Recognizing Plans by Learning Embeddings from Observed Action Distributions

Extended Abstract

Yantian Zha, Yikang Li, Sriram Gopalakrishnan, Baoxin Li, Subbarao Kambhampati
Arizona State University
Tempe, Arizona
{Yantian.Zha,yikangli,sgopal28,baoxin.li,rao}@asu.edu

ABSTRACT

Automated video surveillance requires the recognition of agent plans from videos. One promising direction for plan recognition involves learning shallow action affinity models from plan traces. Extracting such traces from raw video involves uncertainty about the actions. One solution is to represent traces as sequences of action distributions. To use such a representation in approximate plan recognition, we need embeddings of these action distributions. To address this problem, we propose a distribution to vector (Distr2Vec) model, which learns embeddings of action distributions using KL-divergence as the loss function.

KEYWORDS

Word2Vec, Distr2Vec, Plan Recognition, Distribution Sequences

ACM Reference Format:
Yantian Zha, Yikang Li, Sriram Gopalakrishnan, Baoxin Li, Subbarao Kambhampati. 2018. Recognizing Plans by Learning Embeddings from Observed Action Distributions. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

Plan recognition [3, 5] is essential for surveillance and multi-agent collaboration, in order to predict the actions of other agents. Approximate plan recognition using a shallow model can be a fast and efficient way to do this as shown in the work `DUP` [7]. In `DUP`, the shallow model was learned using Word2Vec[1], and the learned embeddings capture the affinity between actions, which we call affinity models. The input traces to Word2Vec in `DUP` are plan traces of single actions at each step (actions are the words for Word2Vec). Where there are traces extracted from sensory data, often there is uncertainty about the recognized action. Thus we must allow a distribution over actions at each step. This is what our work `Distr2Vec` allows. By learning embeddings for action distributions, we create a more powerful data-interface between the perception module and plan recognition module. We demonstrate it's value by comparing the performance of Word2Vec and `Distr2Vec` in plan recognition by using them (separately) in a version of `DUP` for uncertain inputs, called `UDUP` that allows uncertain plan traces as input. In our experiments we vary the perception error rate (PER) and entropy in observation distributions, and compare the training time, and accuracy between the two models. Our full paper can be accessed at https://arxiv.org/abs/1712.01949.

2 PROBLEM FORMULATION AND MODEL

The input of our `Distr2Vec` in `UDUP` is in the form of the following matrix. Each step is a distribution over actions $(a^1, ..., a^k)$ with their probabilities (confidence) $(c^1, ..., c^k)$. The values of T and K represent the number of time-steps and actions per step respectively.

$$\begin{array}{cc} & \xrightarrow{\text{Time}} \\ \text{Actions}\ \big\downarrow & \begin{pmatrix} a_1^1, c_1^1 & a_2^1, c_2^1 & a_3^1, c_3^1 & \cdots & a_T^1, c_T^1 \\ a_1^2, c_1^2 & a_2^2, c_2^2 & a_3^2, c_3^2 & \cdots & a_T^2, c_T^2 \\ a_1^3, c_1^3 & a_2^3, c_2^3 & a_3^3, c_3^3 & \cdots & a_T^3, c_T^3 \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ a_1^K, c_1^K & a_2^K, c_2^K & a_3^K, c_3^K & \cdots & a_T^K, c_T^K \end{pmatrix} \end{array}$$

Taking that matrix (distribution sequence) as the input, our `Distr2Vec` model framework is shown in Figure 1.

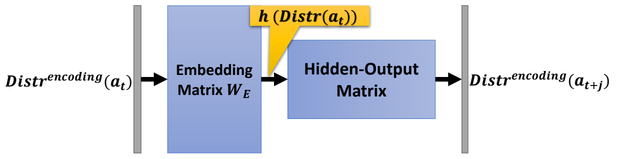

Figure 1: The architecture of our `Distr2Vec` model for learning distribution embeddings and action affinity models.

Like in Word2Vec, we try to maximize the similarity (of the embeddings) between an action distribution and it's neighbors. We minimize Kullback-Leibler (KL) divergence to maximize the similarity between our target and predicted output distribution as follows;

$$D_{KL}(Distr^{encoding}(a_{t+j})||\hat{Distr}^{encoding}(a_{t+j})) \quad (1)$$

where D_{KL} represents the KL divergence. KL-divergence is calculated as per Equ. 2.

$$KL(p||q) \triangleq \sum_{k=1}^{K} p_k log\frac{p_k}{q_k} = \sum_{k=1}^{K} p_k log p_k - \sum_{k=1}^{K} p_k log q_k \quad (2)$$

where q is the output probability distribution of our `Distr2Vec`. q is also an approximation of p, the target distribution $Distr(a_{t+j})$. We try to minimize the *inclusive* KL divergence [2] with the model's target distribution. An advantage of using inclusive KL divergence, is that we avoid computing the derivative of the entropy of p when taking partial derivative of $KL(p||q)$ with respect to model parameters. This is because the values for p (which is $Distr(a_{t+j})$), is a constant with respect to the model parameters. Using this information, we can obtain the Equ. 3.

$$D_{KL}(Distr^{encoding}(a_{t+j})||\hat{Distr}^{encoding}(a_{t+j}))$$
$$= Z(Distr(a_{t+j})) - \sum_{k=1}^{K} c_{t+j}^{k} logp(a_{t+j}^{k}|h(Distr(a_t))) \quad (3)$$

where $Z(Distr(a_{t+j})) = \sum_{k=1}^{K} c_{t+j}^{k} log(c_{t+j}^{k})$ is a constant, and $h(Distr(a_t))$ is the embedding computed by multiplying the embedding matrix W_E and the distribution input vector $Distr^{encoding}(a_t)$ $= \langle 0...0, c_t^1, 0, ..., 0, c_t^2, 0, ...0, c_t^K, 0...\rangle$ encoded from $Distr(a_t)$ (Equ. 4).

$$h(Distr(a_t)) = W_E \times Distr^{encoding}(a_t) \quad (4)$$

Now we elaborate how to combine Equ. 3 with using the hierarchical softmax[4]. If we extend the hierarchical softmax [4] in a normal Word2Vec, to handle a distribution input $Distr(a_t)$, we obtain the probability of an action in the target observed action distribution $Distr(a_{t+j})$:

$$p(a_{t+j}^{k}|h(Distr(a_t))) = \prod_{i=1}^{L(a_{t+j}^{k})-1} \Big\{\sigma(\mathbb{I}(n(a_{t+j}^{k}, i+1) \quad (5)$$
$$= child(n(a_{t+j}^{k}, i))) \cdot v_{n(a_{t+j}^{k}, i)} \cdot h(Distr(a_t)))\Big\}$$

And if we combine Equ. 5 and Equ. 3, we obtain the error function in Equ. 6. This is the error function as we are trying to minimize the KL divergence of Equ. 3.

$$E = Z(Distr(a_{t+j})) - \sum_{k=1}^{K} c_{t+j}^{k} \sum_{i=1}^{L(a_{t+j}^{k})-1}$$
$$\Big\{log\sigma(\mathbb{I}(.)v_{n(a_{t+j}^{k}, i)} \cdot h(Distr(a_t)))\Big\} \quad (6)$$

For details on the weights update derivation of Equ. 6, please refer to the Sec. 3.2, in the aforementioned full paper.

Now with a trained Distr2Vec as the action affinity model, we use it as a subroutine in our UDUP, which is similar to the usage of Word2Vec in DUP in [7]. Please refer to the Sec. 3.3 in our full paper for the details.

3 EVALUATION

We evaluate the performance of Distr2Vec in UDUP by comparing the performance with UDUP that uses other affinity models trained with Word2Vec. In order to train a Word2Vec model from traces that have distributions, we sample from the distributions at each step, and generate sampled traces of single actions (not distributions). The way we sample the distributions gives us two baseline models: a Naive Model (NM), and a Resampling Based Model (RBM).

3.0.1 Two Baseline Models. In NM, we feed what the perception module considers the ground truth to Word2Vec for training affinity models. In RBM, we first calculate the likelihoods of all possible paths from each trace of action distributions. The path weights (PW) can be calculated by multiplying the confidence values of all actions along a path. The top N ground traces (ranked according to PWs) are selected, and then resampled using the roulette wheel resampling approach. The set of traces selected after the resampling step is used to train a Word2Vec model.

3.0.2 Dataset Collection and Testing Methodology. For the experiments, we created a synthetic dataset of action distribution sequences from the 50 Salads Dataset [6]. We chose a synthetic dataset in order to systematically assess the validity and effectiveness of our Distr2Vec approach, as we can modify it to test with different configurations of distributions. Tuning the parameters of the distribution lets us maximally evaluate our model. For more details regarding the dataset collection, please refer to Sec. 4.1 in the linked full paper. As for the testing methodology, we follow the process as in the work of [7]. For details please refer to the Sec. 4.2 in our full paper.

3.0.3 Experiment Analysis. We analyze the effect of the parameters of $w_{entropy}$, PER, and the length of observed action distribution sequences (10,20 and 30), on the accuracy of UDUP with action affinity models. Action affinity model are trained on distribution sequences, by our Distr2Vec, as well as by normal Word2Vec models (i.e., NM and RBM as explained before). We also analyze the training time of the three models. For our experiment results and full analysis, please refer to the Sec. 4.3 and 4.4 in our full paper, including how entropy and PER is defined and adjusted.

To summarize, we can observe that when there is a higher PER, UDUP with Distr2Vec outperforms other models. When the PER is lower, all models perform comparably well. Additionally, when the entropy of the data is higher, the Distr2Vec still produces appreciable-quality action affinity models which keeps up the accuracy of the UDUP. Another benefit of using Distr2Vec with UDUP is that the training time is comparable to the NM. However, the NM loses all information about the uncertainty in the data, and thus is prone to more errors. Distr2Vec training time is also lower than the RBM when we sample more traces. The training time of RBM increases linearly with the number of samples taken per plan trace. Comparing the training time of Distr2Vec model with RBM is more appropriate because both models factor in the uncertainty in the training data, but NM discards it.

4 CONCLUSION

We introduced our Distr2Vec model, that learns embeddings for distributions. We applied Distr2Vec to do plan recognition, by using it to extend DUP [7] to UDUP. Unlike DUP, UDUP can be trained on traces of observed action distributions, and thus can handle uncertainty in the input.

ACKNOWLEDGMENTS

This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014161-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, and the NASA grant NNX17AD06G.

REFERENCES

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[2] Tom Minka et al. 2005. *Divergence measures and message passing*. Technical Report. Technical report, Microsoft Research.

[3] Miquel Ramírez and Hector Geffner. 2009. Plan Recognition as Planning. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*. 1778–1783.

[4] Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* (2014).

[5] Shirin Sohrabi, Anton V Riabov, and Octavian Udrea. 2016. Plan Recognition as Planning Revisited.. In *IJCAI*. 3258–3264.

[6] S. Stein and S. J. McKenna. 2013. Combining Embedded Accelerometers with Computer Vision for Recognizing Food Preparation Activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), Zurich, Switzerland*. ACM.

[7] Xin Tian, Hankz Hankui Zhuo, and Subbarao Kambhampati. 2016. Discovering Underlying Plans Based on Distributed Representations of Actions. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*. 1135–1143.