A Bayesian Classifier for Learning from Tensorial Data

Wei Liu[1,2,⋆], Jeffrey Chan[1], James Bailey[1,2], Christopher Leckie[1,2], Fang Chen[2], and Kotagiri Ramamohanarao[1,2]

[1] Department of Computing and Information Systems, The University of Melbourne
[2] ATP and Victoria Research Laboratory, National ICT Australia

Abstract. Traditional machine learning methods characterize data observations by feature vectors, where an entry of a vector denotes a scalar feature value of a data instance. While this data representation facilitates the application of conventional machine learning algorithms, in many cases it is not the best way of extracting all useful information from the data observations. In this paper we relax the (often unstated) assumption of *vectorizing* features of data instances, and allow a more natural representation of the data in a tensor format. Tensors are multi-mode (aka multi-way) arrays, of whom vectors (i.e., one-mode tensors) and matrices (i.e., two-mode tensors) are special cases. We show that the tensor representation captures useful information that is difficult to provide in the conventional vector format. More importantly, to effectively utilize the rich information contained in tensors, we propose a novel semi-naive Bayesian tensor classification method (which we call `Bat`) that builds predictive models directly on data in tensor form (instead of on their vectorizations). We apply `Bat` to supervised learning problems, and perform comprehensive experiments on classifying text documents and graphs, which demonstrate *(1)* the advantage of the tensor representation over conventional feature-vectorization approaches, and *(2)* the superiority of the proposed `Bat` tensor classifier over other existing learners.

1 Introduction

A major challenge in machine learning is finding an appropriate representation to characterize observed data. Given a set of data observations (instances), traditional feature extraction methods seek to enumerate a list of features that are associated with an instance, and thus interpret an instance by a vector of feature values. This feature formulation strategy is a common, though often unstated, presumption of many supervised and unsupervised learning algorithms that build machine learning models on feature vectors [1].

In this paper, we relax the assumption that features of data observations are linearized into vectors. Instead, we allow a more natural representation of a data instance in the form of a tensor. Tensors are multi-mode (also known as

⋆ Correspondence goes to wei.liu@nicta.com.au

multi-way) arrays, whose one-mode special cases are vectors and two-mode cases are matrices.

Motivation: A major motivation for using tensors to represent instances is that they capture the "interactions" among features in different modes of the tensorial format, which are normally hard to capture using feature vectors. Although there is a rich literature in computer vision and other related fields on the study of data observations that are originally represented by tensors, such as two-mode images and three-mode videos, the more general scenarios where data observations are usually represented by vectors (such as text documents by frequent words, and graphs by frequent subgraphs) have been less well studied. To clearly motivate the use of using tensor representations, in the following we give examples to illustrate how we use *tensors of features* to describe data instances, and their advantages compared to using *vectors of features*.

In the following three motivating examples, we first explain that the tensor formulation can be applied to any conventional data set, and then introduce some examples on document and graph retrieval problems. In Examples 2 and 3, the tensor representation is capturing new information that is not explicit in the original representation, whereas in Example 1 it is capturing information that is explicit in the original representation. The different nature of the three examples is a reflection on the ability of tensor representations in addressing a diverse range of domain problems.

`Example 1` *(Data representation of a data set from an arbitrary domain).* For an arbitrary standard data set, one can first discover a set of closed frequent patterns from the data's categorical features (or discretized continuous features) as shown in Fig. 1(b). Then for each instance one can construct a tensor that uses the frequent patterns as its dimensions in each mode (shown in Fig. 1(c)). In this way, an entry in this tensor indicates whether two closed frequent patterns (if the tensor is of two modes) co-occur in the same instance. This formulation can also be generalized to construct tensors of n modes, where an entry tells whether n distinct patterns co-occur in the same instances.

A special case of the formulation in Example 1 is that one can ignore the step of discovering frequent patterns, but use each unique value of each feature as a dimension in a mode. However, in this case the number of dimensions in each mode could be extremely large even for data sets of small sizes (i.e., when the total number of all features' unique values is very large), which is impractical for use in real domains. Hence we make use of the frequent pattern discovery step, which reduces the number of dimensions in each mode and also keeps the main variance of the original data.

`Example 2` *(Data representation in document classification problems).* While a document is conventionally represented by a vector of frequent words (denoted by "Fw"), a tensor representation can capture more information than a vector. As shown in Fig. 2, a two-mode tensor contains in its diagonal entries the frequency of each Fw in the document, while its off-diagonal entries record the numbers of "pairs" of features that co-occur in the same paragraph. For example, Fw1 and Fw2 co-occur twice (in paragraph 1 and 3) so the entries at

*f*1	*f*2	*f*3	*f*4
1	1	0	1

(a) An arbitrary standard data instance characterized by a list of four binary features (*f*).

Fp1 = {*f*1=1,*f*3=0,*f*4=1}, Fp2 = {*f*1=0,*f*2=0}, Fp3 = {*f*2=1,*f*3=0}
Fp4 = {*f*2=1,*f*4=1}, Fp5 = {*f*3=1,*f*4=0}

(b) Five frequent patterns (Fp) discovered from the overall training data.

	Fp1	Fp2	Fp3	Fp4	Fp5
Fp1	1		1	1	
Fp2					
Fp3	1		1	1	
Fp4	1		1	1	
Fp5					

(c) The data instance in (a) characterized by a two-mode tensor of five frequent patterns.

Fig. 1: Vector and (two-mode) tensor representations of a data instance from an arbitrary domain. We assume binary feature values in this example.

locations (Fw1, Fw2) and (Fw2, Fw1) are 2. In the document representation, we use paragraphs to capture the relations among words, since paragraphs are natural segmentations of the original documents. We note that both local and global weighting methods, including *tf-idf* (term frequency-inverse document frequency), can be applied to the entries of tensors in the same manner to entries of Fw vectors. We also note that it is straightforward to generalize the representation of the two-mode tensor to a n-mode tensor that describes co-occurrences of n frequent words in paragraphs.

`Example 3` *(Data representation in graph classification problems).* Given a graph database, it is common for a data miner to first discover closed frequent subgraphs, and then use these to describe each graph instance [2, 3]. The example shown in Fig. 3 is a graph that contains six closed frequent subgraphs (denoted by "Sg"). Based on the graph's layout (Fig. 3(a)), conventional graph-based machine learning methods (e.g., [4–6, 2, 3]) construct a graph instance by defining the six subgraphs as features and the frequency of each subgraph found in the original graph as feature values (Fig. 3(b)). While this feature vector can provide information on the components of a graph, it does not capture relations among the components that could potentially be very useful for machine learning tasks. In this regard, we use the representation of tensors to capture both the occurrences of subgraphs and their hidden internal relations. As shown in Fig. 3(c), for the same graph instance, we can use a two-mode tensor to first capture in its diagonal entries all information stored in the instance's feature vector, and then record in the tensor's off-diagonal entries the relations/interactions among different features. Note that to enclose more information from the original graph, we give an *asymmetric* design to the tensor: the upper-diagonal entries indicate

(Paragraph 1) ⋯, Fw1, ⋯, Fw2, ⋯, Fw3, ⋯.

(Paragraph 2) ⋯,Fw4, ⋯, Fw1, ⋯, Fw5, ⋯, Fw6, ⋯.

(Paragraph 3) ⋯, Fw2, ⋯, Fw6, ⋯, Fw1, ⋯.

(a) The original layout of a text document, showing only frequent words (Fw).

Fw1	Fw2	Fw3	Fw4	Fw5	Fw6
3	2	1	1	1	2

(b) The text document in (a) characterized by a vector of frequent words.

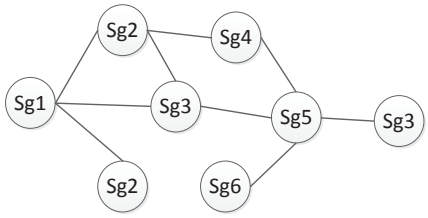
	Fw1	Fw2	Fw3	Fw4	Fw5	Fw6
Fw1	3	2	1	1	1	1
Fw2	2	2	1			1
Fw3	1	1	1			
Fw4	1			1	1	1
Fw5	1			1	1	1
Fw6	1	1		1	1	2

(c) The text document In (a) characterized by a two-mode tensor (a matrix).

Fig. 2: Vector and (two-mode) tensor representations of a text document.

how many "column-wise features" are connected to "row-wise features" and the lower-diagonal entries tell how the number of "row-wise features" that are connected to "column-wise features". For example, there are two "Sg2" connected to one "Sg1", so the upper-diagonal entry in location (Sg1,Sg2) is 2, and the lower-diagonal entry in location (Sg2,Sg1) is 1. Besides the two-mode tensor representation shown in Fig. 3(c), we note that by using the same formulation one can have a generalized n-mode tensor to describe a graph instance (e.g., a three-mode tensor can capture the triad connection among Sg1, Sg2, and Sg3 in Fig. 3(a)).

Having obtained the tensor representation of data observations, a challenge one faces is how to build machine learning models that can discover knowledge from data in the tensor format, so that one can fully utilize the rich information contained in tensors. This is a non-trivial challenge since most standard learning algorithms assume data instances are feature vectors, and it is not straightforward to apply these algorithms on tensorial data. A simple solution is to linearize the tensors into new vectors, and use the new vectors in a conventional learning algorithm. However, such tensor linearizations will *break* the relations among



(a) The original structure of a graph instance, represented by closed frequent subgraphs (Sg).

Sg1	Sg2	Sg3	Sg4	Sg5	Sg6
1	2	2	1	1	1

(b) The graph instance in (a) characterized by a vector of subgraphs.

	Sg1	Sg2	Sg3	Sg4	Sg5	Sg6
Sg1	1	2	1			
Sg2	1	2	1	1		
Sg3	1	1	2		1	
Sg4		1		1	1	
Sg5			2	1	1	1
Sg6					1	1

(c) The graph instance in (a) characterized by a two-mode tensor (a matrix).

Fig. 3: Vector and (two-mode) tensor representations of a graph instance.

features in different modes, which is equivalent to assuming independence among entries of tensors[1].

To tackle this tensor learning challenge, in this paper we proposed a generative semi-naive Bayesian classifier, which can be trained directly on data in tensor formats with respect to relations among features in different modes. The semi-naive property of our method enables the learning of inter-mode relations in an effective manner. We analyse why the assumption made in simple naive Bayes classifiers is not ideal for learning tensorial representations, and also why other existing semi-naive Bayesian classifiers are not suitable for capturing the precise information represented in tensors. These shortcomings of existing Bayesian classifiers motivate us to design a novel type of learning algorithm that can make the best use of tensorial representations.

In brief, the contributions we make in this research are as follows:

1. We propose to characterize data instances by using tensors of features, which contain much richer information than using feature vectors.

[1] This is because the ordering of entries in the linearization is not used by a classifier, but the ordering of entries in the original tensor can potentially play an important role in disclosing the relations among features in different modes.

2. We introduce a semi-naive Bayesian tensor (`Bat`) learning method, which builds classifiers by making use of the relations among features in different modes of a tensor. The `Bat` method can be applied directly to data in tensor format without tensor vectorization.
3. We apply the tensor representations and the `Bat` method to graph and document classification problems, and comprehensively evaluate our method with comparison to existing naive and semi-naive Bayesian methods.

The rest of the paper is organized as follows. We review related work in Section 2. Section 3 introduces our `Bat` method and discusses the differences between `Bat` and other existing Bayesian learners. Experimental results are presented in Section 4. We conclude in Section 5.

2 Related Work

Existing methods on tensor analysis mostly focus on decomposing tensors into factor matrices [7–9], whereas the problem of how to build classifiers directly on tensor data has not been well studied.

Tao *et al.* [10] proposed supervised tensor learning (STL) as a generalization of support vector machines, where the authors learn weight vectors separately from each mode of a data tensor. However, there is no theoretical guarantee that the weight vector learned on separate modes will provide global optima of training bias minimization. In the recent Bayesian learning literature, averaged one-dependence estimators (AODE) [11] have been introduced as a special form of the one dependence estimator (ODE), which relaxes the naive Bayes' independence assumption by making each feature a parent of other features. This method is improved by weightily averaged one-dependence estimators (WAODE) [12], which give different levels of importance to parent features by examining the mutual information between those features and the class variable. Since making each feature a parent of other features incurs very high computational costs, subsumption resolution of AODE (AODEsr) has been proposed to speed up AODE's learning process by eliminating features that are generalizations or specializations of another feature. Another way of building dependencies among features is to look at the hidden relationship between pairs of features, which gives the proposal of hidden naive Bayes (HNB) [13]. Bayesian networks are also popular ways to discover the dependencies among features, among which K2 [14] and TAN [15] have been two of the most popular methods.

3 Bayesian Tensor Classification

The strong feature-independence assumption used in NB ignores possible correlations among features. Hence when the data has multiple modes, the potentially useful interactions among features will not be taken into the classification rule of NB, which could degrade the classification performance. To address this problem, in the following we introduce a method that is specifically designed to tackle data

with multiple modes. Our proposed method belongs to the taxonomy of semi-naive Bayesian learning models [16], since it enhances the conditional probability estimation of naive Bayes by relaxing its attribute independence assumption.

In contrast to using a vector of features (e.g., $\mathbf{x}$) to represent a data observation, we describe an instance by using a n-mode tensor of features, denoted as $\mathcal{X} \in \mathbb{R}^{m \times m \ldots \times m}$. Vectors ($\mathbb{R}^m$) and matrices ($\mathbb{R}^{m \times m}$) are specifications of (1-mode and 2-mode) tensors. Without loss of generality, we present our learning method by using tensors of two modes (i.e., $\mathcal{X} \in \mathbb{R}^{m \times m}$). Then each entry of $\mathcal{X}$ can be viewed as a new feature value (denoted by $\mathcal{X}_{i,j}$ which represents the relations between features i and j), and the tensor $\mathcal{X}$ represents a set of m^2 features. To relax the assumption of conditional independence made in NB, we assign a "parent" feature to other features that share the same dimension with the parent feature. The scenario when entries share the "same dimension" of a tensor is analogical to when entries are in the "same row" or "same column" of a matrix. All entries are in the same dimension in a one-mode tensor (a vector).

The notion of the "parent" feature is the same as the concept of a parent node/vertex used in graphical models such as Bayesian networks, where features that are not connected represent variables that are conditionally independent of each other. In other words, the features that are independent in NB conditioned on the class variable will only be independent in our model given both the class and the parent. Therefore, instead of computing $P(y, \mathbf{x})$ by $P(y)P(\mathbf{x}|y)$ as in NB, we estimate $P(y, \mathbf{x})$ by

$$P_{\mathcal{X}_{p_1,p_2}}(y, \mathbf{x}) = P(y, \mathcal{X}_{p_1,p_2})P(\mathbf{x}|y, \mathcal{X}_{p_1,p_2}) = P(y, \mathcal{X}_{p_1,p_2}) \prod_{i,j=1}^{m} P(\mathcal{X}_{i,j}|y, \mathcal{X}_{p_1,p_2}) \tag{1}$$

where the first step assigns a parent feature $\mathcal{X}_{p_1,p_2} \in \mathcal{X}$, while the second step utilizes conditional feature-independence (given the class and the parent feature), where p_1 and p_2 are respectively the row and column indices. For ease of interpretation, in Eq. 1 we use $\prod_{i,j=1}^{m}$ to represent the operation $\prod_{i=1}^{m}\prod_{j=1}^{m}$. Since we assume the relation among features exists only if they share the same dimension in the tensor, for a given parent feature $\mathcal{X}_{p_1,p_2}$, the actual conditional probabilities we use are:

$$\begin{aligned} &P_{\mathcal{X}_{p_1,p_2}}(y, \mathbf{x}) \\ &= P(y, \mathcal{X}_{p_1,p_2}) \prod_{i=1}^{m} P(\mathcal{X}_{i,p_2}|y, \mathcal{X}_{p_1,p_2}) \prod_{j=1}^{m} P(\mathcal{X}_{p_1,j}|y, \mathcal{X}_{p_1,p_2})P(y) \prod_{i \neq p_1, j \neq p_2} P(\mathcal{X}_{i,j}|y) \end{aligned} \tag{2}$$

where the first two products ensure $\mathcal{X}_{p_1,p_2}$ is a parent feature of $\mathcal{X}_{i,p_2}$ and $\mathcal{X}_{p_1,j}$ only when they are from the same row or column of a two-mode tensor. And the last product means when they are not from the same row or column, we use standard NB (the naive feature-independence assumption) to compute the posterior probabilities.

To make an unbiased selection of parent features, we make each element in tensor $\mathcal{X}$ a parent at a time, and use the average of the probabilities conditioned on each parent as the final classification rule. It is reported in [12] that the performance of semi-naive Bayesian models can be improved by taking into account

the mutual information between the parent feature and the class variable. By using the definition from information theory, the mutual information $I_{i,j}$ (i.e., assume a two-mode tensor) between a feature $\mathcal{X}_{i,j}$ and the class variable y is

$$I_{i,j} = \sum_{\mathcal{X}_{i,j}} P(\mathcal{X}_{i,j}, y) \log \frac{P(\mathcal{X}_{i,j}, y)}{P(\mathcal{X}_{i,j})P(y)} \tag{3}$$

where the summation is on all unique values in feature $\mathcal{X}_{i,j}$. After applying this mutual information, the label of an instance $\mathcal{X}$ is determined by our `Bat` method using:

$$label = \arg\max_y P(y|\mathbf{x}) \propto \arg\max_y \frac{\sum_{p_1,p_2=1}^{m} I_{p_1,p_2} P_{\mathcal{X}_{p_1,p_2}}(y, \mathbf{x})}{\sum_{p_1,p_2=1}^{m} I_{p_1,p_2}} \tag{4}$$

where $P_{\mathcal{X}_{p_1,p_2}}(y, \mathbf{x})$ is defined in Eq. 2. Eq. 4 is the final classification rule of our Bayesian tensor classifier `Bat`.

For tensorial training data with two modes, the time complexity of estimating Eq. 4 is $O(2mt)$, where t is the number of training instances, and its space complexity is $O(k(mv)^2)$, where k is the number of classes and v is the average number of values in each feature.

3.1 Advantages of `Bat`

Similar to NB, `Bat` only needs to update the conditional probabilities when a new training instance becomes available, hence one advantage of `Bat` is that it is capable of incremental learning.

AODE [11] is a special case of `Bat` when each training instance is of one mode (i.e., a feature vector). However, when training instances are of more than one mode, AODE does not provide a way to handle the data. If we linearize the data tensor into a vector (like what we do to make tensor data learnable to other classifiers), AODE will have to enumerate each entry in the vector (linearized tensor) to be a parent of other entries that are in the same dimension of a tensor. This will lead to a training time complexity of $O(tm^2)$. However in `Bat`, tensor linearization is not needed, and an entry will be a parent of another entry if only they share the same row index or column index. Therefore, besides the specific focus on inter-relations of features, another advantage of `Bat` is that its training time complexity (i.e., $O(2mt)$) is an order of magnitude lower than that of AODE (i.e., $O(tm^2)$).

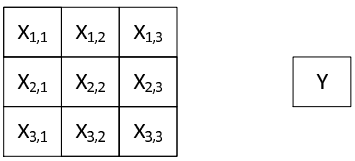


Fig. 4: An illustrative example to explain the differences among NB, AODE, and `Bat` (see Example 4 at Sec 3.1).

`Example 4` *(Differences among NB, AODE, and `Bat`).* Suppose we have a data set associated with a class variable y and a 3×3 two-mode tensor (9 features). As shown in Fig. 4, the 9 features are indexed as $\mathcal{X}_{1,1}$, $\mathcal{X}_{1,2}$, ..., $\mathcal{X}_{3,3}$.

Then taking feature $\mathcal{X}_{2,2}$ as an example, NB estimates the joint likelihood (from the Bayesian rule) of this feature and class variable simply by $P(y)P(\mathcal{X}_{2,2}|y)$, which means $\mathcal{X}_{2,2}$ is (conditionally) independent of other features. AODE estimates this joint likelihood by $\prod_{i,j=1}^{3} P(y, \mathcal{X}_{i,j})P(\mathcal{X}_{2,2}|y, \mathcal{X}_{i,j})$, which means AODE has to enumerate other features to be parent features of $\mathcal{X}_{2,2}$. This is a reflection of the fact that AODE cannot distinguish data of vector features from tensor features. In contrast, `Bat` estimates the joint likelihood by $P(y, \mathcal{X}_{1,2})P(\mathcal{X}_{2,2}|y, \mathcal{X}_{1,2})P(y, \mathcal{X}_{2,1})P(\mathcal{X}_{2,2}|y, \mathcal{X}_{2,1})P(y, \mathcal{X}_{2,3})P(\mathcal{X}_{2,2}|y, \mathcal{X}_{2,3})$ $P(y, \mathcal{X}_{3,2})P(\mathcal{X}_{2,2}|y, \mathcal{X}_{3,2})$, which means `Bat` only uses the features that are at the same column or row of the target feature (i.e., $\mathcal{X}_{2,2}$) to be parent features (i.e., $\mathcal{X}_{1,2}$, $\mathcal{X}_{2,1}$, $\mathcal{X}_{2,3}$, and $\mathcal{X}_{3,2}$). This design of `Bat` gives it the advantage of learning the specific structures of tensors.

`Bat` has a lower risk of overfitting the training data compared to AODE, since `Bat` still assumes conditional independence between features that are not in the same rows or columns (just like NB in this case), while AODE would have to assume none of the features are conditionally independent from each other given only the class variable.

The AODEsr method [17] is also closely related to `Bat`. However, different to our method, it infers the interdependence relationship by inspecting generalization/specialization or duplications among features. For example, given two features *Gender* and *Pregnant*, the feature value "*Gender=female*" is a generalization of "*Pregnant=yes*". Such types of so-called subsumption resolutions are used by AODEsr to discover highly correlated features.

Relation to Bayesian networks: `Bat` can be viewed as a special case of Bayesian networks, where the network structure is pre-defined in order to learn the underlying knowledge hidden behind the correlation among features. Such a pre-defined structure specialised by `Bat` captures the dependence among variables in a common dimension of a tensor, which is generally not captured by a vectorized format. It is possible that Bayesian networks can also learn some network structures to approximate the feature dependencies, but this would involve manual tuning on the selection of structure search algorithms. The final structure of a Bayesian network could also be too complicated to capture all useful tensorial information, which makes it infeasible in practice.

While classification error is commonly used to estimate the performance of classifiers, two other factors, namely *bias* and *variance*, can be decomposed from classification results that contribute to the error. The bias of a classifier is the difference between *the expected value (i.e., the central tendency) of the class variable returned by the classifier* and *the true values of the labels*. The variance of a classifier is the portion of the total error that is due to deviations from the expected value (i.e., the central tendency) of the classifier [18]. An ideal classifier is the one that has both low bias and low variance.

Why is `Bat` conjectured to outperform other methods? One reason that `Bat` is conjectured to outperform alternative methods is that `Bat` makes weaker conditional independence assumptions than NB, hence the bias of `Bat` is

expected to be lower[2] than that of NB. Since `Bat` considers dependencies among features from different modes, the bias of AODE is conjectured to be higher than that of `Bat`. Furthermore, because AODE assumes parent-child relationships among *all* features, its variance is conjectured to be higher than that of `Bat`. These comparisons of bias, variance, and errors are what we use to evaluate `Bat` against other classifiers in the experiment section.

Another reason that puts `Bat` at an advantage in learning tensorial data is that `Bat` uses a representation that we believe is more likely to capture dependencies between features and allows this dependency information to be used explicitly as part of the training process. Bayesian networks also could potentially learn this feature dependence, but it would require complicated structure search which could be infeasible in practice.

3.2 Limitations of `Bat`

`Bat` can only be trained on tensorial data with a *given* number of modes. The number of modes needed in a tensor to best describe the original data can be domain specific, so the proposed `Bat` is not designed with a mechanism that can automatically find the best number of modes. Another limitation is that `Bat` can only learn from categorical features values. However we note that this is also the case for all NB type classifiers, and is not a problem specific to `Bat`. Numerical feature values can be applied to `Bat` after they are discretized.

4 Evaluation

The objectives of our experiments are to evaluate *(1)* the effectiveness of the tensor formulations compared to traditional feature vectors, and *(2)* the classification performance of `Bat` compared to naive and other semi-naive Bayesian learners. We implement `Bat` in Weka [19], and evaluate our method with comparisons to NB, AODE [11], AODEsr [17], HNB [13], K2 [14], TAN [15], STL [10], logistic regression, SVMs with RBF/Sigmoid/Soft margin kernels, and decision trees. All the methods in our comparison are from Weka (version 3.6.7) and we use their default parameter settings (e.g., 10 single trees for a random forest, $\gamma = 0.01$ for SVM with RBF kernels etc.). The results are obtained from 5-fold cross validation with 10 repeated runs. We use the same bias and variance estimation method as was used in [18].

4.1 Data sets

We use two types of data sets in our experiments: graphs and text documents. Instances in all of the data sets are transformed into tensors of two modes. We note that the ways to model the relationship between subgraphs or between words are the same as the ones we demonstrated in Examples 2 and 3.

[2] However, because the classification rule of `Bat` is derived from higher dimensions than that of NB, its variance might be larger than NB.

Table 1: Statistics of graph (chemical compound) data sets. "#Inst" represents the number of instances in each data set.

Name	#Inst	#Classes	Sources
AID1481	217968	2	ATPase Inhibition
AID83	27784	2	Breast Cancer
AID81	40700	2	Colon Cancer
AID1446	217968	2	Janus Kinase
AID123	40152	2	Leukemia
AID1	40460	2	Lung Cancer
AID1531	289475	2	Mek Inhibitors
AID33	40209	2	Melanoma
AID47	40447	2	Nerve Cancer
AID109	40691	2	Ovarian Cancer
PTC-M	336	5	Mice Toxic
PTC-R	349	5	Rats Toxic

Table 2: Statistics of document data sets. "#Fw" represents the number of frequent words extracted from each data set.

Name	#Fw	#Classes	Sources
oh0	3183	10	OHSUMED collection
oh5	3013	10	OHSUMED collection
oh10	3239	10	OHSUMED collection
oh15	3101	10	OHSUMED collection
re0	2887	13	Reuters-21578
re1	3759	25	Reuters-21578
tr11	6430	9	TREC
tr12	5805	8	TREC
tr23	5833	6	TREC

We use chemical compounds as graphs where atoms in compounds are treated as nodes of graphs, and bonds that connect atoms are treated as edges of graphs. The labels of the chemical compounds are obtained from two sources: *(1)* Bioassays of anti-cancer activity and kinase inhibition (AID)[3]: the task is to predict whether a compound is positive or negative in anti-cancer activities or in kinase inhibition activities. The original data sets contain a large number of compounds (shown in Table 1). We randomly sample 1000 compounds from each data set for evaluation. *(2)* Toxicology prediction (PTC)[4]: the task is to predict the carcinogenicity of compounds on mice and rats. Each chemical compound is associated with a carcinogenicity class from {CE, SE, P, E, EE, IS, NE, N}. Following the settings of [2], we use {CE, SE, P} as positive classes, {NE, N} as negative ones, and discard other neutral classes.

The text document data sets are obtained from various sources, including the OHSUMED collection [20], the Reuters-21578 text collection[5], and the TREC repository[6]. We use the frequent words that are originally extracted by Han *et al.* [21] to construct tensors. Details of the document data sets are listed in Table 2.

[3] `http://pubchem.ncbi.nlm.nih.gov`
[4] `http://www.predictive-toxicology.org/ptc/`
[5] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`
[6] `http://trec.nist.gov`

Table 3: Performance of each classifier on data sets in vector and tensor formats.

Data sets	Average of AUC-PR on all class labels											
	NB		AODE		AODEsr		HNB		WAODE		K2	
	Vector	Tensor*	Vector	Tensor*	Vector	Tensor*	Vector	Tensor*	Vector	Tensor*	Vector	Tensor*
ATPase	.9347	.9708	.9572	.9696	.9383	.9758	.9614	.9789	.8998	.9758	.9533	.9752
Breast	.9292	.9480	.8599	.9554	.9061	.9622	.9304	.9715	.9304	.9715	.8956	.9597
Colon	.8525	.8964	.8853	.9007	.9003	.9236	.9081	.9168	.9182	.9266	.8315	.9081
Jak2	.8981	.9296	.9278	.9309	.9171	.9446	.8875	.9440	.9212	.9465	.9066	.9390
Leuk.	.8694	.9608	.9134	.9677	.9196	.9801	.9316	.9776	.9681	.9832	.8939	.9695
Lung	.8459	.9317	.8540	.9366	.9339	.9594	.8874	.9538	.9324	.9594	.8768	.9403
Mek	.9674	.9725	.9063	.9713	.8941	.9744	.9080	.9763	.9333	.9763	.9731	.9738
Mela.	.8679	.9370	.9222	.9401	.8639	.9580	.8974	.9586	.9081	.9586	.8876	.9444
Nerve	.8602	.8840	.8543	.8870	.8423	.9086	.8964	.8994	.8665	.9080	.8628	.8975
Ovar.	.8924	.9318	.8931	.9362	.9225	.9554	.9371	.9436	.8752	.9591	.8600	.9467
Mice	.8897	.9413	.8545	.9475	.9102	.9666	.9262	.9567	.9189	.9691	.9566	.9567
Rats	.8499	.9383	.9255	.9402	.9465	.9568	.9037	.9544	.8677	.9581	.9038	.9476
oh0	.3860	.4028	.6213	.6795	.7158	.7871	.6727	.7198	.7150	.7637	.1246	.1301
oh5	.3182	.3529	.5534	.5915	.6642	.7282	.6223	.6487	.6230	.6890	.1242	.1302
oh10	.4106	.4233	.6507	.6762	.6734	.7333	.6596	.7186	.7101	.7276	.1248	.1352
oh15	.3402	.3658	.6054	.6172	.6891	.7076	.6079	.6550	.6480	.6950	.1235	.1276
re0	.4763	.5103	.6447	.6735	.6616	.7035	.6092	.6745	.6815	.7018	.2071	.2247
re1	.3785	.4001	.5414	.5688	.6392	.6406	.5726	.6047	.5911	.6337	.1616	.1696
tr11	.3404	.3659	.7350	.7440	.7505	.7838	.7069	.7307	.7293	.7838	.1697	.1703
tr12	.3280	.3514	.5743	.6102	.7211	.7444	.6433	.6502	.6759	.6805	.0832	.0847
tr23	.3105	.3162	.6252	.6397	.6848	.6961	.6374	.6789	.6783	.6961	.2411	.2598
Win	16		19		17		19		18		20	
Tie	5		2		4		2		3		1	
Loss	0		0		0		0		0		0	
t-test	2×10^{-6}		5×10^{-6}		2×10^{-7}		3×10^{-8}		1×10^{-7}		1×10^{-4}	

*: The "tensor" data are *linearized*, since existing classifiers can only handle feature vectors.

Table 4: Bias of each classifier. "*F. test*" represents the Friedman significance test, which compares classifiers by their rankings.

Data sets	Bias of each learner's classification performance								
	Bat	NB	AODE	AODEsr	HNB	WAODE	K2	TAN	STL
Win	124	27	40	104	95	112	25	35	37
Tie	40	13	18	50	39	43	39	35	37
Loss	4	128	110	14	34	13	104	98	94
F. test	Base	4×10^{-6}	4×10^{-6}	0.016	2×10^{-4}	2×10^{-4}	3×10^{-5}	4×10^{-6}	4×10^{-6}

4.2 Effectiveness of Tensor Formulation

We first test the effectiveness of using tensors to represent data instances, with comparisons to the traditional way of using feature vectors. To inspect the influence of tensors, we compare the classification accuracy of different representations using the same classifier on the same data set. Since most of the existing classifiers can only handle data instances by using feature vectors, the data in tensor formats are linearized into one mode before building classifiers on them.

As some of the data sets (i.e., the document data) have multiple labels, we use average precision (AvgPrec) as the evaluation metric for a class variable. AvgPrec evaluates the ranking performance of queried objects, which is also geometrically referred to as the area under the precision-recall curve (AUC-PR) [22]. Since AvgPrec only evaluates the performance of rankings of one class label, we use the mean of the AvgPrec of all class labels to examine the performance of a classifier, which is equivalent to the mean of AUC-PR of all class labels. For

Table 5: Variance of each classifier.

Data sets	Variance of each learner's classification performance								
	Bat	NB	AODE	AODEsr	HNB	WAODE	K2	TAN	STL
Win	83	61	80	108	71	112	43	10	29
Tie	42	30	46	33	39	35	43	12	38
Loss	43	77	42	27	58	21	82	146	101
F. test	Base	0.002	1	0.016	0.827	0.049	0.016	5×10^{-5}	2×10^{-4}

Table 6: Average AUC-PR of each classifier.

Data sets	AUC-PR of each learner's classification performance								
	Bat	NB	AODE	AODEsr	HNB	WAODE	K2	TAN	STL
Win	148	27	49	125	101	129	39	38	46
Tie	9	8	10	20	13	14	11	11	12
Loss	11	133	109	23	54	25	118	119	110
F. test	Base	4×10^{-6}	4×10^{-6}	0.016	2×10^{-4}	0.016	4×10^{-6}	4×10^{-6}	2.1×10^{-4}

Table 7: Comparisons between Bat and other classifiers that are either not Bayesian based or not generative learners. "LogReg" is short for logistic regression.

Data sets	AUC-PR of each learner's classification performance						
	Bat	LogReg	RBF	Sigmoid	Soft	C4.5	Forest
Win	124	53	5	30	68	24	85
Tie	1	25	6	19	23	12	16
Loss	1	48	115	77	35	90	25
F. test	Base	8×10^{-4}	1×10^{-8}	5×10^{-5}	6×10^{-4}	1×10^{-6}	0.016

clarity, in the remainder of the paper we use AUC-PR to denote the mean of the AvgPrec of all class labels in the classification tasks.

The classifiers' performance on data in vector formats and tensor formats are shown in Table 3. For each classifier, we compare the list of AUC-PR values on all data sets between their feature vector representations and their tensor representations. Demšar *et al.* [23] have reported that t-tests are appropriate to compare *pairs* of classifiers. Hence we perform t-tests between vector and tensor data formats on each classifier, under the null hypothesis that the AUC-PR on vector and tensor formats are not significantly different. As shown in the bottom of Table 3, the p-values are all extremely small for each classifier. This suggests that the rich information contained in the tensorial format has significantly improved the performance of all classifiers.

4.3 Effectiveness of Bat Tensor Learning

In the evaluation of our Bat method, besides AUC-PR values of each classifier we also look at the biases and variances[7] that contribute to the errors of classification on each data set. Comparisons between Bat and other classifiers, in terms of bias, variance and accuracy, are respectively shown in Tables 4, 5 and 6. Classifiers that have to take data instances by their feature vectors are trained

[7] We use the same bias and variance estimation method as in [18].

on linearized tensors. Due to page limits, we only present statistics concluded from the comparisons in Tables 4 to 7.

In contrast to previous subsections where pairs of classifiers (i.e., trained on data of vector and tensor formats respectively) are compared, in this experiment we compare multiple classifiers all together with multiple data sets. In such multiple classifier comparisons, Demšar *et al.* [23] have reported that the most appropriate measure is to perform Friedman tests on the rankings of the classifiers. So we rank the classifiers on each data set by their bias, variance and AUC-PR values, and conduct Friedman tests under the null hypothesis that their rankings are not significantly different. p-values from these tests that are lower than 0.05 reject the null hypothesis with 95% confidence. In addition, we also perform t-tests in each data set separately to summarize the number of wins, ties, and losses of each classifier, under the 95% significance level.

As we can observe from Tables 4 and 5, the bias of `Bat` is almost always the lowest among all classifiers, while its variance is slightly higher than those of AODEsr and WAODE. This phenomenon confirms our preceding analysis that `Bat` reduces the bias by taking into account the interactions among features of different dimensions. It also shows that the introduction of feature interactions increases the dimensionality of the learner, which usually comes at the cost of increased variance. However as shown in the bottom of Table 5, the increased variance of `Bat` is *not significantly* different to that of AODE and HNB. Note that in these tables, the comparisons are performed for multiple classifiers on 21 datasets. So when we have 8 alternative classifiers in the evaluation, each classifier will need to be compared 21×8 times (and hence the total count of win/tie/loss is 21×8). In other words, the win/tie/loss states how often the classifier in that column scores better/neutral/worse than classifiers in any other columns.

The figures in Table 6 show that the overall errors of `Bat`, which are affected by both its bias and variance, are significantly better than all other methods in the comparison. We can also see that NB generates the worst results, which indicates that the *naive* conditional independence assumption made in NB is detrimental to the learning process when the data contains richer information in tensor formats than vector formats. It is also noticeable that AODEsr and HNB both perform better than AODE, K2 and TAN, which suggests that the "feature elimination" strategy used in AODEsr and the "hidden variable" used in HNB are more beneficial to tensorial data than using averaged dependence estimators or using Bayesian networks. However, the constraint that AODEsr and HNB require feature vectors as learning inputs ignores the relations among different modes of features, and hence limits their performance in tensorial data.

In addition, to validate the advantages of `Bat` in learning from tensorial data, we also conduct comparisons with other classifiers that are not dependent on the Bayesian setting, or are not generative classifiers. These comparisons include logistic regression, SVMs with the radial basis function kernel (RBF), Sigmoid kernel, as well as the linear soft margin kernel. We also include decision tree based methods, such as C4.5 [24] and random forests. It is important to note

that, like other existing classifiers, these non-generative or non-Bayesian models can only handle vector features (instead of tensors) and hence the experiments on these models are done on linearized tensors. The comparison results are shown in Fig. 7. It is easy to see that `Bat` is able to statistically outperform all of the other methods in the comparisons.

5 Conclusion and Future Work

In this research we propose to formulate data observations by using tensorial formats, which capture more information than traditional feature vector representations. To effectively learn from the tensorial data, we designed a novel semi-naive Bayesian tensor learner `Bat`, which builds classifiers directly on data of tensors without linearizing them into vectors. `Bat` uses feature dependence by learning the interactions of features among different modes of the training data. This gives it the advantage that it can fully utilize the rich information contained in tensorial data, which leads to much higher classification accuracy compared to existing Bayesian methods. We evaluate `Bat` using data of text documents and chemical compound graphs, whose classification results confirm the advantage of using tensor formats to represent observations and the superiority of `Bat` in learning tensorial data.

In the future, we plan to apply `Bat` to other domains such as image classification and video semantic analysis. It is also interesting to examine the performance of `Bat` when data are represented by tensors in three or even higher modes.

References

1. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. 3rd edn. Morgan Kaufmann Publishers Inc. (2011)
2. Kong, X., Yu, P.: Semi-supervised feature selection for graph classification. In: Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). (2010) 793–802
3. Kong, X., Fan, W., Yu, P.: Dual active feature and sample selection for graph classification. In: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). (2011) 654–662
4. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). (2003) 286–295
5. Fei, H., Huan, J.: Structure feature selection for graph classification. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM). (2008) 991–1000
6. Saigo, H., Nowozin, S., Kadowaki, T., Kudo, T., Tsuda, K.: gBoost: a mathematical programming approach to graph classification and regression. Machine Learning **75**(1) (2009) 69–89
7. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM review **51**(3) (2009) 455

8. Liu, W., Kan, A., Chan, J., Bailey, J., Leckie, C., Pei, J., Kotagiri, R.: On compressing weighted time-evolving graphs. In: Proceedings of CIKM'12. 2319–2322
9. Liu, W., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K.: Mining labelled tensors by discovering both their common and discriminative subspaces. In: Proceedings of SDM'13.
10. Tao, D., Li, X., Wu, X., Hu, W., S.J., M.: Supervised tensor learning. Knowledge and Information Systems **13**(1) (2007) 1–42
11. Webb, G., Boughton, J., Wang, Z.: Not so naive bayes: Aggregating one-dependence estimators. Machine Learning **58**(1) (2005) 5–24
12. Jiang, L., Zhang, H.: Weightily averaged one-dependence estimators. In: Proceedings of the 9th Pacific Rim International Conference on Artificial intelligence. (2006) 970–974
13. Jiang, L., Zhang, H. and, C.Z.: A novel Bayes model: Hidden naive Bayes. IEEE Transaction on Knowledge and Data Engineering **21**(10) (2009) 1361–1371
14. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning **9**(4) (1992) 309–347
15. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2-3) (1997) 131–163
16. Kononenko, I.: Semi-naive bayesian classifier. In: Proceedings of the European working session on learning on Machine learning, Springer (1991) 206–219
17. Zheng, F., Webb, G., Suraweera, P., Zhu, L.: Subsumption resolution: an efficient and effective technique for semi-naive bayesian learning. Machine Learning **87**(1) (2012) 93–125
18. Webb, G.: Multiboosting: A technique for combining boosting and wagging. Machine Learning **40**(2) (2000) 159–196
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: an update. SIGKDD Explorations Newsletter **11**(1) (2009) 10–18
20. Hersh, W., Buckley, C., Leone, T., Hickam, D.: Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of SIGIR. (1994) 192–201
21. Han, E., Karypis, G.: Centroid-based document classification: Analysis and experimental results. In: Proceeding of he European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). (2000) 424–431
22. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning (ICML). (2006) 233–240
23. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30
24. Quinlan, J.R.: C4.5: programs for machine learning. Volume 1. Morgan kaufmann (1993)