Hierarchical Latent Knowledge Analysis for Co-occurrence Data

Hiroshi Mamitsuka MAMI@KUICR.KYOTO-U.AC.JP
Institute for Chemical Research, Kyoto University, Gokasho, Uji, 611-0011, Japan

Abstract

Two-mode and co-occurrence data have been frequently seen in the real world. We address the issue of predicting unknown co-occurrence events from known events. For this issue, we propose a new method that naturally combines observable co-occurrence events with their existing latent knowledge by using hierarchical latent variables. This method builds a space-efficient hierarchical latent variable model and estimates the probability parameters of the model with an EM algorithm. In this paper, we focus on a data set of protein-protein interactions, a typical co-occurrence data set, and apply our method to the problem of predicting unknown protein-protein interactions, an important research issue in current computational biology. Using a real data set of protein-protein interactions and latent knowledge, we tested the performance of our method, comparing it with other recent machine learning approaches, including support vector machines. Our experimental results show that our method clearly outperforms the predictive performance obtained by other approaches. The results indicate that both our idea of using existing knowledge as a latent variable and our methodology implementing it are effective for drastically improving the predictive performance of existing unsupervised learning approaches for co-occurrence data.

1. Introduction

A two-mode and co-occurrence data set is found in a variety of data sets in the real world, such as product pairs in purchasing records and co-occurred words in texts, etc. We address the issue of predicting unknown co-occurrence events from known co-occurrence events. For this issue, we propose a new methodology that combines such observable co-occurrence events with already existing their latent knowledge. In this methodology, we extend an existing latent variable model for co-occurrence data to a space-efficient hierarchical latent variable model and estimate the probability parameters of this model with an EM algorithm. This work is motivated by the problem of predicting protein-protein interactions, the data of which consists of two discrete-valued attributes, each value of which corresponds to a protein. A computational method for precisely predicting unknown protein-protein interactions has been strongly awaited, since the data set of protein-protein physical interactions has been accumulated by recently developed high-throughput experimental techniques but any effective method to analyze the data set has not been proposed yet. Currently available data sets of protein-protein interactions are said to contain pseudo-positive examples as well as cover only a small part of all possible interactions (von Mering et al., 2002). Thus predicting protein-protein interactions has been considered a very difficult problem. Our method in this paper would be then useful for predicting co-occurrence events in other various domains, if this method achieves a high performance for predicting protein-protein interactions.

A protein is not a simple discrete value but can be characterized biologically. That is, each protein has its background knowledge (information). In this paper, we utilize a functional classification of proteins as the background knowledge[1]. Given a functional classification of proteins, a protein usually falls into more than one class, and we cannot observe (determine) the class to only which a protein belongs. For example, 'galactokinase' has roughly three functions, i.e. carbohydrate utilization, transcriptional control and cytoplasm. The other proteins, which are interacting with galactokinase, would have some functions relating to the three functions and would differ, being dependent on the three functions. However, we cannot explicitly observe (specify) the functions, through which galactokinase is interacting with another protein, from given protein-protein interaction data only. Thus naturally a protein function class can be a latent variable of a protein when it is interacting with an-

[1]We note that any type of categorical background knowledges can be used in our framework.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

other protein, and then we propose a new hierarchical latent variable model, which can use such background knowledge as a latent variable, for predicting unknown co-occurrence events.

The purpose of this paper is to empirically evaluate the predictive performance of our proposed methodology using real data sets by comparing with the latest learning approaches, particularly support vector machines (SVMs). The data of protein-protein interactions and proteins' background knowledge, i.e. a classification of proteins, can be transformed into a table (data set) in which each record corresponds to a pair of proteins that are interacting with each other and each attribute corresponds to a pair of classes. Furthermore, each attribute value indicates whether each protein of a corresponding row falls into each class of a corresponding column or not. Using this table, we can run any other learning methods to compare with our method.

In our experiments, we used an actual data set of protein-protein interactions with a functional classification of proteins and tested the performance of our method with other methods on this data set. Experimental results have shown that our proposed method clearly outperformed other methods compared. In particular, the performance comparison between our method and each of the other methods is statistically significant for all cases in both average prediction accuracies and precisions. These results indicate that our idea of using existing background knowledge as a latent variable and our methodology implementing it are highly effective for the problem of unsupervised learning from co-occurrence data sets.

2. Methods

We use the following notation in this paper. We denote a variable by a capitalized letter, e.g. X, and the value of a corresponding variable by that same letter in lower case, e.g. x. Now let X and Y be observable random variables taking on values $x_1, ..., x_L$ and $y_1, ..., y_M$, respectively, each of which corresponds to a protein. More concretely, x and y correspond to interactor 1 and interactor 2 of protein-protein interactions, respectively. Let N be the total number of given examples (interactions). Let S and T be latent random variables taking on values $s_1, ..., s_U$ and $t_1, ..., t_V$, respectively, each of which corresponds to a class of proteins in this paper. Concretely, s corresponds to a class taken by interactor 1 and t corresponds to that by interactor 2. Let Z be a discrete-valued latent variable taking on values $z_1, ..., z_K$, each of which corresponds to a latent cluster. Let $n(x, y)$ be the value obtained from the given co-occurrence data (of protein-protein interactions)[2] as follows: If proteins x and y are interacting with each other then $n(x, y)$ is 1, otherwise it is 0. Let $v_{x,y}(s, t)$ be the value obtained from a classification of proteins as follows: If protein x belongs to protein class s and protein y belongs to protein class t then $v_{x,y}(s, t)$ is 1, otherwise it is 0. Similarly, let $u_{x,y}(s, t)$ be the value obtained from a classification of proteins as follows: $u_{x,y}(s, t) = 3$ for a 4-tuple (x, y, s, t) that satisfies both $x \in s$ and $y \in t$, 2 for (x, y, s, t) that satisfies $y \in t$ only, 1 for (x, y, s, t) that satisfies $x \in s$ only, and 0 for the other (x, y, s, t).

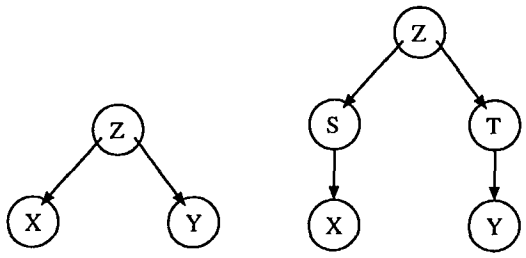

Figure 1. Graphical models of (a) AM and (b) HAM

2.1. Proposed Method: Hierarchical Latent Variable Model for Co-occurrence Data

2.1.1. Hierarchical Aspect Model (HAM)

We first explain aspect model (AM for short) which is a simple latent variable model for co-occurrence data (Hofmann, 2001). The model has been used in mainly two application fields, collaborative filtering recommender systems and semantic analysis in natural language processing (Hofmann, 2001).

An AM for X and Y with K clusters has the following form:

$$p(x, y; \theta) = \sum_{k}^{K} p(x|z_k; \theta) p(y|z_k; \theta) p(z_k; \theta).$$

The graphical model of AM is shown in Figure 1 (a). This model cannot deal with existing categorical background knowledges (e.g. classes) of two observable co-occurrence events, and a possible model which can satisfy the purpose is as follows:

$$p(x, y; \theta) = \sum_{l,m}^{U,V} p(x|s_l; \theta) p(y|t_m; \theta) p(s_l, t_m; \theta). \quad (1)$$

We need to store $p(x|s)$ in memory only when protein

[2] In estimating the parameters of our model, we can let $n(x, y)$ be the co-occurrence frequency between x and y in the given data.

x belongs to class s, because we can constantly fix $p(x|s) = 0$ if x does not belong to s. Practically, the number of classes to which an observable value (protein) belongs is limited to a certain small number, say ten or so, and thus the space complexity of $p(x|s)$ (and $p(y|t)$) is roughly linear in the number of values (proteins). However, the space complexity of $p(s,t)$ in Eq. (1) is quadratic in the number of classes. We then further extend this model to a more space-efficient hierarchical model, in which two latent variables S and T depend on another latent variable Z as follows:

$$p(s,t;\theta) = \sum_{k}^{K} p(s|z_k;\theta)p(t|z_k;\theta)p(z_k;\theta).$$

Using the $p(s,t;\theta)$, the new model for X and Y with S, T and K has the following form:

$$p(x,y;\theta) = \sum_{l,m,k}^{U,V,K} p(x|s_l;\theta)p(y|t_m;\theta)p(s_l|z_k;\theta)p(t_m|z_k;\theta)p(z_k;\theta).$$

We call the model HAM, standing for *hierarchical aspect model*, and a graphical model of HAM is shown in Figure 1 (b).

2.1.2. Estimating the Probability Parameters

When the number of clusters K and training data D are given, a possible criterion for estimating probability parameters of HAM is the maximum likelihood (ML), in which parameters are obtained to maximize the likelihood (log-likelihood) of the training data: $\theta^{ML} = arg\max_\theta \log p(D;\theta)$, where $\log p(D;\theta) = \sum_i \sum_j n(x_i,y_j) \log p(x_i,y_j;\theta)$.

In our experiments, we employ a time-efficient general scheme, called EM (Expectation-Maximization) algorithm (Dempster et al., 1977), to obtain the ML estimators of HAM. The algorithm starts with initial parameter values and iterates both an expectation step (E-step) and a maximization step (M-step) alternately until a certain convergence criterion is satisfied. The EM algorithm for a general hierarchical latent variable (or mixture) model has been already shown, e.g. in Bishop and Tipping (1998).

In E-step, we estimate the latent class using the complete data log-likelihood as follows:

$$w(z_k,s_l,t_m|x_i,y_j;\theta) = \frac{p(x_i|s_l;\theta)p(y_j|t_m;\theta)p(s_l|z_k;\theta)p(t_m|z_k;\theta)p(z_k;\theta)}{p(x_i,y_j;\theta)}.$$

In M-step, we take a corresponding summation over $w(z_k,s_l,t_m|x_i,y_j;\theta)$ with $n(x_i,y_j)$:

$$\theta_{x_i|s_l} \propto \sum_{j,k,m} n(x_i,y_j)\ w(z_k,s_l,t_m|x_i,y_j;\theta_{old}).$$

$$\theta_{y_j|t_m} \propto \sum_{i,k,l} n(x_i,y_j)\ w(z_k,s_l,t_m|x_i,y_j;\theta_{old}).$$

$$\theta_{s_l|z_k} \propto \sum_{i,j,m} n(x_i,y_j)\ w(z_k,s_l,t_m|x_i,y_j;\theta_{old}).$$

$$\theta_{t_m|z_k} \propto \sum_{i,j,l} n(x_i,y_j)\ w(z_k,s_l,t_m|x_i,y_j;\theta_{old}).$$

$$\theta_{z_k} \propto \sum_{i,j,l,m} n(x_i,y_j)\ w(s_l,t_m|x_i,y_j;\theta_{old}).$$

Note that in the above procedure, we do not need to store the $w(z_k,s_l,t_m|x_i,y_j;\theta)$ of E-step in memory. The required memory size of the procedure depends on only the parameters of our model, i.e. $p(x|s), p(y|t), p(s|z), p(t|z)$ and $p(z)$. Note further that K, i.e. the size of latent clusters, can be given as a certain constant, say 100, and as mentioned earlier the size of $p(x|s)$ and $p(y|t)$ is roughly linear in the number of given events (proteins). Thus the total space complexity of HAM is only linear in the number of input events. This means that for an arbitrary integer d, HAM can be straightforwardly extended to apply to d-mode and co-occurrence data with roughly linear space complexity. In this sense, HAM is a space-efficient model.

2.2. A Classification Set of Proteins

An usual classification of proteins currently available in the molecular biology field takes a tree structure, in which a leaf (terminal node) is one of the smallest classes and an internal node is the class representing all leafs dominated by the node. We examined the tree size which suits the issue of predicting physical protein-protein interactions. Here the problem is how we cut the tree, using given training data. We borrow the idea by Li and Abe (1998) which cut a thesaurus using the MDL (Minimum Description Length) principle (Rissanen, 1978) and given training data. We briefly explain the method for cutting the tree, for which a probabilistic structure is assumed, and when the tree is given, the maximum likelihood (ML) criterion is used to estimate the parameters in the tree.

2.2.1. The Description Length

We need some notation to describe the description length for a classification (i.e. tree-cut) given a set of training examples. Let C denote a classification and c denote a class in C. In other words, c is a node in the protein classification tree and C is a set of nodes. Here

a class must be neither an ancestor nor a descendant of another class of C. Let S be given examples and $n(c)$ be the number of examples fallen into c. Here, $|S| = \sum_{c \in C} n(c)$. Let $r(c)$ be the number of leafs which the node c dominates. The description length of a set of estimated parameters $\hat{\psi}$ and examples S given a tree cut C is as follows:

$$L(\hat{\psi}, S|C) = \lambda \cdot L(\hat{\psi}|C) + L(S|\hat{\psi}, C),$$

where λ is a constant. The first and second terms, called the model and data description length, respectively are given as follows:

$$L(\hat{\psi}|C) = \frac{\log |S|}{2} \quad \text{and} \quad L(S|\hat{\psi}, C) = -\sum_{i \in S} \log \hat{p}(i),$$

where the parameter $\hat{p}(i)$ is given by ML as $\hat{p}(i) = \frac{\hat{p}(c)}{r(c)}$ if $i \in c$, where $\hat{p}(c) = \frac{n(c)}{|S|}$.

In our experiments, before running the methods for predicting protein-protein interactions, we fixed λ at a certain value and obtained a set of classes which minimizes the description length.

2.2.2. Obtaining a Classification with MDL

We explain two steps for obtaining a classification with the minimum description length. We first count the number of examples which fall into each leaf in the protein classification tree. We then perform a bottom-up dynamic programming, to obtain the classification which provides the minimum description length among all possible classifications (tree-cuts) of the given tree. The algorithm utilizes the property that the data description length can be calculated independently for each subtree, and the time complexity of the algorithm is $O(R \cdot |S|)$, where R is the total number of leafs.

2.3. Methods Compared in Our Experiments

From the data of protein-protein interactions and a classification of proteins, we prepare a training data set in which each record corresponds to a pair of proteins interacting with each other, i.e. a pair (x, y) that satisfies $n(x, y) = 1$, and each attribute corresponds to a class pair. We generated two types of training data sets, which we call *data.2* and *data.4*, and we use the data.2 to check all methods compared and the data.4 to check SVMs only. In the data.2, attribute values are all binary and equal to $v_{x,y}(s, t)$, and in the data.4, attribute values correspond to $u_{x,y}(s, t)$. Let $\mathbf{x}_i$ be the i-th record of each of the data sets. The data set is a one-class data set, and we can run any unsupervised learning methods on this set.

2.3.1. Simple Probabilistic Model

We first tested a simple method, which computes the frequencies of co-occurred classes from a given data set and predicts the interaction of a given new pair using the frequencies. In the simple method, we consider two variations, in which one is averaging the frequencies (SimAve) and the other selects the maximum frequency (SimMax) for predicting interactions of a given protein pair. The procedure of the method can be summarized as follows:

1. For all s and t, $p(s, t)$ is computed as follows:

$$p(s, t) = \frac{q(s, t)}{\sum_{s,t} q(s, t)},$$

   where $q(s, t) = \sum_{i,j} n(x_i, y_j) \cdot v_{x_i, y_j}(s, t)$.

2. For any new proteins x and y, $\hat{p}(x, y)$, i.e. the probability that they are interacting with each other, is calculated as follows:

$$\text{SimAve:} \quad \hat{p}(x, y) = \frac{\sum_{x \in s, y \in t} p(s, t)}{\sum_{x \in s, y \in t} 1}.$$

$$\text{SimMax:} \quad \hat{p}(x, y) = \max_{x \in s, y \in t} p(s, t).$$

2.3.2. One-Class Support Vector Machine

We then tested 'one-class support vector machine (one-class SVM, hereafter OC.SVM)' proposed by Schölkopf et al. (2001). The goal of OC.SVM is to find a hyper plane h that separates given examples from the origin in a hyper space at a threshold ρ. For this goal, the following quadratic problem is solved:

$$\min \quad \frac{1}{2}\|h\|^2 + \frac{1}{\mu N} \sum_i \xi_i$$

subject to

$$(h \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i \quad (i = 1, \cdots, N), \quad \xi_i \geq 0,$$

where Φ be a kernel map.

In our experiments, we used LIBSVM ver.2.36, which has an option to run the OC.SVM and can be downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm/. In prediction, for each example, LIBSVM gives a binary output, indicating whether the example belongs to the class of training data or not.

2.3.3. Support Vector Classifier

We finally tested a support vector classifier (hereafter, SVC), a well known and the most high-performance

supervised learning approach. We then have to generate negative training examples, because our data set is a one-class data set. For each training data set, we randomly generated negative examples, the number of which is the same as that of positive training examples and each example of which is not in positive training examples and test examples. In our experiments, we first used SVM^{light} (Joachims, 1999), which can be downloaded from http://svmlight.joachims.org/, and then to confirm the results obtained by SVM^{light}, we used the above LIBSVM with an option of running the support vector classifier. The two softwares outputted approximately the same performance results for all cases in our experiments.

For both OC.SVM and SVC, we used linear kernels in our experiments. Other types of kernels, such as polynomial and radial basis functions, were also tested within the limits of main memory, but no significant improvements were obtained. In the next section, the experimental results of SVCs trained by data.2 and data.4 are shown as SVC.2 and SVC.4, respectively, and the results of OC.SVM are also shown in this manner.

3. Experimental Results

3.1. Data

We focus on the proteins of yeast, and used the MIPS (Mewes et al., 2002) comprehensive yeast genome database to obtain all data in our experiments. We used 3,918 proteins (ORFs), whose gene names are written in 'orf_gene_alias', which was downloaded from ftp://ftpmips.gsf.de/yeast/CYGD/. We then selected 8,651 physical protein-protein interactions[3], each protein of which is one of the above 3,918 proteins, from 'PPI_151002.tab' which was also downloaded from ftp://ftpmips.gsf.de/yeast/PPI/.

We further used the MIPS database to obtain a classification of proteins. The classification we used is the functional classification catalogue (funcat for short) of the database. The information of the catalogue is stored in two files, i.e. 'funcat.scheme' and 'funcat.100102', both of which were downloaded from ftp://ftpmips.gsf.de/yeast/catalogues/funcat/. The 'funcat.scheme' contains the name list of protein classes, and 'funcat.100102' contains the list of the protein classes to which each protein belongs. The classes shown in 'funcat.scheme' takes a tree structure, whose root has thirty children, which are the thirty most major classes in the funcat. In our experiments, we first used the thirty major classes and then increased the number of protein classes.

[3]We did not distinguish between interactor 1 and interactor 2 in the 'PPI_151002.tab' table. Thus when the table contains a pair (A, B), which indicates that interactor 1 is protein A and interactor 2 is protein B, we used not only (A, B) but also (B, A), except for the case in which A = B. Thus, in our experiments, $L = M$ and $U = V$.

3.2. Five-fold Cross Validation

The evaluation was done by five-fold cross validation. That is, we split the data set into five blocks of roughly equal size, and in each trial four out of these five blocks were used as training data, and the last block was reserved for test data. The results were then averaged over the five runs.

In order to evaluate the predictive ability of each of the methods in a supervised manner, we randomly generated negative examples not to overlap with any positive example. Co-occurrence data is unsupervised data in nature. We note that the negative examples are synthetically generated for evaluating the methods described in this paper and may not be negative examples biologically. The number of negative examples for each test data set is the same as that of positive examples. That is, we fixed the positive/negative ratio at 50:50 so that the prediction accuracy of a random guessing (and a predictor, which outputs only one label) should be minimized. We used two types of negative examples. Examples in the first type are from the 3,918 proteins in 'orf_gene_alias', and the other from the 8,651 physical protein-protein interactions. Hereafter we call the two test data sets containing the first and second type as Data-1 and Data-2, respectively. Both interactors 1 and 2 of a negative example in Data-2 correspond to two proteins contained in positive examples, and thus to discriminate positive examples from negative examples in Data-2 is more difficult than that in Data-1.

In our experiments, we tested six methods, i.e. HAM, SVC, OC.SVM, SimAve, SimMax and AM (aspect model), with varying the size of protein classes, for Data-1 and Data-2. The sizes[4] tested were 30, 100, 200, 300 and 400. We note that 400 is the maximum size of classes, in each of which at least one example of the data set used falls. Throughout the experiments, we fixed the size of K at 100 for AM and 30 for HAM. All our experiments were run on a Linux workstation with Intel Xeon 2.4GHz processor and 4Gbyte of main memory. The actual computation time of HAM was within two minutes for all cases. Thus the computational burden of HAM is extremely small.

[4]The approximate sizes are shown. Exact sizes are 30, 88, 202, 297 and 398.

Table 1. Average prediction accuracies and t-values (in parentheses) for Data-1.

#CLASSES	HAM	SVC.2	SVC.4	OC.SVM.2	OC.SVM.4	SIMAVE	SIMMAX	AM
30	**79.0**	64.2 (**27.0**)	63.4 (**46.4**)	57.9 (**37.9**)	57.8 (**40.7**)	56.9 (**30.2**)	56.7 (**32.7**)	
100	**78.3**	67.0 (**20.1**)	65.3 (**32.2**)	59.3 (**25.3**)	57.8 (**28.2**)	58.4 (**26.7**)	58.1 (**23.0**)	47.6
200	**77.6**	68.1 (**17.4**)	66.1 (**21.5**)	58.8 (**22.5**)	56.8 (**23.7**)	58.9 (**21.6**)	58.7 (**18.8**)	(**46.3**)
300	**76.6**	68.7 (**19.1**)	67.3 (**29.2**)	61.3 (**23.9**)	58.3 (**37.5**)	64.6 (**23.1**)	64.2 (**13.8**)	
400	**76.2**	68.6 (**20.2**)	67.5 (**35.6**)	61.6 (**29.0**)	57.9 (**53.1**)	64.8 (**32.8**)	64.3 (**16.5**)	

Table 2. Average prediction accuracies and t-values (in parentheses) for Data-2.

#CLASSES	HAM	SVC.2	SVC.4	OC.SVM.2	OC.SVM.4	SIMAVE	SIMMAX	AM
30	**67.2**	59.7 (**30.6**)	55.3 (**90.0**)	52.6 (**42.1**)	52.5 (**46.1**)	53.0 (**75.0**)	52.7 (**64.8**)	
100	**67.5**	61.3 (**16.3**)	55.4 (**38.2**)	53.2 (**37.6**)	51.6 (**38.8**)	53.8 (**41.9**)	52.5 (**74.7**)	49.4
200	**67.6**	62.6 (**11.6**)	56.4 (**20.1**)	52.8 (**32.0**)	50.9 (**34.5**)	55.2 (**37.6**)	53.1 (**34.7**)	(**61.9**)
300	**66.4**	62.9 (**7.11**)	57.5 (**34.1**)	55.4 (**34.9**)	52.0 (**32.6**)	60.0 (**52.2**)	58.4 (**52.5**)	
400	**65.8**	62.6 (**6.15**)	57.6 (**21.4**)	55.3 (**24.8**)	51.2 (**52.4**)	59.1 (**23.1**)	58.4 (**19.2**)	

3.2.1. ROC CURVES

We first checked ROC curves obtained by each of the methods. The ROC curve is drawn by plotting 'sensitivity' against '1 - specificity'. The sensitivity is defined as the probability of correct prediction given that the actual pair is interacting, and the specificity is defined as the probability of correct prediction given that the actual pair is not interacting.

Figure 2 shows the ROC curves obtained by all methods tested for Data-1 and Data-2, when the size of protein classes varies from 30 to 400 (The sizes are shown in the parentheses). Note that the ROC curve obtained by OC.SVM is simply given as a point because it outputs a simple binary for each example. As shown in the figure, HAM clearly outperformed all of the other five methods including OC.SVM and SVC, for all cases, and the SVC achieved the second best for all cases. The results of three unsupervised methods, i.e OC.SVM, SimAve and SimMax, are almost similar and better than AM. AM, which cannot use the protein class information, was the worst among them. AM's ROC curves are nearly on diagonal lines, and this means that the results of AM are close to those obtained by random guessing. That is, if we do not use the existing knowledge of proteins, it is truly difficult to predict unknown protein-protein interactions.

3.2.2. AVERAGE PREDICTION ACCURACIES

Next, we computed average prediction accuracies for each method. We obtained the prediction accuracy for each case by first sorting examples according to their likelihoods (or outputted scores) and then finding a threshold which discriminate positive from negative examples to maximize the discrimination (prediction) accuracy, with varying the threshold.

We further used the 't' values of the (pairwise) mean difference significance test for statistically comparing the accuracy of HAM with that of another method. The t values are calculated using the following formula: $t = \frac{|ave(D)|}{\sqrt{\frac{var(D)}{n}}}$, where we let D denote the difference between the accuracies of the two methods for each data set in our five trials, $ave(W)$ the average of W, $var(W)$ the variance of W, and n the number of data sets (five in our case). For $n = 5$, if t is greater than 4.604 then it is more than 99% statistically significant that HAM achieves a higher accuracy than the other.

Tables 1 and 2 show the average prediction accuracies and t-values (in parentheses) for HAM, SVC.2, SVC.4, OC.SVM.2, OC.SVM.4, SimAve, SimMax and AM. As shown in the tables, HAM outperformed the other methods, being statistically significant for all cases. Concretely, for Data-1 and Data-2, the best accuracies of HAM reached approximately 80% and 68% and were approximately 10 and 5% better than those of the other methods.

3.2.3. PRECISIONS AT RECALL OF 30%

We further evaluated each of the methods by using 'recall' and 'precision'. The recall is defined as the proportion of correctly predicted examples out of positive examples, and the precision is the proportion of correctly predicted examples out of those examples predicted to be positives. If we actually check the interactions of predicted pairs by wet-lab experiments, normally a relatively small subset will be chosen, for which a reasonably high precision can be achieved. In the current settings, a reasonable point is, say recall 0.3. We then checked the precision obtained by each of the method when the recall is fixed at 0.3.

Table 3. Precisions at recall of 30 % and t-values for Data-1.

#CLASSES	HAM	SVC.2	SVC.4	SIMAVE	SIMMAX	AM
30	**97.9**	83.0 (**17.4**)	75.2 (**58.3**)	61.9 (**69.4**)	53.6 (**51.6**)	
100	**98.5**	87.8 (**13.8**)	78.6 (**35.8**)	59.6 (**74.1**)	62.8 (**28.2**)	
200	**98.9**	88.2 (**22.5**)	79.8 (**59.9**)	59.1 (**60.4**)	54.9 (**37.5**)	48.1 (**98.5**)
300	**98.4**	89.6 (**16.6**)	80.7 (**38.1**)	80.3 (**38.1**)	74.4 (**21.1**)	
400	**98.2**	89.6 (**15.7**)	80.6 (**36.1**)	78.7 (**31.3**)	74.4 (**20.4**)	

Table 4. Precisions at recall of 30 % and t-values for Data-2.

#CLASSES	HAM	SVC.2	SVC.4	SIMAVE	SIMMAX	AM
30	**86.1**	73.4 (**25.1**)	60.2 (**45.1**)	56.0 (**67.1**)	48.4 (**48.1**)	
100	**88.7**	76.5 (**9.56**)	61.7 (**36.6**)	54.4 (**41.6**)	47.9 (**24.0**)	
200	**89.3**	77.6 (**16.2**)	62.1 (**50.0**)	56.5 (**51.7**)	51.4 (**24.3**)	48.8 (**40.3**)
300	**89.0**	79.8 (**13.0**)	64.0 (**58.8**)	70.3 (**64.9**)	62.9 (**29.6**)	
400	**87.9**	80.4 (**8.40**)	64.2 (**46.4**)	67.7 (**48.3**)	62.9 (**21.9**)	

Tables 3 and 4 show[5] the precisions and t-values (in parentheses) for HAM, SVC.2, SVC.4, SimAve, SimMax and AM. From the tables, HAM perfectly outperformed other methods again, and the performance advantages of HAM are also statistically significant for all cases. Overall, these results indicate that our methodology is highly effective for the problem of predicting co-occurrence events.

4. Concluding Remarks

We have proposed a new methodology that combines observable co-occurrence events with their existing latent knowledge for the problem of predicting co-occurrence events. From the experiments using actual co-occurrence data sets, we have shown that the proposed method greatly improved the performance of the existing methods for the given same data sets.

In Figure 2, all other unsupervised approaches (simple probabilistic methods and OC.SVM) achieved a nearly same performance. This implies that their performance is the current limitation of existing unsupervised learning approaches for the problem, so long as they use the tables (i.e. data.2 and data.4) transformed from the given co-occurrence data and their background knowledge. The SVC outperformed the unsupervised approaches, and its performance will be improved if we use a larger number of negative examples to train it. However, co-occurrence data is unsupervised data in nature. That is, synthetically generated negative examples are not true negative examples but just unknown. In this sense, unsupervised approaches should be applied to the data, and note that we can use any background knowledge of the co-occurrence events in our methodology. Thus we believe that our approach of using existing latent knowledge as a latent variable in hierarchical aspect model is an important key technique for improving the predictive performance of existing unsupervised learning for co-occurrence data in a variety of application domains.

[5]The results of OC.SVM are not shown because its output for each example is binary and we cannot measure the precision at recall of 30% for a set of examples.

References

Bishop, C. M., & Tipping, M. E. (1998). A hierarchical latent variable model for data visualization. *IEEE Transactions on PAMI, 20*, 281–293.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B, 39*, 1–38.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning, 42*, 177–196.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods - support vector learning*. MIT Press.

Li, H., & Abe, N. (1998). Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics, 24*, 217–244.

Mewes, H. W., et al. (2002). MIPS: A database for genomes and protein sequences. *NAR, 30*, 31–34.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*, 465–471.

Schölkopf, B., et al. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation, 13*, 1443–1471.

von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S. G., Fields, S., & Bork, P. (2002). Comparative assessment of large-scale datasets of protein-protein interactions. *Nature, 417*, 399–403.

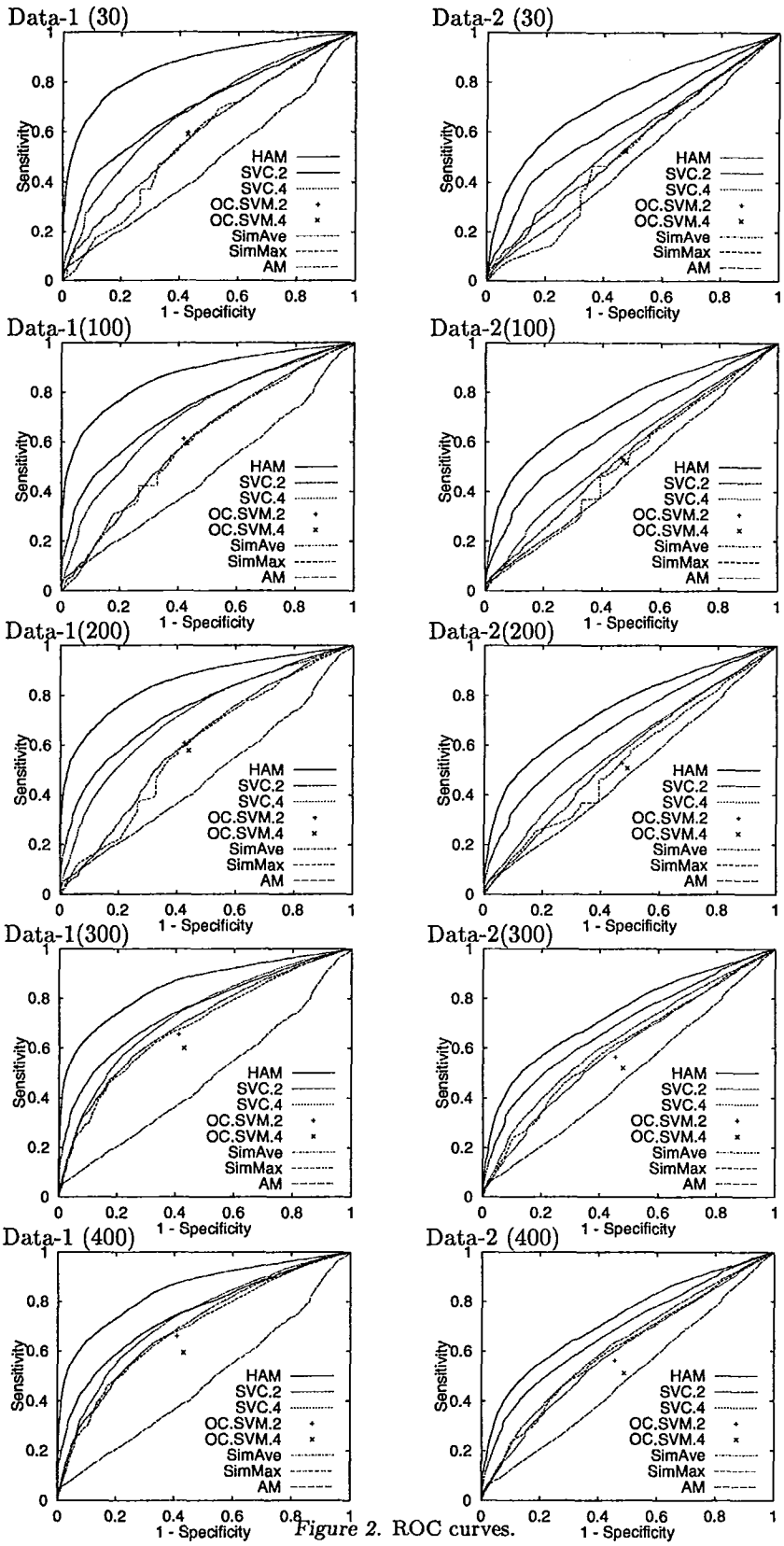

Figure 2. ROC curves.