CHAPTER X

SEMANTIC ACTION PARAMETER INFERENCE THROUGH MACHINE LEARNING METHODS

J. G. VICTORES, S. MORANTE, A. JARDÓN and C. BALAGUER

Robotics Lab, Universidad Carlos III de Madrid; jcgvicto@ing.uc3m.es

This paper presents the initial steps towards a robot imagination system, which aims at providing robots with the cognitive capability of imagining how a set of actions can affect a robot's environment, even if the robot has never seen the specific set of actions applied to its environment before. This robot imagination system is part of a human-inspired and goal-oriented infrastructure, which first learns the semantics of actions by human demonstration, and is then capable of performing the inverse semantic reconstruction process through mental imagery. A key factor in this system is distinguishing how different actions affect different features of objects in the environment. Simple probabilistic and other machine learning methods, tested to perform this first step of the inference process, are presented and compared in this paper. The inference of results of composed actions is generated as the sum of the contributions of each of the query word components. As an initial prototype, the actual learning process has been performed using synthetically created minimalistic environments as datasets, and a limited amount of training words for the learning process.

1 Introduction

Every day, humans perform thousands of actions. For robotic systems to be able help us with our daily life tasks, they must be endowed with recognition capabilities, storage, and reproduction of a great subset of actions to be performed in corresponding situations. The amount of available actions, their imprecision in their execution, and a strong dependence on the preferences of different end-users make the hard-coding of parameters an invalid solution of the general case. At most, one would expect to categorize actions to then evaluate their characteristics. This process of categorization has already been performed: languages classify different actions, binding different categories with different words (specifically, *verbs*). Obtaining knowledge on the mappings between words and instances from the world is often referred to as 'symbol grounding' (Harnad, 1990), or bridging the semantic gap.

In this paper we study the application of different machine learning algorithms to discover the 'grounding' of actions: the relation between words and their corresponding actions, and how these actions can modify the world. The 'grounding' part of this work is closely related to works in the field of action recognition, also known as direct action recognition (Poppe, 2010), and has also been targeted in the field of learning by imitation, also known as robot programming by demonstration (Calinon, 2010). These works are almost completely aimed at learning kinematics of the actions that humans perform, such as the trajectory of a human arm when reaching for an object, or the kinematics and dynamics of performing power grips. However, recent literature from the areas of neuro-science and psychology tends to indicate that the human brain encodes actions as end-goals related to the affordances of objects. For instance, when children imitate others grasping a person's ear, they tend to imitate the action goal (which ear to grasp) rather than the kinematic aspects of the action (which hand is used to perform the grasping) (Bekkering, 2000).

An attempt to learn goal-oriented tasks is shown in (Calinon and Guenter, 2005) where, despite they aim to learn the trajectory to perform an action, they also code some goals to be achieved, even in a distinct way than learned. A deeper understanding of object uses can be found in (Fitzpatrick, 2003), where the effects of pushing/pulling actions on objects to acquire affordances (what actions objects can 'afford') are learnt. A great variety of techniques has been used to try to learn object affordances, such as Self-Organizing Feature Maps (Cos-Aguilera, 2004), SVM (Dogar, 2007), and Bayesian Networks (Montesano, 2008).

Working in the line of learning about actions in a goal-oriented fashion, we focus on action recognition by detecting changes in the object involved. This means, for instance, recognizing a rotation action by noticing the change in the orientation of the object. Additionally, to analyze the effectiveness of the learning and providing the cognitive capability of robot imagination, we ask the system how the composition of these actions can affect the robot's environment, even if the robot has never seen the actions applied conjunctly to its environment before.

The rest of the paper is organized as follows: Section 2 outlines the proposed custom space, Section 3 explains different algorithms to generate action parameters, Section 4 compares the algorithm presented, and Section 5 discusses about limitations and deficiencies providing several conclusions.

2 Semantic Grounding Database

The semantic grounding database is represented and stored as an incremental set of tagged points in an *n*-dimensional feature space (see Fig. 1), denoted *F*. Every time an action is performed, we measure the variation of the values of the features we extract from the object we are interested in, and create a new point in *F* using these variations as coordinates.

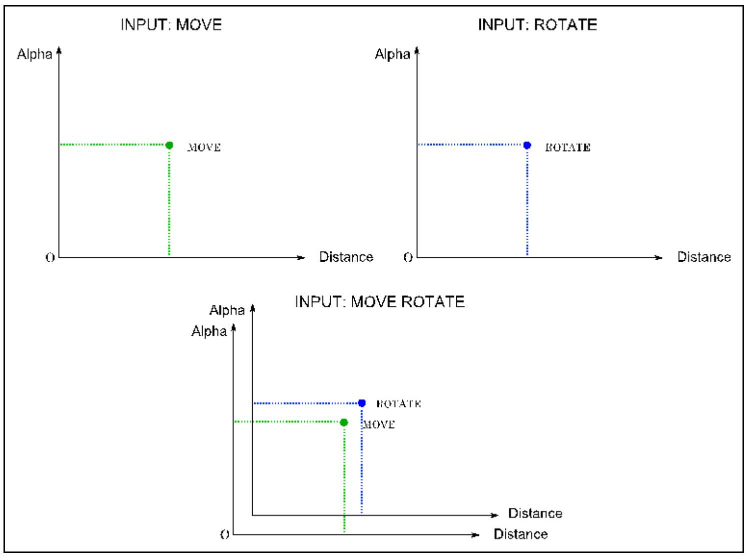

Fig. 1. Semantic Grounding Space example where *n*=2

This new point is tagged with the name of the action, thus enhancing the point with a semantic label that will be used for posterior inferences. The semantic label is a single word. If more than one word is input, overlapping points with the different words as tags (i.e. ‘move rotate’ becomes ‘move’ and ‘rotate’).

For the experiments presented in this paper, we have fixed the dimensions to three (X, Y and ALPHA), which represent the translation and the rotation of an object in a 2-dimensional space. As semantic input we consider only two words: one for representing a pure translation movement, ‘move’, and another one for representing a pure rotation movement, ‘rotate’. As stated, the grounding process is repeated for each sample, until the database has been populated with a sufficient amount of samples. We will start from this populated situation to test our algorithms.

3 Models of Study

The aim of presented algorithms is to obtain coherent and valuable parameters for the corresponding semantic input we introduce. The methods presented cover the most intuitive approximation, starting with a simple Arithmetic Mean of the samples with the required tag. We also test a Neural Network system, specifically trained to work in an inverse way to its conventional use. We construct a Gaussian Mixture Model and extract the representative parameters. Finally, a Support Vector Machine modified to work as a regressor is also tested.

3.1 Arithmetic Mean Model

The Arithmetic Mean (AM) model is a relatively naive approach. Upon receiving a pair <word, [x, y, alpha]>, the Look-Up Table of each element of the output layer is updated with its corresponding mean. A dependency graph is depicted in Fig. 2.

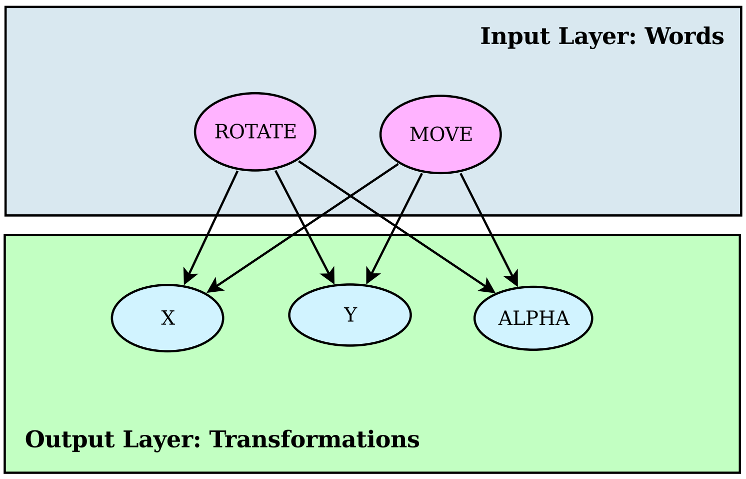

Fig 2. Arithmetic Mean Model fully connected dependency graph

An advantage of the AM is that its incremental form for performing online learning is well-known and simple to implement, as in Equation (1).

$$A_{n+1} = A_n + \frac{v_{n+1} - A_n}{n+1} \tag{1}$$

3.2 Neural Network Model

A classical three-layered fully connected Feed-Forward Neural Network (NN) model has been used for this application, as seen in Fig. 2. Its number of input perceptrons has been set to 2 (corresponding to the 2 input words), its number of output perceptrons has been fixed at 3 (corresponding to the 3 used features), and 50 has been the number of perceptrons selected for the hidden layer implementing a sigmoid function. The NN is trained by back-propagation, a supervised learning method, set at 10 epochs. The number of perceptrons in the hidden layer has been empirically determined, and offers a tradeback with the number of epochs necessary to achieve a certain behavior.

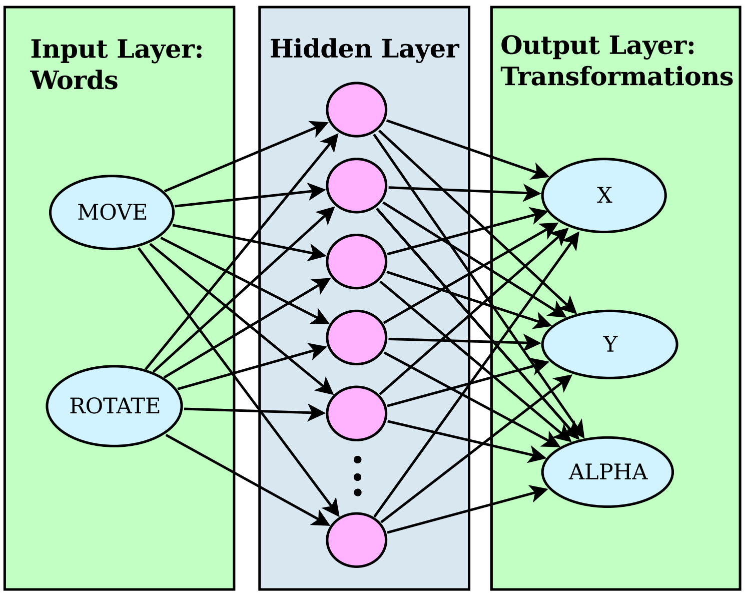

Fig. 3. Three-layered fully connected Feed-Forward Neural Network model

It is important to notice that these NN are used in an inverted way when compared to their habitual use. Instead of classifying a combination of inputs into a delimited number of classes, we use the class as the network input, and the values belonging to the class as the network output. The resulting network we achieve with this method is a kind of 'pseudo-inverse NN', which allows us to interpolate between intermediate values of the cloud of samples.

3.3 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) represents the probability distribution (density estimation) of the points. We set the number of mixtures to one (formally K=1 or M=1), and feed the GMM with all the samples with the same word. The values used are the Gaussian means.

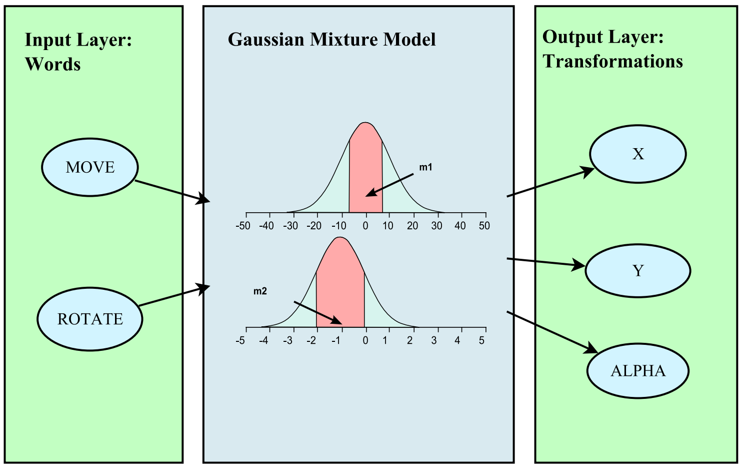

Fig. 4. Gaussian Mixture Model

Because the use of a single component, the GMM actually corresponds to a single multivariate Gaussian model. The only component can be summed out of the general weighted GMM expression of Equation (2).

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{2}$$

This results in a single (unit weight) component expressing the multivariate Gaussian probabilistic density function of Equation (3).

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} exp \left\{ -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) \right\} \tag{3}$$

The actual fitting of the parameters to the data is performed using the Expectation-Maximization (EM) algorithm, an iterative method for performing Maximum Likelihood Estimation (Reynolds, 2008).

3.4 Support Vector Regression (SVR)

The Support Vector Regression method (Fig. 5) is a modified version of a Support Vector Machine (SVM). SVM are normally used in classification problems, but SVR are adapted to perform regressions.

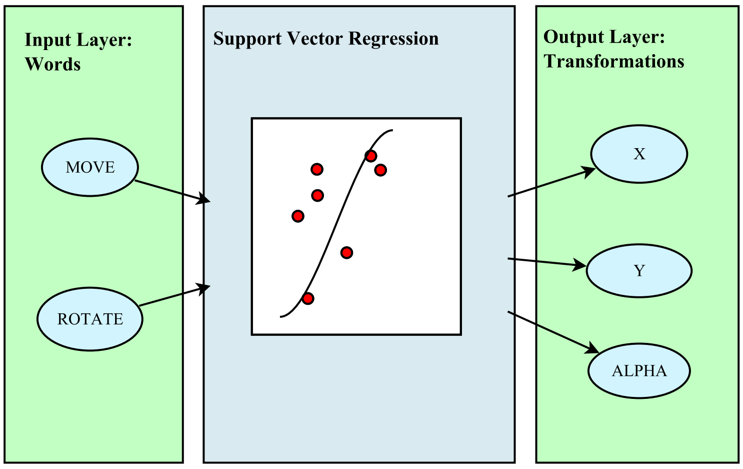

Fig. 5. Support Vector Regression

We use a non-linear regression version, which uses a kernel function that transforms the data into a higher dimensional feature space in order to perform a linear separation. The kernel function is a Radial Basis Function, seen in Equation (4).

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{4}$$

This process is called ‘Kernel trick’ and it is used to map samples into other spaces, hoping that samples will gain linear structure in the new space. In this new space, SVM uses ‘epsilon intensive loss’ to estimate the quality of estimation. This function considers the residuals in two ways: if residual is less than epsilon, then there is no loss, if it is bigger, then, an amount of loss is added. The main idea of SVR is performing a linear regression in the higher dimension space, with the epsilon intensive function.

4 Experiments for Comparison of Models

To compare the presented models, we train each with the same experimental dataset. The dataset is composed by 5 ‘move’ sequences, and 5 ‘rotate’ sequences. The sequences are composed by 7 frames (100 by 100 pixel black and white images), such as those seen in Figure 4. The contents of a ‘move’ sequence is a rectangle that advances 60 pixels on X and Y while maintaining its rotation. Each ‘rotate’ sequence depicts a 60º angle rotation while maintaining its position.

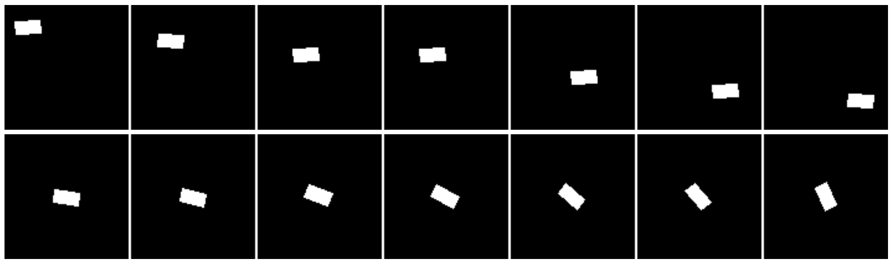
Fig. 5. Image sequences: a) move, b) rotate

An additional 1% standard deviation noise has incorporated to each feature (X, Y and ALPHA) to emulate camera perturbation effects and inefficiencies during segmentation. Note that this noise may significantly affect the behavior of the algorithms, as the training dataset is relatively small. The programming language used was Python, mostly using the tools provided by Scikit-Learn (Pedregosa, 2011).

Table 1 and Table 2 depict the outputs of the different tested algorithms assigned to the ‘rotate’ and ‘move’ word, respectively. For compactness, the value at the right of the ± operator represents the standard deviation.

Table 1. Table for ROTATE values

Employed techniques	X	Y	Alpha
Arithmetic Mean	0	-0.2	65.62
Neural Network (*)	0.05 ± 0.26	-0.12 ± 0.1	64.83 ± 0.22
Gaussian Mixture Model (**)	0 ± 1.41	-0.2 ± 0.74	65.62 ± 2.81
Support Vector Regressor	0	-0.1	64.55

Table 2. Table for MOVE values

Employed techniques	X	Y	Alpha
Arithmetic Mean	59.8	59.6	-0.26
Neural Network (*)	59.75 ± 0.2	59.56	-0.74 ± 0.43
Gaussian Mixture Model (**)	59.8 ± 0.74	59.6 ± 1.35	-0.26 ± 2.8
Support Vector Regressor	59.9	59.1	0.71

It is important to notice the difference in meaning of the standard deviation of the NN algorithm and the GMM algorithm. The NN algorithm's standard deviation is marked with a (*) because the output of the NN is stochastically variable. This is due to the fact that the network's weights are set randomly at initialization. The table values of the NN are the averaged values of running the algorithm 5 times. The GMM algorithm's standard deviation (**), however, describes the dispersion of the input data, a fixed parameter with an accuracy given by the estimation based on the maximization of the likelihood of how the distribution expresses the original training data. It is computed as the square root of the diagonal of the estimated covariance matrix.

The final Root-Mean-Square (RMS) errors on composition are illustrated in Table 3. They have been computed as the direct sum of the values found on Table 1 and Table 2, setting the target values to X=60, Y=60 and ALPHA=60º (which is the intuitive composition of 'rotate' and 'move' a human could deduce from the input). This table represents a final metric on the accuracy of the algorithm, while obviating other metrics or advantages which will be considered in the following final section.

Table 3. Table for MOVE and ROTATE composition errors (RMS)

Employed techniques	RMS
Arithmetic Mean	5.403
Neural Network	4.97 ± 0.45
Gaussian Mixture Model	5.403
Support Vector Regressor	5.367

5 Conclusions

The bases for future research on robot imagination systems of actions and action compositions using algorithms from the field of machine learning have been set. The following points review some of the benefits and drawbacks of the studied algorithms on this specific minimalistic dataset:

- The Arithmetic Mean offers a fast and accurate solution. Additionally, the possibility of easily implementing its incrementally learning version will surely be attractive to many readers. However, it lacks a certain degree of flexibility and deep philosophical interpretations that may be found abundance within other algorithms.

- Our Neural Network implementation has proved to work well, even having being trained inversely. It is the most and perhaps only bio-inspired algorithm of the ones tested. This is also manifest in the fact that its results are not always the same, much like in a human's actions. Moreover, it offers a quite unique possibility of simultaneously activating several inputs. These results were, however, omitted in the experiment section due to bogus results: output more similar to the mean than to the sum of actions.

- We consider the Gaussian Mixture Model the most attractive option. The results are precise as with the Arithmetic Mean, but may be distributed with a standard deviation similar to that of the input values if desired. This can be important if, for example, the algorithm determines a great degree of dispersion is a representative characteristic of a certain action. Additionally, several algorithms which have not been used here can be used to perfect the model. For example, Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC) may be used to determine an optimal number of Gaussian components other than K=1. Furthermore, recent studies indicate Bayes-Optimal estimators may achieve higher performance than methods based on maximization of likelihood.

- The Support Vector Regressor algorithm has been proved to be a valid method in terms of computation cycles and accuracy. However, the authors consider that further study is required in order to unveil its full potential.

Moreover, the authors would like to transmit a final question to the robotic scientific community: "How are robots going to transform the world if they can't even imagine how?"

Acknowledgements

The research leading to these results has received funding from the ARCADIA project DPI2010-21047-C02-01 funded by CICYT project grant on behalf of Spanish Ministry of Economy and Competitiveness.

References

Bekkering, H., Wohlschlager, A., and Gattis, M. 2000. Imitation of gestures in children is goal-directed. The Quarterly Journal of Experimental Psychology: Section A, 53(1), 153–164.

Calinon, S., D'halluin, F., Sauser, E., Caldwell, D. and Billard, A. 2010. Learning and reproduction of gestures by imitation. IEEE Robotics and Automation Magazine, vol. 17, no. June, pp. 44–54.

Calinon, S., Guenter, F., and Billard, A. 2005. "Goal-directed imitation in a humanoid robot. In Robotics and Automation". ICRA 2005. Proceedings of the 2005 IEEE International Conference on (pp. 299–304). IEEE.

Cos-Aguilera, I., Hayes, G., and Cañamero, L. (2004). Using a SOFM to learn object affordances. In Procs 5th Workshop of Physical Agents (WAF'04). University of Edinburgh.

Dogar, M. R., Cakmak, M., Ugur, E., and Sahin, E. (2007, October). From primitive behaviors to goal-directed behavior using affordances. In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on (pp. 729–734). IEEE.

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. 2003. Learning about objects through action-initial steps towards artificial cognition. In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on (Vol. 3, pp. 3140–3145). IEEE.

Harnad. S. 1990. The symbol grounding problem. Physica D: Nonlinear Phenomena, 42(1):335–346.

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning Object Affordances: From Sensory-Motor Coordination to Imitation. Robotics, IEEE Transactions on, 24(1), 15–26.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830.

Poppe, R. 2010. A survey on vision-based human action recognition. Image and vision computing, 28(6), 976–990.

Reynolds, D. (2008). Gaussian mixture models. Encyclopedia of Biometric Recognition, 2(17.36), 14-68.