0.71	0.40	0.49	**-0.01**	0.68	0.15	0.48	0.07	0.67	0.01	0.50
car	1728	6	4	0	100	70	94.11	8.65	2.56	18.15	7.59	17.45	7.28	10.89	8.61	**4.54**	3.09	12.73	8.75	7.58	7.00
credit-rating	690	15	2	40	60	56	86.22	**0.00**	3.80	0.37	1.17	1.63	2.58	**0.00**	0.63	1.72	1.97	1.69	4.82	2.10	2.49
german-credit	1000	20	2	35	65	70	75.04	**0.00**	3.56	1.71	1.70	1.71	1.72	2.52	0.84	3.51	2.01	2.19	0.81	3.48	1.83
kr-vs-kp	3196	36	2	0	100	52	99.44	11.65	1.91	5.91	4.11	7.28	3.31	**3.53**	2.60	5.27	2.30	5.03	2.69	6.03	2.04
optidigits	3823	64	10	100	0	10	97.57	4.88	1.19	2.22	0.96	0.90	0.98	1.21	0.56	**0.31**	0.46	5.62	2.32	3.46	1.97
pendigits	6655	16	10	100	0	11	98.86	10.52	1.25	2.22	1.22	1.11	1.26	1.18	0.73	**0.52**	0.84	4.56	1.95	2.14	1.18
pima-diabetes	768	8	2	100	0	65	75.68	**0.42**	4.78	2.51	1.95	3.77	1.80	1.28	0.84	4.39	1.57	1.53	0.79	4.10	1.61
satimage	4435	36	6	100	0	24	90.90	9.05	1.85	0.86	0.39	0.08	0.45	0.84	0.55	**0.07**	0.58	6.58	3.26	4.85	2.20
segment	2310	19	7	100	0	14	97.97	6.82	1.72	1.36	1.42	0.89	1.57	0.82	0.98	**0.34**	0.88	5.07	6.77	2.19	1.94
spambase	4601	57	2	100	0	61	95.45	5.62	1.23	1.62	0.78	0.93	0.66	0.85	0.31	**0.76**	0.62	2.74	1.49	1.80	0.71
splice	3190	60	3	0	100	50	95.42	**0.00**	1.14	1.79	1.10	6.75	3.43	1.49	0.71	1.81	0.70	2.20	0.99	3.67	1.88
tic-tac-toe	958	9	2	0	100	65	94.79	25.15	4.40	16.27	7.71	17.14	6.02	**11.28**	5.30	13.78	4.79	13.12	3.62	14.95	2.85
vehicle	846	18	4	100	0	26	75.16	14.10	3.47	1.78	1.22	**1.07**	0.91	1.16	1.31	1.18	0.74	5.69	4.41	3.66	2.76
vowel	990	10	11	100	0	9	94.58	32.27	4.95	7.65	2.90	2.93	4.67	4.21	2.63	**1.55**	4.52	11.89	7.48	7.30	10.48
waveform	5000	40	3	100	0	33	83.53	3.56	1.44	0.31	0.79	**0.13**	0.84	0.38	0.88	0.84	0.98	1.52	1.74	1.98	1.38
yeast	1484	8	10	100	0	31	61.14	**3.53**	3.73	7.52	6.01	9.86	5.30	4.41	2.77	9.24	3.90	5.07	2.32	10.31	5.29

Table 4. T-test result for base classifier comparison

Mean(Column - Row)	DE				OBE				SBE			
	K2	Bag	J48	RF	K2	Bag	J48	RF	K2	Bag	J48	RF
K2		2.28± 0.25	-0.24±0.25	2.74±0.30		6.71±0.43	1.79±0.35	6.48±0.46		9.10±0.56	4.93±0.50	9.54±0.59
Bag	-2.82±0.25		-3.24±0.18	-0.08±0.14	-6.71±0.43		-4.92±0.23	-0.23±0.18	-9.10±0.56		-4.17±0.26	0.45±0.21
J48	0.42±0.25	3.24±0.18		3.16±0.22	-1.79±0.35	4.92±0.23		4.69±0.27	-4.93±0.50	4.17±0.26		4.62±0.33
RF	-2.74±0.30	0.08±0.14	-3.16±0.22		-6.48±0.46	0.23±0.18	-4.69±0.27		-9.54±0.59	-0.45±0.21	-4.62±0.33	

Table 5. The effect of aggregation schemes

Agg. Scheme		DE				OBE				SBE			
		Bag		RF		Bag		RF		Bag		RF	
		diff	std	diff	std	diff	std	diff	std	diff	std	diff	std
2 view	chain 0%	3.97	5.08	3.86	4.81	2.65	2.66	2.79	3.60	3.97	5.08	3.86	4.81
	chain 20%	3.45	5.05	3.14	4.77	2.11	2.46	2.10	3.66	3.45	5.05	3.14	4.77
	chain 50%	2.44	3.12	2.38	3.70	1.38	1.67	1.98	3.01	2.44	3.12	2.38	3.70
	chain 85%	1.41	1.64	1.27	2.19	1.04	1.12	0.78	1.51	1.41	1.64	1.27	2.19
	skewed	2.73	3.89	2.76	4.55	2.01	2.70	2.47	4.23	2.73	3.89	2.76	4.55
4 view	chain 0%	7.71	8.85	7.89	8.57	5.35	7.22	5.20	5.59	7.71	8.85	7.89	8.57
	chain 20%	6.88	8.69	7.17	8.21	4.89	6.90	5.04	5.28	6.88	8.69	7.17	8.21
	chain 50%	4.95	7.33	5.22	7.21	3.58	6.27	3.32	4.34	4.95	7.33	5.22	7.21
	chain 85%	4.29	6.92	4.30	7.19	2.38	3.90	2.73	4.19	4.29	6.92	4.30	7.19
	skewed	5.37	6.66	5.74	7.12	2.68	3.69	3.20	4.30	5.37	6.66	5.74	7.12

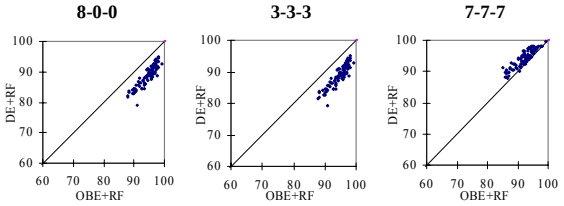

Figure 1. Scatter charts comparing DE+RF with OBE+RF. Each points (*x*, *y*) in a scatter chart represents the result on a dataset, where *x* is the accuracy of OBE+RF and *y* is the accuracy of DE+RF on that dataset.

count information, we just set the weight of each example in the view by its count.

Since the UCI datasets are not aggregated, we used two *aggregation schemes* to create N (= 2 or 4) training views from a table with an ordered set $\boldsymbol{A}$ of attributes:

- *Chained aggregation*: Pick an ordering for the training views, and ensure that a fixed fraction of the attributes in each view overlap with the next. Specifically, given the percentage of attribute overlap, $q \in [0,1]$, the number of group-by attributes for each view is about $n_t = (1+q)\cdot|\boldsymbol{A}|/N$; i.e., a fraction $1-q$ of attributes only appear in a single training view. Each consecutive pair of views (including the first and last) share $q\cdot|\boldsymbol{A}|/N$ common attributes. Compared to skewed aggregation, chain-style aggregation is a "balanced" scheme.
- *Skewed aggregation*: Pick a subset $\boldsymbol{E}$ of *emphasized attributes* to be in group-by attributes of each training view. In addition to these $\boldsymbol{E}$ attributes, each training view contains $|\boldsymbol{A}-\boldsymbol{E}|/N$ unique attributes from $\boldsymbol{A}$.

Ten-fold cross validation is used to evaluate the accuracy. Methods DE, OBE, and SBE have associated base classifiers. We combine J48, Bag, RF, and K2 with each of them, leading to 12 different techniques (e.g., DE+J48), each of which was run on data created via 12 different aggregation schemes (in which q varied from 0 to 85%, and emphasized attributes were randomly selected) for each of the 17 UCI datasets.

First, our results demonstrate that effective learning from projections and counts is achievable. The right half of Table 3 shows the difference (diff) between the best accuracy on the original table (best over J48, Bag, RF, K2) and the accuracy of a proposed aggregate learning method as indicated by the header. These differences are averaged over all aggregation schemes. The standard deviation (std) of the difference is also shown in the table. Since Bag and RF are better, we omit J48 and K2 from the table. Surprisingly, for most datasets, the best aggregate learning method is almost as good as the best model learned from the original table. The datasets on which the aggregate learning methods do not perform well are *car*, *tic-tac-toe* and *yeast*. These datasets are the ones that have very few attributes (6 to 9). That means the number of predictor attributes in each training views is around 3; this number is usually too small to learn a good model.

Second, we compare the four base classifiers. Table 4 shows the accuracy difference between pairs of base classifiers for each of DE, OBE and SBE. The differences shown are averaged over all datasets and aggregation

Table 6. T-test result: the accuracy of different RF-based method relative to Naïve Bayes and to the RF learned from the original table

Assump-tion	Max # parents	DE + RF		OBE + RF		SBE + RF	
		NB	Orig. RF	NB	Orig. RF	NB	Orig. RF
No	1	-0.64±0.14	0.60±0.17	-0.62±0.13	0.62±0.16	-0.22±0.08	1.02±0.16
	2	3.09±0.50	0.60±0.31	3.00±0.53	0.50±0.32	0.08±0.26	-2.41±0.55
	3	5.94±0.86	1.55±0.28	5.56±0.85	1.17±0.29	0.39±0.28	-4.00±0.81
	4	9.82±0.59	2.22±0.32	8.70±0.57	1.11±0.31	0.43±0.37	-7.17±0.58
	5	11.62±0.73	3.00±0.37	9.51±0.73	0.90±0.37	-1.03±0.36	-9.64±0.67
OBE	1	-0.61±0.14	0.66±0.14	-0.61±0.14	0.66±0.13	-0.26±0.09	1.01±0.17
	2	3.46±0.58	1.27±0.39	3.50±0.55	1.31±0.36	0.44±0.25	-1.74±0.45
	3	7.78±1.03	2.53±0.27	7.46±0.98	2.21±0.28	1.38±0.44	-3.87±0.64
	4	10.97±0.77	3.10±0.39	10.13±0.73	2.26±0.41	1.16±0.39	-6.70±0.66
	5	10.79±0.83	3.61±0.41	9.33±0.87	2.16±0.48	0.12±0.39	-7.05±0.67
SBE	1	-0.99±0.16	0.58±0.13	-0.87±0.14	0.70±0.14	-0.39±0.09	1.18±0.16
	2	-0.81±0.17	0.73±0.18	-0.65±0.15	0.89±0.20	-0.06±0.11	1.48±0.24
	3	-0.63±0.11	0.66±0.12	-0.74±0.12	0.55±0.13	-0.05±0.06	1.24±0.15
	4	-0.76±0.12	0.82±0.14	-1.05±0.13	0.53±0.15	-0.05±0.06	1.53±0.17
	5	-1.02±0.12	0.89±0.13	-1.32±0.13	0.59±0.15	-0.14±0.05	1.77±0.18

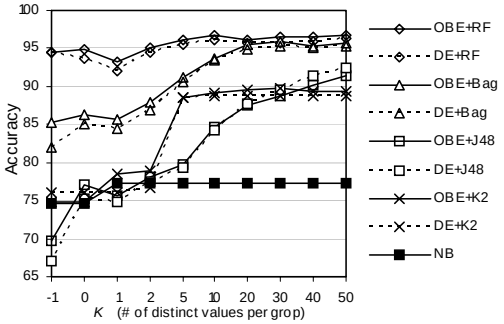

Figure 2. Accuracy curves of the aggregate learning methods

schemes. The error ranges are the 95% confidence intervals based on the two-sided *t*-test. A difference is statistically significant if the confidence interval does not include 0. The result shows that Bag and RF consistently out perform K2 and J48; thus, the widely accepted superiority of ensembles over individual tree classifiers holds for base learners in learning from counts. We see a similar trend as we drill down to the per-dataset, per-aggregation-scheme level.

Finally, we show how aggregation schemes affect the aggregate learning methods. In Table 5, "diff" is the difference between the best accuracy on the original data and the aggregate learning method indicated by the header. These differences are averaged across all 17 datasets. As we would expect, larger numbers of views result in increased information loss in the aggregation process, which leads to lower accuracy. Similarly, if we fix the number of views, a decrease in the overlap between views results in increased information loss.

5.2. Accuracy on Synthetic Datasets

To understand the characteristics of the proposed methods of learning from projections and counts, we generated synthetic datasets. Datasets generated by Bayes Nets of different structures were used to test the sensitivity of these methods to the independence assumptions. Next, since the results suggested that DE and OBE perform comparably and are superior to SBE, we created decision-tree-based datasets to compare DE and OBE in depth.

5.2.1. Bayes-Net-Based Synthetic Data. To understand how independence assumptions affect the performances of the aggregate learning methods, Bayes-Net-based synthetic datasets are generated as follows. For each dataset, we first create a Bayes Net with 19 nodes. Eighteen of them represent the predictive attributes (X_1, …, X_{18}) and the 19th is the class label (Y). The complexity of each network is controlled by two parameters: the independence assumption and the maximum number of parents for each node. “No assumption” means no independence assumption needs to hold on the network. “OBE (or SBE) assumption” means the generated network guarantees the OBE (or SBE) assumptions. Note that with the same maximum number of parents, the "no assumption" network is the most complex one. The SBE assumption results in the least complex network of the three. When a particular independence assumption is held fixed, the network grows more complex as the maximum number of parents grows. When the maximum number of parents is 1, the resulting network is, in fact, a Naïve Bayes model.

After the network is created, the CPTs (conditional probability tables) are randomly assigned. Then, 5000 examples are generated according to the network. For each set of dataset parameters, we generate 10 datasets based on 10 different networks with same parameters. For each dataset, we create 3 training views $\mathbf{T}_1$, …, $\mathbf{T}_3$, where $\mathbf{T}_1$ contains attributes Y, X_1–X_8, X_{13}, X_{14}, $\mathbf{T}_2$ contains Y, X_5–X_{12}, X_{15}, X_{16}, and $\boldsymbol{T_3}$ contains Y, X_1–X_4, X_9–X_{12}, X_{17}, X_{18}.

The result is shown in Table 6. Since Bag has similar behavior to RF, we only show the result of RF. The difference in accuracy between DE+RF and Naïve Bayes is shown in the first column; the difference between DE+RF and RF learned from the original table is shown in the second column. The comparisons of OBE+RF (and SBE+RF) with Naïve Bayes and RF learned from the original table are also shown.

First note that, in Table 6, SBE is highly sensitive to its independence assumption. When the "SBE assumptions" do not hold, SBE can perform much worse than learning from the original data. Also note that the second to last column shows that the performance of SBE is very similar to that of Naïve Bayes, even when the SBE assumptions hold. That suggests that the "SBE independence assumptions" might be as strong as Naïve Bayes (at least, on these datasets). On the other hand, DE and OBE are neither sensitive to the independence assumptions nor the complexity of the network. In fact, the performances of these two are quite comparable (also see Table 3), and clearly superior to SBE and NB (mainly from Table 6).

Interestingly, in these experiments, DE+RF and OBE+RF slightly outperform the best models learned from the original data.

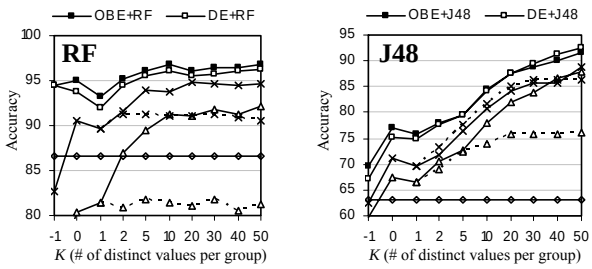

Figure 3. Accuracy curves of bases classifiers on training views (View 1 View 2 View 3)

5.2.2. Decision-Tree-Based Synthetic Data. To study when DE outperforms OBE or vice versa, we generate decision-tree-based synthetic datasets as follows. For each dataset, we randomly create a decision tree that uses 8 attributes. We then generate training examples from this tree, and add an additional 12 attributes of random noise. Each training example thus has 20 attributes. After obtaining a table of 5000 training examples, we create three aggregate views. The first view contains a random subset of attributes, in which p_1 are randomly selected from the 8 useful attributes. The second and third views contain p_2 and p_3 of these attributes, chosen by similar means. We can thus see p_1, p_2 and p_3 as parameters for our synthesized dataset. By varying these parameters, we can obtain datasets with different characteristics. For brevity, we use "p_1-p_2-p_3" to denote the parameter settings for a dataset. Ten datasets are generated for each of the following parameter settings: 8-8-8, 7-7-7, 6-6-6, 5-5-5, 4-4-4, 3-3-3, 8-0-0, 8-4-0, 6-6-0 and 8-5-2. In the interest of space, we do not show detained results, but provide the following summary and some examples in Figure 1.

- When the predictive power of the training views varies greatly (e.g., 8-0-0), OBE is better than DE.
- When none of the views is highly predictive (e.g., 3-3-3), OBE is better than DE.
- If each view is predictive (e.g., 7-7-7), DE is better.

5.3. Performance on a Real-World Dataset

We now show our results on a real-world dataset[2] with one million records. A mail order company, Deep End, wants to build a predictive model of product profitability (e.g., several levels from "good" to "bad") from information about the product and how it is presented in catalogs. Deep End wants to out-source the data mining project, but does not want to reveal the costs of its products: this is considered as a business secret. Thus, instead of the original table **U**, which has item information (*Cost*, *Category*, *Division*), presentation details (*Year*, *Media*, *Drop*, *Page*, *PriceCut*, *Focus*, *Front*, *Back*, *Seq*, *Retail*, *Area*, *Color*) and the class label (*Profitability*), Deep End gives the data mining company three views: View1 ($\mathbf{T}_1$), View2 ($\mathbf{T}_2$), Veiw3 ($\mathbf{T}_3$).

$\mathbf{T}_1$ = SELECT $\boldsymbol{S}_1$, Y, COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_1$;

$\mathbf{T}_2$ = SELECT $\boldsymbol{S}_2$,AVG(Z_2),STDEV(Z_2),Y,COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_2$,Y;

$\mathbf{T}_3$ = SELECT $\boldsymbol{S}_3$,AVG(Z_3),STDEV(Z_3),Y,COUNT(*) FROM **U** GROUP BY $\boldsymbol{S}_3$,Y;

where Y = *Profitability*, $\boldsymbol{S}_1$ = {*Year*, *Media*, *Drop*, *Page*, *PriceCut*, *Focus*, *Front*, *Back*, *Seq*, *Retail*}, $\boldsymbol{S}_2$ = {*Year*, *Media*, *Division*, *PriceCut*, *Area*, *Color*}, Z_2 = *Cost*, $\boldsymbol{S}_3$ = {*PriceCut*, *Focus*, *Front*, *Back*, *Color*, *Area*} and Z_3 = *Cost*. Note that: (1) the aggregation prevents exact cost information from being revealed, and (2) the projection ensures that no sensitive combination of attributes, e.g., *Page* and *Cost* (because knowing the page number can significantly increase the chance of determining the item that has a particular cost), is released. To measure accuracy, we randomly selected 5000 records as test examples and aggregated 900,000 non-test records to create the three training views. Then, the sampling method described in Section 4 was used to transform $\mathbf{T}_2$ and $\mathbf{T}_3$ into count views, sampling at most K distinct Z_i-values per $\boldsymbol{S}_i$-group. This process is repeated 5 times.

The average accuracies of OBE, DE and NB at different K-values are shown in Figure 2, where K=-1 means all information about Z_i is removed from the training views, K=0 means STDEV(Z_i) and COUNT(*) are removed, and K=1 means STDEV(Z_i) is removed. It can be seen that: (1) OBE+RF and DE+RF produce the best models; (2) the information that the aggregated attribute Z_i provides improves the quality of every method; (3) increasing K generally improves the accuracy; and (4) the improvement usually converges around K=20, which means we can obtain good models without expanding the training view too much.

Figure 3 shows the accuracy curves of OBE and DE (the top two solid curves) together with the accuracy curves of two base classifiers (RF and J48, since others show the same trend) on different training views (the lower three solid curves) at different K-values. It can be seen that: (1) the aggregate learning methods are always better than base classifiers trained on individual views; and (2) OBE and DE can significantly improve the accuracy when the accuracies of base classifiers are not good (note the left part of each chart).

A caution should be made that we use weighted training examples to represent the counts, but in Weka setting the weight of an example to n is actually *not* equivalent to duplicate the example to n identical copies. The dotted curves in Figure 3 are the accuracy curves for base classifiers (RF and J48) trained on examples that are generated by simply duplicating training examples K times for View 2 and View 3. We expect to see horizontal straight lines (maybe with small random vibration) like the dotted curves for RF. However, we are surprised that it is not the case for J48; i.e., simply duplicating training examples can improve the accuracy of J48 significantly

[2] The dataset is real, but the problem scenario is hypothetical and the company's name has been changed.

on this dataset. More experiments and inspection of Weka code should be made to understand this behavior.

6. Related Work and Conclusion

Learning from projections has been studied in computational learning theory, e.g., [21, 9], but this work assumes that the learner can choose desired projections. Related work in statistical disclosure control includes studies [10, 31] on bounding cell entries of contingency tables using marginals. The contingency table can be thought of as the original table, and the marginals are the projections. However, the approach is based on linear programming, in which the number of linear constraints can be exponentially large. Estimating joint probabilities from marginals is a classical problem [2, 3] in Statistics. The standard solution is iterative proportional fitting (IPF). Although several efficiency-related improvements [22, 32] have been proposed, the space requirement is proportional to the product of the domain sizes. We have tested IPF on the UCI datasets that meet IPF's space requirement without requiring discretization, using the chained aggregation scheme to create the training views (*car* with 2 training views, *tic-tac-toe* and *yeast* with 3). The 10-fold cross-validation accuracies are shown below:

Dataset	IPF	DE		OBE		SBE	
		Bag	RF	Bag	RF	Bag	RF
Car	87.39	81.89	89.87	78.24	77.37	77.66	87.39
Tic-tac-toe	74.22	84.45	84.87	87.06	84.04	87.06	83.73
yeast	59.23	52.89	54.51	56.40	57.81	45.96	59.09

Although the experiment is not extensive, the result suggests that our methods are comparable to IPF in accuracy. As to scalability, as long as the memory requirement is met, IPF runs reasonably fast. However, when we try to apply IPF to datasets with even modest numbers of attributes, the program runs out of memory.

Another related field is multi-view learning [26] or co-training [4], in which the learner is iteratively trained using multiple projections of data. However, the projections are assumed to contain keys so that tuples in different projections can be linked together, and given the class labels, the projections are assumed to be independent. Multi-relational data mining (e.g., [13]) considers the problem of learning from multiple relations, but explicit linking between records in different relations is usually assumed. Work on structure uncertainty in statistical multi-relational learning (e.g., [19]) and work (e.g., [25]) on learning SVMs from regions of feature space is also related.

To summarize, although learning from aggregated information has rich connections to a wide range of work, learning classification models from aggregated training views defined by SQL-style `GROUP BY` queries has not been studied, and we believe this is a promising research direction with many applications.

7. References

[1] N.R. Adam and J.C. Wortmann. Security-Control Methods for Statistical Databases: A Comparative Study, *ACM Computing Surveys*, 1989.

[2] W.P. Bergsma and T. Rudas. Marginal Models for Categorical Data, *Annals of Statistics*, 2002.

[3] Y. Bishop. Discrete Multivariate Analysis: Theory and Practice, *MIT Press*, 1977.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training, *COLT*, 1998.

[5] L. Breiman. Bagging Predictors. *Machine Learning*, 1996.

[6] L. Breiman. Random Forests. *Machine Learning*, 2001.

[7] L. Chen, Z. Huang and R. Ramakrishnan. Cost-based Labeling of Groups of Mass Spectra. *SIGMOD*, 2004.

[8] G.F. Cooper, E.Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data, *Machine Learning*, 1992.

[9] Eli Dichterman. Learning with Limited Visibility. CDAM Research Report, LSE-CDAM-98-0, 1998.

[10] A. Dobra and S.E. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. *Stat. J. of the UNECE*, 2001.

[11] P. Domingos and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under 0-1 Loss, *Machine Learning*, 1997.

[12] P. Domingos and G. Hulten. Mining Hight-Speed Data Streams. *SIGKDD*, 2000.

[13] S. Dzeroski and N. Lavrac. *Relational Data Mining*, 2001.

[14] Y. Freund and R.E. Schapire. Experiments with a New Boosting Algorithm. *ICML,* 1996.

[15] N. Friedman, D. Geiger and M. Goldszmidt. Bayesian network classifiers, *Machine Learning*, 1997.

[16] N. Friedman, L. Getoor, D. Koller and A. Pfeffer. Learning Probabilistic Relational Models, *IJCAI*, 1999.

[17] A. Garg and D. Roth. Understanding Probabilistic Classifiers, *EMCL* 2001.

[18] J. Gehrke, V. Ganti, R. Ramakrishnan and W-Y. Loh. BOAT – Optimistic Decision Tree Construction, *SIGMOD*, 1999.

[19] L. Getoor, D. Koller, B. Taskar and N. Friedman. Learning probabilistic relational models with structure uncertainty. *AAAI Workshop on learning statistical models from relational data*, 2000.

[20] L.K. Hansen and P. Salamon. Neural Network Ensembles. *IEEE Trans. on PAMI*, 1990.

[21] M. Kearns. Efficient Noise-Tolerant Learning from Statistical Queries. *J. ACM*, 1998.

[22] R. Jiroušek and S. Přeučil. On the Effective Implementation of the Iterative Proportional Fitting Procedure, *Computational Statistics & Data Analysis*, 1995.

[23] A.J. Knobbe, A. Siebes and B. Marseille. Involving Aggregate Functions in Multirelational Search, *PKDD*, 2002.

[24] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. *SIGMOD*, 2003.

[25] G.M. Fung, O.L. Mangasarian and J.W. Shavlik. Knowledge-based Support Vector Machine Classifiers. *NIPS* 2002.

[26] I. Muslea, S. Minton and C.A. Knoblock. Active + Semi-supervised Learning = Robust Multi-View Learning, *ICML* 2002.

[27] C. Perlich and F. Provost. Aggregation-Based Feature Invention and Relational Concept Classes, *SIGKDD* 2003.

[28] F. Provost and P. Domingos. Tree Induction for Probability-based Ranking, *Machine Learning*, 2003.

[29] J.R. Quinlan. C4.5: Programs for Machine Learning, *Morgan Kaufmann*, 1993.

[30] P. Samarati and L. Sweeney. Generalizing Data to Provide Anonymity when Disclosing Information, *PODS*, 1998.

[31] A.B. Slavkovic and S.E. Fienberg. Bounds for Cell Entries in 2-Way Tables Given Conditional Relation Frequencies. *PSD*, 2004.

[32] Y.W. Teh and M. Welling. On Improving the Efficiency of the Iterative Proportional Fitting Procedure, *AIStats*,2003.

[33] I.H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools with Java Impl., *Morgan Kaufmann*, 2000.

[34] B. Zadrozny and C. Elkan. Obtaining Probability Estimates from Decision Trees and Naïve Bayesian Classifiers, *ICML*, 2001.