Probabilistic User Modeling in the Presence of Drifting Concepts

Vikas Bhardwaj and Ramaswamy Devarajan

Department of Computer Science
Columbia University, New York NY 10027, USA
{vsb2108,rd2446}@columbia.edu

Abstract. We investigate supervised prediction tasks which involve multiple agents over time, in the presence of drifting concepts. The motivation behind choosing the topic is that such tasks arise in many domains which require predicting human actions. An example of such a task is recommender systems, where it is required to predict the future ratings, given features describing items and context along with the previous ratings assigned by the users. In such a system, the relationships among the features and the class values can vary over time. A common challenge to learners in such a setting is that this variation can occur both across time for a given agent, and also across different agents, (i.e. each agent behaves differently). Furthermore, the factors causing this variation are often hidden. We explore probabilistic models suitable for this setting, along with efficient algorithms to learn the model structure. Our experiments use the Netflix Prize dataset[1], a real world dataset which shows the presence of time variant concepts. The results show that the approaches we describe are more accurate than alternative approaches, especially when there is a large variation among agents. All the data and source code would be made open-source under the GNU GPL.

1 Introduction

In this article we investigate prediction problems in which there are multiple evolving entities, which we refer to as *agents*. We address two complications that frequently arise in problems that involve modeling agents: variation among (1) agents and (2) within an agent across time. Let's consider the problem of estimating the probability of a movie reviewer's rating. Here we have features describing the context such as genre, release time, reviewer's preferences etc. In this problem we may have both kinds of variation. Clearly, there will be variation among the agents, as every person is different. Also, there will be variation in the user's ratings because of his tastes, mood etc. which vary with time.

These variations pose problems to the modeler. One approach would be to learn a single model that applies to all agents. This approach has the advantage that training examples from all agents can be pooled together. Given plentiful

[1] http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 328–339, 2010.
© Springer-Verlag Berlin Heidelberg 2010

training observations for an agent, another approach is to learn a different model for each agent from only that agent's examples, and then use the appropriate agent specific model when making predictions. However, if training examples are scarce, this agent specific approach is susceptible to overfitting. Even when training data is relatively plentiful for a given agent there may be contexts that have not been observed very often.

Our approach takes the middle ground between these two extremes. Like the agent specific or heterogeneous approach, we learn a different model for each agent, but unlike it, training observations of multiple agents may have an influence on the learned probabilities for a given agent. We do this by identifying a neighborhood of agents with similar probability profiles and then combining their models if training data is scarce.

In addition to variation across agents, we consider patterns of change within a single agent over time. Problems in which class distributions change with time are said to exhibit concept drift. See, for example, Widmer [15]. Our approach to modeling concept drift utilizes hidden Markov models [12], that have found widespread use in many sequential processing tasks.

2 Related Work

We investigate the approaches to model multiple evolving users or entities over time. Our approach is closely related to previous work in Concept drift and Collaborative Filtering.

Concept Drift has received considerable attention both in the field of Data Mining [16,11,15,14] and Computational Learning Theory [1,5]. Previous work on concept drift have addressed the problem of concept drift over time, but concept drift across agents has not received particular attention. Incremental (*or online*) concept drift was discussed by [15]. Widmer and Kubat's *FLORA* framework used a window based learning system. The *FLORA2* algorithm and the *FLORA3* algorithm addressed the issue of recurring contexts. [7], addressed concept drift with the *CVFDT* algorithm, which uses flexible windowing and decision trees. [11] have also used Decision Trees for online concept learning, the algorithm OnlineTree2 introduces multiple flexible windows. Concept Drift has also been applied in the field of Network Security to model the user behavior and to use it for Anomaly Detection.[9].

Collaborative Filtering Methods have become a popular system on the Internet and are often used as techniques to complement content-based filtering systems [6]. [10] uses a Collaborative Framework based on Fuzzy Association Rules and Multiple-level Similarity (FARAMS). Recommender Systems often rely heavily on Collaborative filtering, where past transactions are analyzed to establish connections between users and products [8]. [8] introduces an approach which uses Neighborhood models and Latent factor models to do Collaborative Filtering. The filtering methods uses both implicit and explicit feedback from user (*agents*), to improve results. Current Recommender Systems consider the relations between the various users (*agents*), but do not consider the Concept Drift over time.

Other work similar to what we present here include approaches for training probabilistic models by combining examples from multiple contexts. Interpolated Markov models [3] are models that combine different order Markov models based, in part, on the number of training examples observed in each case. This approach was originally developed for finding genes in microbes. More recently variable length HMMs [14] were introduced. These models dynamically adapt the length of an HMM memory using context trees. Interpolated hidden Markov models have also been used in many domains such as finding genes and in automatic instrument recognition [13].

3 Overview

Figure 1 shows diagrams of some of the graphical probability models we use here. In this example there are N agents. For each agent we have multiple observations at various time points. We use Y_a^t to represent the output random variable for agent a at time t. Each Y is associated with M observable features, denoted by the feature vector X.

When we are learning a classifier to predict the output Y of each agent, there can be two extreme cases, (i) the Homogeneous Case, where we learn a single model for all agents (Fig. 1 A) and (ii) the Heterogeneous Case, where we learn a model for each agent, (Fig. 1 B).

Both these approaches are flawed in many real world scenarios, as there is usually some difference between the behavior of each agent, but yet its never

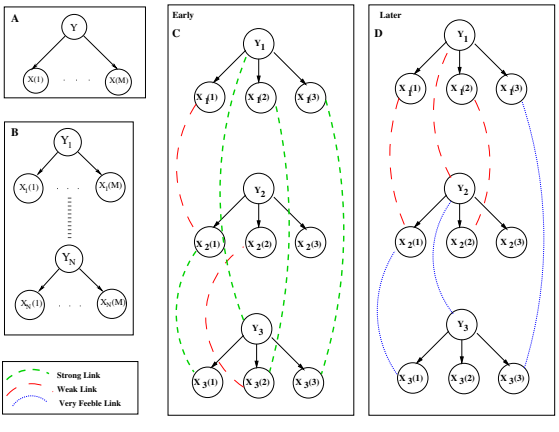

Fig. 1. Illustration of Bayesian Probabilistic Models used in our approach. Solid lines with arrows represent probabilistic dependencies. Dashed lines in C and D represent influences between agent models on learned probability parameters. **A**: A single model is learned for all agents *viz. Homogeneous Model.* **B**: A model is learned for each agent *viz. Heterogeneous Model.* **C**: An Interpolated Model near beginning of training. **D**: An Interpolated Model later during training, strength of ties have weakened, and new ties are found. Influences change with more training. See difference between C and D.

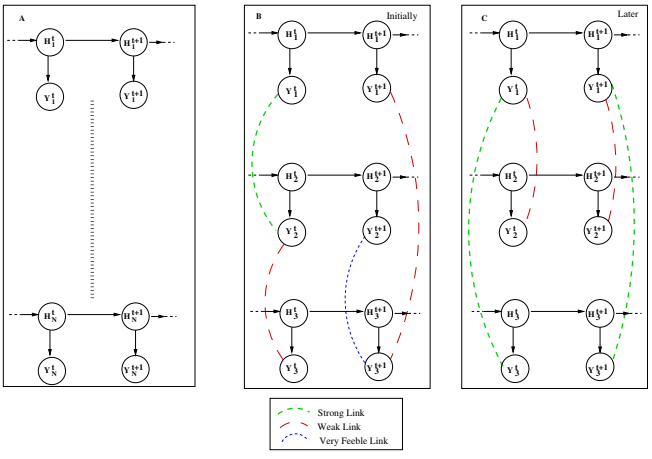

Fig. 2. Illustration of Hidden Markov Models used in our approach. **A**: An HMM is learned for each agent (*similar to the Heterogeneous Model, but captures concept drift over time*). **B**: An Interpolated HMM near beginning of training. **C**: An Interpolated HMM later during training, strength of ties have weakened, and new ties are found.

true that all agents are entirely different from each other. Thus to model predictive classifiers close to real world scenarios, we choose an approach which is somewhere in between the two extreme cases.

Figure 1 C and D illustrate our approach to modeling variation among agents. In this example we have three agents and a set of three features per agent. Early on during training, or when a new agent appears, and training data for individual agent is scarce, its probabilities are influenced heavily by neighboring agents. This is illustrated in the figure with dashed lines connecting the nodes. Outputs(*class*) of Agent_1 and Agent_3 are similar, but $X_{(1)}$ of Agent_1 is closer to $X_{(1)}$ of Agent_2. As we have more training data, we learn that output of Agent_1 is closely tied to Agent_2, and closeness between $X_{(3)}$'s of Agent_1 and Agent_3 has grown weaker.

To model concept drifts over time, we investigate Hidden Markov Models, as shown in Fig. 2. Figure 2 A illustrates the case where we learn one HMM for each agent. Variables H_i^t represent the Hidden Variable for Agent_i at time t.

The Hidden Variables capture the drift over time as each hidden variable relearns its probability distribution for state transitions and emissions, as new training data is encountered. Figure 2 B and C, illustrate an Interpolated HMM, in which we have a model for each agent and an agent's prediction probabilities are influenced both by its own training data and that of other agent's in its neighborhood similar to the static HMM explained above.

Thus, these Interpolated models attempt to capture the real world scenario, where there are varying relationships among agents. Agent_1 may be closer to Agent_2 in some ways and closer to Agent_3 in some ways.

4 Methods

As our setting involves multiple agents over time, we assume that we have labeled data for N agents through time t. For our models in Figure 1, we estimate a naive Bayes model for each agent.

For agent a at time t' we have the labeled data set $\{(\mathbf{x_a^t}, y_a^t) | t \leq t'\}$ where $1 \leq y_a^t \leq C$ is a discrete class value (C possible class values) with feature values $\mathbf{x_a^t} = (x_a^t(1), ..., x_a^t(M))$ where $x_a^t(j)$ is the value of feature j for an observation at time t.

Assumptions

- Throughout this article, we shall assume that each of the features $X(j)$ take discrete values $1, 2, \ldots, s_j$ where s_j is the size of the domain for $X(j)$. Our approach, however, is general and can be applied to the continuous case as well.
- We assume the number of features and their domains are constant across agents and time.
- We use a value of 0 to denote a missing feature value.
- We do not require the observations of different agents to be aligned in time.

Homogeneous and Heterogeneous Models

To learn the homogeneous model we pool all examples from all agents into a single training set $T = \{(\mathbf{x_a^t}, y_a^t) | \forall a, t < t'\}$. We compute counts $n_j(k, c)$, the number of examples in which feature j had a value of k with class c, and $n(c)$, the total number of observations of class c. Then we compute the maximum *a-posteriori* (MAP) estimates with pseudo-counts of 1 for all parameters:

$$\hat{P}(X(j) = k | Y = c) = \frac{n_j(k, c) + 1}{\left(\sum_{1 \leq k' \leq s_j} n_j(k', c)\right) + s_j} \tag{1}$$

and

$$\hat{P}(Y = c) = \frac{n(c) + 1}{\left(\sum_{1 \leq c' \leq C} n(c')\right) + C} \tag{2}$$

Here we have dropped the agent subscripts because we use the same model for each agent.

Learning in the heterogeneous case is the same as the homogeneous case with the exception that each agent's model is learned from a training set consisting of only that agent's examples.

Interpolated Model

Given plentiful training data (and assuming stationary concepts) we would expect the heterogeneous model $\hat{P}_a(Y_a | X_a)$ to be more accurate than the homogeneous model $\hat{P}(Y|X)$ on predictions for any agent a. On the other hand, if training data for agent a is scarce there is concern of over-fitting. Even if training data for a is scarce, however, we may have many examples for other agents.

This happens if the numbers of observations for each agent is skewed so that some agents have many observations while others have few. In this case, we'd like to be able to use training examples from the other agents to help estimate the model for a. Our approach is to form a model for a by combining its own MAP estimate with the MAP estimates of other agents in the neighborhood of a. We call this the interpolated approach.

Since learning naive Bayes models from complete data is just a set of separate estimation problems, one for each conditional distribution and one for the prior of the class variable, we describe our approach in terms of estimating just a single distribution for a discrete domain.

Let W be the random variable we are estimating a distribution for (*i.e.*, $P(W)$ is either $P_a(Y)$ or $P(X_a(j)|Y_a = c)$) and K be the number of values in W's domain. We begin with the table of counts, where $n(a, k)$ is the number of observations of value k for agent a, and first compute the agent specific MAP estimates $\hat{P}_a(W)$ using equations analogous to Equations 1 or 2 (but of course using only a's observations). This estimate is identical to the estimate used in the heterogeneous model for a.

We construct the interpolated distribution for a, $\hat{P}_a^I(W)$ as a mixture of its MAP estimate and its "neighborhood" distribution $P_a^N(W)$ (described below):

$$\hat{P}_a^I(W) = \lambda_a \times \hat{P}_a(W) + (1 - \lambda_a) P_a^N(W) \tag{3}$$

where $0 \leq \lambda_a \leq 1$ is the mixing coefficient that determines the relative contributions of a's MAP distribution and its neighborhood distribution. We set λ_a using the logistic function so that it smoothly varies as its number of observations grows. M here is a tuneable parameter that controls how fast we transition from the neighborhood distribution to the MAP distribution. A large value of M causes a slow transition.

$$\lambda_a = \frac{1}{1 + \exp(M - \sum_{1 \leq k \leq K} n(a, k))} \tag{4}$$

An agent's neighborhood distribution is a combination of the MAP distributions of its *eligible* neighbors and an average agent distribution. We represent the "distance" $d(a, a')$ between agents a and a' (for estimating W) with the Kullback-Leibler divergence:

$$d(a, a') = \sum_{1 \leq k \leq K} \hat{P}_a(W = k) \log \left(\frac{\hat{P}_a(W = k)}{\hat{P}_{a'}(W = k)} \right) \tag{5}$$

Next, we form an "average agent" model $\bar{P}(W)$ that is the average of the MAP models for all agents:

$$\bar{P}(W) \propto \prod_a \hat{P}_a(W) \tag{6}$$

where the product is a normalized point-wise product of distributions. Notice that $\bar{P}$ is not the same as the estimate learned by the homogeneous model because $\bar{P}$ is influenced equally by all agents whereas in the homogeneous model

an agent's influence is proportional to how many observations it has. Our reasoning is that when little or nothing is known about an agent, which is when $\bar{P}$ has influence, we believe it is more reasonable to model it more like the average agent than the average training example.

An agent a' is an eligible neighbor of a if both (i) $\lambda_{a'} > 0.5$ and (ii) $d(a, a')$ is less than the distance between $a's$ MAP distribution $\hat{P}_a(W)$ and the agent average distribution $\bar{P}(W)$. The first condition ensures that eligible neighbors have seen enough training examples so that their MAP estimates are more likely to be reliable and the second condition prevents distant "neighbors" from having any influence. We define the neighborhood of a, $\mathcal{N}_a$, to be its D closest eligible neighbors. If there are less than D eligible neighbors then $\mathcal{N}_a$ contains all eligible neighbors, which could be zero. Finally, we construct the neighborhood distribution P_a^n by taking the normalized point-wise product of the average agent distribution with the MAP distributions of all neighbors:

$$P_a^n(W) = \left(\prod_{a' \in \mathcal{N}} \hat{P}_{a'}(W) \right) \times \bar{P}(W) \tag{7}$$

We now compute the interpolated model with Equation 3. In this way we compute all distributions for all agents. Before we move on, we stress a few important details about this approach:

- The neighborhood for a is distribution specific. It is possible that a has a different set of neighbors when estimating $P(X(i)|Y = c)$ than it has when estimating $P(X(i)|Y = c')$. Although, this can be seen as a virtue, this may be exasperating problems associated with using a generative model for discriminative classification.
- The neighborhood distribution behaves like an evolving bias or prior. Implicit in this formulation is that while agents may be diverse, the distributions tend to cluster.
- If the MAP distribution for an agent is near the center of its neighborhood the interpolated distribution will be similar to the MAP. On the other hand if most of an agent's neighbors are in the same direction then the neighborhood distributions combine to pull the interpolated distribution toward that center.

Hidden Markov Models

We now turn toward our approach for representing changes within an agent over time. For this we use hidden Markov models. We explore both a regular HMM with one model for each agent and an Interpolated version which applies the interpolated approach to an agent's HMM.

Our HMM consists of two states. These state represent different modes of the agent's distribution over class labels. In the domain of our dataset, the hidden states could correspond to changes in a reviewers' calibration or even changes in actual human reviewer associated with a given user ID, though we have no definite knowledge of what these states maybe, we can learn their behavior. To learn the parameters of the HMM for agent a at time t we train an HMM using

the expectation-maximization algorithm [4] using the single sequence $y_a^1, ..., y_a^t$. Thus, this approach is similar to the heterogeneous method, in which each agent's model is trained using only examples from that agent. But, of course, it is different, as here we allow the agent's probability distribution over the class to change with time.

In the Interpolated hidden Markov model, we learn an interpolated distribution for the emissions, similar to the method explained above. For each agent, the probability distribution of the emissions are learned by using its distribution and the distributions of its "neighbors".

The HMMs we consider here, is in some ways the simplest possible in that the emissions involve a single variable. More complex HMMs in which emissions include both the class and features are possible as well.

5 Evaluation

We conducted a set of experiments to assess that how well our approaches model evolving heterogeneous agent behavior in real world domains. We wish to (i) compare the predictive accuracies of the various models, (ii) investigate the effect of the training set size, and (iii) use HMMs to model concept drift over time We have have assembled a publicly available real world data set involving humans. Our set is derived from the Netflix prize[2] data. This data set contains over 100 million movie ratings (integer values 1-5) for 400,000+ users and 17,770 movies from 1999 to 2005. We use a subset of this data in our experiments. We extracted all the ratings from 2000 randomly selected users (the agents in this problem) for a total of 422,692 ratings. The rating is the class variable, and we divided ratings such that ratings (1,2,3) are classified as low or '0' and ratings (4,5) are classified as high or '1'. Making it a binary class problem We have 103 features for each rating: (i) The day of week of the rating, (ii) The month of the rating (iii) The movie's year of release. The remaining features are the ratings (if available) of 100 "heavy" users who have rated more than 1,000 movies. These 100 users are disjoint from the agents.

The first experiment was designed to see how well the interpolated method models the prior distribution $P(Y_a)$. In this experiment the 30% of the examples with greatest time index were held aside for evaluation. We trained models with successively more training data. We use all data from the time of the earliest instance through t where t is set so that 1%, 2%, ..., 70% of the available labeled examples are in the training set. Each trained model predicts the value of the held aside 30%. We measure the performance of each trained model with the test set log-likelihood. The parameter M was set to 20.

Figure 3 shows the results of this experiment. As expected, the heterogeneous model shows the most improvement as the size of the training set increases and the homogeneous model shows the least with the interpolated model in between. The interpolated model surprisingly continues to slightly improve throughout the whole range. Given the large amount of data it is unlikely that this trend is caused

[2] http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

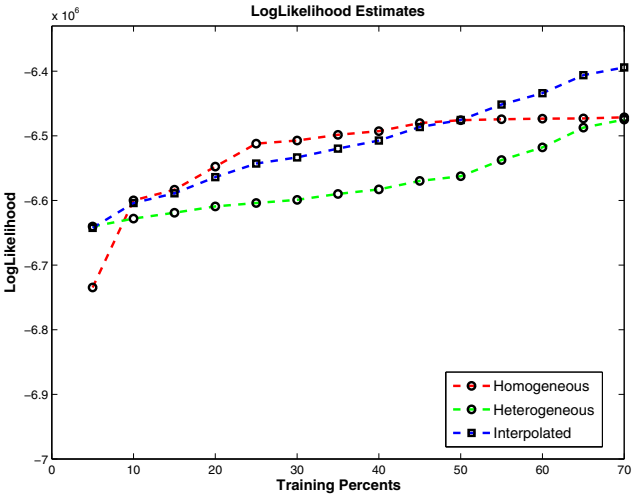

Fig. 3. Log-Likelihood Estimates for Homogeneous, Heterogeneous and Interpolated Model

by estimation error, but suggests that drift may be present. Comparing the interpolated curve to the others, we see that performance begins close to the homogeneous model and then improves as the individual agent models begin to diversify.

These trends suggest high inter-agent variability. We see that the interpolated models perform best in settings with diverse agents. From these results we conclude that the interpolated approach can lead to more accurate estimation of probability distributions, especially if agents are diverse.

To compare the predictive accuracy of the interpolated model, we compute ROC curves (Fig. 4) for each of our static models. In this experiment we use all features (as well as class labels) to train the models with the first 70% of the data and predict the labels on the final 30%. Although the interpolated model was the most accurate in terms of test-set log likelihood this doesn't translate into a clear win in the ROC curves. It is only slightly better than the homogeneous and heterogeneous models.

Our investigations into using HMMs to model concept drift focus on the predicting the class values using only the priors and not the features. Similar to the first experiment, 30% of the examples with greatest time (in total, not per agent) index were held aside for evaluation. A separate HMM is trained from the first 40% using only that agent's observed label sequence. We also learn an Interpolated HMM for each agent by applying the interpolated approach to the emission distributions. We compare the results of the HMM to the heterogeneous model and the Interpolated-HMM to the Interpolated Model. We measure the performance of each trained model with the test set log-likelihood. Figure 5 shows the results of the 2 types of HMMs learned. The results showed strong evidence of concept drift, as the difference in the test set log likelihoods (summed over

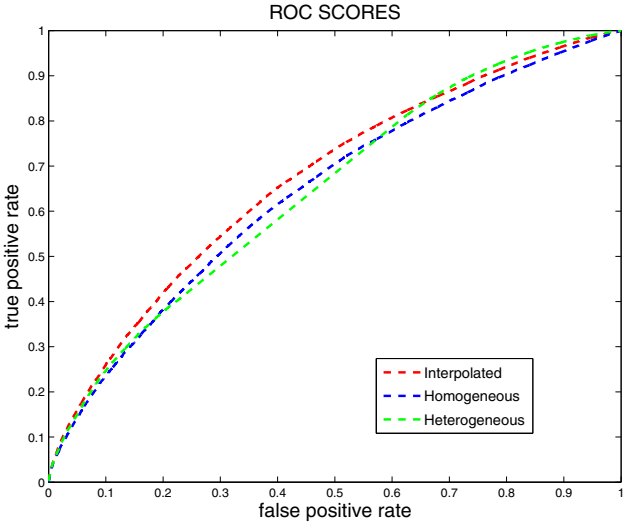

Fig. 4. ROC Curves for Homogeneous, Heterogeneous and Interpolated Models

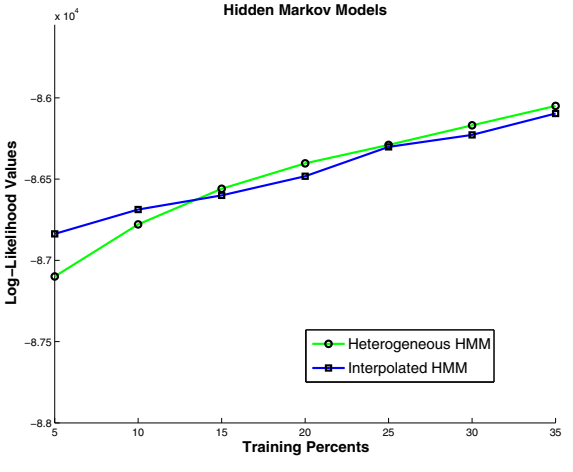

Fig. 5. Log-Likelihood of Heterogeneous and Interpolated Hidden Markov Models

all agents) of the interpolated model and the HMMs was in favor of the HMMs. We see that initially the heterogeneous HMM starts lower than the interpolated HMM, which can be attributed to lack of training data for each agent. As more training data is made available, both models almost tie with each other, but still increasing, which suggests the presence of concept drift. Observing the agent specific log likelihoods shows that most of the difference between that static models and the HMMs is accounted for by a relatively small number of agents.

It might be that these agents are strongly drifting while the others are weakly or not drifting. This suggests an alternate approach in which some agents are modeled with HMMs while others are modeled with static models.

6 Conclusion and Future Work

In this paper we investigated probabilistic models for multiple evolving agents. We addressed the common complications that may arise in such a setting, viz. (i) concept drift over time for a given agent, (ii) varying relationships among different agents. We compare and contrast 3 different static models, viz. (i) the homogeneous model in which we learn a single model for each agent, (ii) a heterogenous model where we learn a different model for each agent. (iii) an interpolated model where similar to the heterogeneous approach, we have a different model for each agent, but unlike it, observations from other agents can have an influence on an agent's model. Then we investigate hidden Markov models to address the issue of concept drift over time. Lastly we applied the interpolated approach to HMM to get an interpolated hidden Markov model for each agent. The contribution of this paper is the introduction of a novel method for learning agent models that involves combining the models of multiple agents in the same neighborhood.

Importantly, the concept of neighborhood is context specific, so the neighboring agents that combine to form one distribution may be distinct from those that combine in another. We demonstrated the validity and need of such models in real world scenarios. In settings with diverse agents, models learned with our interpolated approach proved to be empirically better than purely homogeneous or purely heterogeneous models. Concept drift over time was investigated using hidden Markov models and strong evidence of concept drift was found.

Our Interpolated Hidden Markov Model , that is used in conjunction with hidden variables, can be used to represent cases where there are time varying concepts affected by unknown factors. For example, such models may find use in risk assessment in financial data such as stocks. Thus, modeling the evolving relationship among various agents together with accounting for time varying concepts can prove to be very useful in several fields like finance, marketing or medicine. This work can be further extended by considering models where there are more than one type of hidden variable, each type affecting different observable features of the agent. Hence this can effectively model cases where certain features of an agent vary over time and others remain consistent. A typical setting would be a behavioral analysis of a multi-agent system which integrates models into an intelligent structure that improves the efficiency of an agent[2].

References

1. Case, J., Jain, S., Kaufmann, S., Sharma, A., Stephan, F.: Predictive learning models for concept drift. In: Richter, M.M., Smith, C.H., Wiehagen, R., Zeugmann, T. (eds.) ALT 1998. LNCS (LNAI), vol. 1501, pp. 276–290. Springer, Heidelberg (1998)

2. Coulondre, S., Simonin, O., Ferber, J.: Dynamo: a behavioural analysis model for multi-agent systems. In: Proceedings 1999 International Conference on Information Intelligence and Systems, pp. 614–621 (1999)
3. Delcher, A., Kasif, S., Fleischmann, R., Peterson, J., White, O., Salzberg, S.: Alignment of whole genomes. Nucleic Acids Research 27(11), 2369–2376 (1999)
4. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B 39, 1–38 (1977)
5. Helmbold, D.P., Long, P.M.: Tracking drifting concepts by minimizing disagreements. Machine Learning, 27–45 (1994)
6. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: SIGIR 1999: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 230–237. ACM, New York (1999)
7. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 97–106. ACM, New York (2001)
8. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD 2008: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 426–434. ACM, New York (2008)
9. Lane, T., Brodley, C.E.: Approaches to online learning and concept drift for user identification in computer security. In: KDD, pp. 259–263. AAAI Press, Menlo Park
10. Leung, C.W.-k., Chan, S.C.-f., Chung, F.-l.: A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowl. Inf. Syst. 10(3), 357–381 (2006)
11. Núñez, M., Fidalgo, R., Morales, R.: Learning in environments with unknown dynamics: Towards more robust concept learners. J. Mach. Learn. Res. 8, 2595–2628 (2007)
12. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
13. Virtanen, T., Heittola, T.: Interpolating hidden markov model and its application to automatic instrument recognition. In: ICASSP 2009: Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Washington, DC, USA, pp. 49–52. IEEE Computer Society, Los Alamitos (2009)
14. Wang, Y., Zhou, L., Feng, J., Wang, J., Liu, Z.-Q.: Mining complex time-series data by learning markovian models. In: ICDM 2006: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, pp. 1136–1140. IEEE Computer Society, Los Alamitos (2006)
15. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. 23(1), 69–101 (1996)
16. Zhang, P., Zhu, X., Shi, Y.: Categorizing and mining concept drifting data streams. In: KDD 2008: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 812–820. ACM, New York (2008)