Inferring Capabilities by Experimentation

Ashwin Khadke[1] and Manuela Veloso[2]

[1] Robotics Institute, Carnegie Mellon University
[2] Department of Machine Learning, Carnegie Mellon University

Abstract. We present an approach to enable an autonomous agent (learner) in building a model of a new unknown robot's (subject) performance at a task through experimentation. The subject's appearance can provide cues to its physical as well as cognitive capabilities. Building on these cues, our active experimentation approach learns a model that captures the effect of relevant extrinsic factors on the subject's ability to perform a task. As personal robots become increasingly multi-functional and adaptive, such autonomous agents would find use as tools for humans in determining "What can this robot do?". We applied our algorithm in modelling a NAO and a Pepper robot at two different tasks. We first demonstrate the advantages of our active experimentation approach, then we show the utility of such models in identifying scenarios a robot is well suited for, in performing a task.

1 Introduction

Suppose we get a multipurpose domestic robot for our home but are not familiar with all of its functionalities. How do we identify what tasks it can perform? Appearance and specifications of a robot can convey information about its physical as well as cognitive capabilities. Seeing a legged robot equipped with a camera and a microphone could make one wonder if it can climb stairs, recognize faces, detect hand gestures or interpret voice commands. How do we identify which of these appearance-deduced tasks it can actually perform? Moreover, although physical appearances can provide rich cues about a robot's capabilities they are not sufficient to identify the scenarios in which it can function well. The robots Roomba and Braava appear similar and are both used for cleaning floors. But the Roomba can clean carpets and not wet floors, which is exactly the opposite of what Braava can do. For a human working collaboratively with a robot, knowing the robot's strengths and shortcomings is especially useful. A robot's spec sheet provides information about different sensors and actuators. But inevitably how well a robot performs a task depends on the way it is programmed and for a naive user, this is difficult to determine simply based on appearance and specifications. Experimenting with a robot can help identify the scenarios it is well suited for. But experimentation is tedious and intelligent robots are capable of learning new skills and adapting to new scenarios over time. This motivates the need for an autonomous system that can intelligently experiment with robots, identify their skills and quantify their applicability in different scenarios.

In this paper we tackle the problem of an autonomous agent (learner) building a model of a robot (subject) at performing a particular task through experimentation. The outcomes of these experiments can be non-deterministic. We call such models as Capability Models (Section 3). We assume the learner can infer tasks a subject can potentially perform from its appearance and, present a method to build a model of the subject at one of these tasks. Apart from the subject's inherent capabilities, certain extrinsic factors may affect its performance at the task. Assuming some of these factors are controllable and the learner can choose values for such factors in the experiments it conducts, we provide an approach to pick values for controllable factors that generate the most informative outcomes (Section 4.1). However, knowing the set of extrinsic factors relevant for a particular robot and a task a priori is not always feasible. We present a model refinement method to identify relevant factors from a set of candidates (Section 4.2). Further we show that Capability Models can be used in quantifying a robot's ability to perform a task in different scenarios thereby identifying situations a robot is well suited for (Section 5).

We applied our algorithm in modelling a NAO robot at the task of kicking a ball. We show that active experimentation leads to faster learning of the subject's model than passively observing it perform and that our model refinement approach correctly identifies the set of relevant factors missing from the model (Section 6.1). Furthermore, we learned a Capability Model for a Pepper robot programmed to pick up and clear objects off a table. From the learned model, we identified the types of objects the robot can pick up. We demonstrate how this knowledge is useful in improving performance at a collaborative clear-the-table task (Section 6.2).

2 Related Work

Earlier works on learning from experimentation either addressed domains that are inherently deterministic [1,2] or assumed that the experiments they conduct are deterministic [3]. Owing to noisy actuation and sensing, outcomes of experiments with robotic systems are non-deterministic and, hence the problem of deducing a robot's capabilities through experimentation is challenging.

Affordance [4] is a relation between a certain effect, a class of objects and certain robot action. Learning affordances [5,6] is similar to learning the physical capabilities of a robot. However, these approaches learn a mapping from motor commands to effects on different objects characterized by raw sensory input. It is difficult for a human to use these models in identifying scenarios a robot is well suited for. Another approach [7] uses visual features (Size, Shape, Color etc.) to characterize objects and learns models that capture the effect (Object speed etc.) of a robot's actions (Tap, Push, etc.) on different objects. However, all of these works [5,6,7] adopt passive approaches to learn and do not use their existing models to reason about what to explore next. Moreover, they provide no concrete method to quantify a robot's capability.

Modeling a robot's capability can be thought of as learning a forward model [8]. Such models predict the change in a robot's state brought about by its ac-

tions. Active approaches to learn forward models exist [9,10,11,12,13]. However, these models only predict the immediate effects of an action i.e. predict the next state given the current state and action and provide no approach to identify from the forward model, the likelihood of successful task execution in different scenarios. An idea to mitigate this problem is to learn a forward model over higher level states [14,15]. But the emphasis in these works is on learning a policy for a task [11,12] or a combination of tasks [10,13] or learning higher level actions (sub-policies) relevant for a task [14].

3   Capability Models

Suppose the subject is an anthropomorphic robot (Fig. 3a) and the learner chooses to build a model of the subject at the task of kicking a ball. Extrinsic factors that the learner may consider include, the size of the ball and turf on which the subject is playing. Among these factors, ball size is controllable. An experiment for this task would constitute the learner commanding the subject to kick a ball in certain direction from a particular position and observing the outcome. A robot's perception and actuation is noisy and therefore the outcome is not deterministic. To capture this non-determinism we use a Bayesian Network.
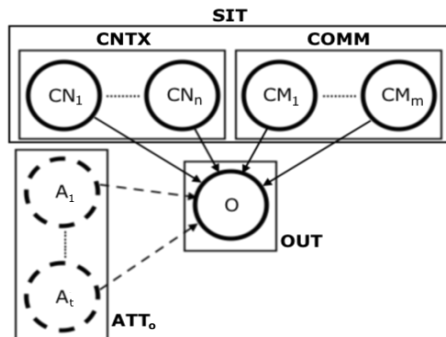

Fig. 1: Capability Model

We introduce Capability Model (Fig. 1), a Bayesian Network which consists of three types of nodes namely:

- SIT = CNTX ∪ COMM, is the set of variables that describe the situation in which the subject is performing the task.
  - CNTX is the set of extrinsic factors (context) for the task.
  - COMM is the set of commands given to the subject.
- OUT is the set of variables denoting the outcomes of the task.
- ATT_o is the set of attributes of variable $o \in$ OUT. These variables are not explicitly accounted in the model. Section 4.2 discusses their need and utility.

We capture the subject's ability to perform a task in the conditional probability tables associated with this Bayesian Network.

Fig. 2 presents a Capability Model for the *BallKick* task discussed before using the notation we just introduced. *Position* represents the location of the ball with respect to the subject before it kicks. *KDc* and *KDo* denote the commanded and observed kick direction respectively. *KDo* is *None* if the subject attempts but fails to kick the ball or does not attempt to kick. The set OUT need not always be a singleton. How far a subject kicks a ball could be another outcome for the *BallKick* task.

SIT = {*KDc, Position, Turf, BallSize*},
CNTX = {*Turf, BallSize*},
COMM = {*Position, KDc*},
OUT = {*KDo*}, ATT_{KDo} = {*BallColor*}

(b)

Position ∈ {*LeftSide, Middle, RightSide*},
Turf ∈ {*Grass, Synthetic, Sand*},
BallSize ∈ {*Small, Large*},
BallColor ∈ {*Yellow, Orange*},
KDc ∈ {*Left, Mid, Right*},
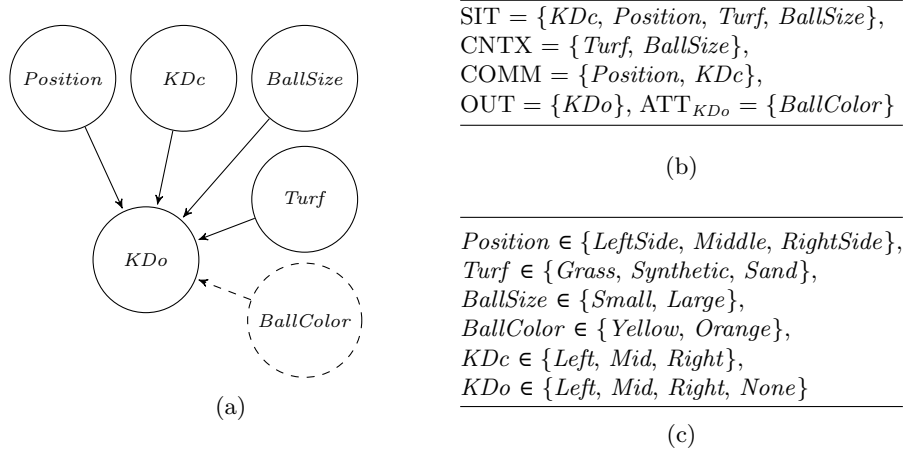*KDo* ∈ {*Left, Mid, Right, None*}

(c)

(a)

Fig. 2: Capability Model for the *BallKick* task. (a) depicts the Bayesian Network, (b) shows the type of each variable in the model and, (c) describes the values each variable can take.

4 Building Capability Models

Building a model involves identifying the right factors to include in the Bayesian Network and learning the conditional probabilities associated with it. First, we present a method to learn the conditional probabilities assuming the learner knows the right factors and structure of the network is fixed. Later, we describe our method to refine the model if need be. Here we introduce some notation.

– A Bayesian Network is a tuple (G, θ)
   • $G \equiv (V, E)$ is a graph, V is the set of nodes and E is the set of edges.
   • V = SIT ∪ OUT are random variables with Multinomial distribution
   • E capture the conditional dependencies amongst the nodes V
   • θ parameterize the conditional probability distributions
– Q ⊂ SIT are variables a learner can control in an experiment. COMM ⊂ Q. An instantiation[3] of Q is called a *Query*.
– Instantiations of CNTX, COMM and OUT are called *Context*, *Command* and *Outcome* respectively. *Situation* = *Context* ∪ *Command*

---

[3] An instantiation of a set of random variables is a mapping from variables in the set to values in their domain. $Query \leftarrow \cup_{q \in Q}\{q : v_q\}$ where $v_q \in Domain(q)$

4.1 Active Learning for Bayesian Networks

We adopt a Bayesian approach to learn parameters and use the algorithm presented in [16] to build a distribution over θ. We describe it here in short.

The algorithm starts with a prior $p(\theta)$ and builds a posterior $p'(\theta)$ by actively experimenting with the subject. In each experiment, it picks a *Query* and requests the subject to perform the task. Variables in SIT $\setminus$ Q either have a fixed value (Eg. *Turf* in the *BallKick* task) or are assigned some value by the environment. A standard Bayesian update on the prior p, for the parameters of the conditional distributions identified by the *Situation* i.e. $P(o|Situation)\ \forall o \in$ OUT, based on the *Outcome* yields the posterior p'. The posterior p' becomes the prior for the next experiment.

To generate a *Query* from the current estimate of $p(\theta)$ we need a metric to evaluate how good the current estimate is. We can then quantify the improvement in the estimate brought about by different queries and pick the one which leads to the biggest improvement. Let $\theta^{\star}$ be the true parameters of the model and θ' be a point estimate. $\sum_{o \in \text{OUT}} D_{KL}(P_{\theta^{\star}}(o)||P_{\theta'}(o))^4$ denotes the error in point estimate θ', where $P_{\theta^{\star}}(o)$ and $P_{\theta'}(o)$ are distributions of variable o parameterized with $\theta^{\star}$ and θ' respectively. $\theta^{\star}$ is not known, but we do have $p(\theta)$, which is our belief of what $\theta^{\star}$ is given the prior and observations. Error in point estimate θ' with respect to $p(\theta)$ can be quantified as in Eq. (1)

$$\text{Error}_p(\theta') = \sum_{o \in \text{OUT}} \int_{\theta} D_{KL}(P_{\theta}(o)||P_{\theta'}(o))p(\theta)d\theta \tag{1}$$

$$\text{ModelError}(p) = \min_{\theta'} \text{Error}_p(\theta') \tag{2}$$

We use ModelError(p) of a distribution p, defined in Eq. (2), as the measure of quality for the estimate $p(\theta)$. Lower the ModelError, better the estimate. We would want to see observations that reduce the ModelError associated with $p(\theta)$ to improve our estimate. But the learner can only control the *Query*. For a particular *Query* we take an expectation over possible observations to evaluate the Expected Posterior Error (EPE) as defined in Eq. (3). In every experiment the learner picks the *Query* with the lowest EPE.

$$\text{EPE}(p, Query) = E_{\Theta \sim p(\theta)}\big(E_{Outcome \sim P_{\Theta}(\text{OUT}|Query)}(\text{ModelError}(p'))\big) \tag{3}$$

In Eq. (3) p' represents the posterior obtained after updating prior p with sample drawn from $P_{\Theta}(\text{OUT}|Query)$. For further details, please refer [16]. θ_T as defined in Eq. (4) parameterize the conditional probabilities of the Capability Model for task T. We compute θ_T using the learned distribution $p(\theta)$.

$$\theta_T = \int_{\theta} \theta p(\theta)d\theta \tag{4}$$

---

[4] $D_{KL}(P_1(o)||P_2(o)) = \sum\limits_{v_o \in Domain(o)} P_1(o = v_o)\log\Big(\dfrac{P_1(o = v_o)}{P_2(o = v_o)}\Big)$, P_1 and P_2 are distributions of o.

4.2 Model Refinement

Every subject may have a different set of extrinsic factors relevant for a task. If a subject only detects balls of a certain color, a variable *BallColor* should be included in the model for the *BallKick* task. We assume the learner starts with a minimal set CNTX and variables representing some of the relevant factors may be missing. In Section 3 we defined ATT_o to be attributes of variable $o \in \text{OUT}$ not explicitly accounted in the model. We assume that the missing variables, if any, belong to these sets. Including all of them makes the model unnecessarily large and difficult to learn. We need a metric to quantify the dependence of variables in OUT on the attributes to identify the relevant ones.

An attribute $A_j \in \text{ATT}_o$ is relevant if the subject's performance (distribution of o) is drastically different for different values of A_j in at least one of the observed *Situation*s. In each experiment, attributes are chosen randomly independent of the *Situation*. After every experiment, the learner computes $\hat{P}(A_j|Situation)$ and $\hat{P}(o|A_j, Situation) \forall o \in \text{OUT}, \forall A_j \in \text{ATT}_o$. $\hat{P}$ is the estimate of the true distribution from the observations in the past experiments. We use the metric defined in Eq. (5), to quantify the dependence of o on the attribute A_j in a *Situation*.

$$R(o, A_j|Situation) = \frac{I(o, A_j|Situation)}{\min(H(\hat{P}(o|Situation)), H(\hat{P}(A_j)))} \tag{5}$$

$$I(o, A_j|Situation) = H(\hat{P}(o|Situation)) - \sum_{a \in Domain_{valid}(A_j|Situation)} H(\hat{P}(o|A_j = a, Situation))\hat{P}(A_j = a) \tag{6}$$

In Eq. (6), $H(P)$ is the entropy of distribution P and $Domain_{valid}(A_j|Situation)$ are values of A_j that have been observed sufficiently in a *Situation*. As A_j is sampled independently of the *Situation*, $\hat{P}(A_j) \approx \hat{P}(A_j|Situation)$. Therefore, $I(o, A_j|Situation)$ is approximately the Mutual Information of o and A_j given the *Situation*, and $R(o, A_j|Situation)$ is the Coefficient of Mutual Information. If $R(o, A_j|Situation)$ is greater than threshold R_{Th}, the learner adds A_j to CNTX, updates the parents of variable o in the graph and creates conditional probability tables for the updated graph.

Algorithm 1 outlines the overall method. Function BestQuery (Line 10) computes a *Query* using the approach described in Section 4.1. Function IdentifyDependence (Line 16) applies the model refinement method presented above.

5 Quantifying Capabilities

To determine how well a robot performs a task in different scenarios, we need a reference that captures the expected performance and, a metric that quantifies how the robot fares against this standard. We assume the reference for task T, is a distribution of the outcome variables conditioned on the commands i.e. $P_{\text{ref}}^T(\text{OUT}|\text{COMM})$. Eq. (7) shows a possible reference for *BallKick* task (Fig. 2).

$$P_{\text{ref}}^{BallKick}(KDo|KDc, Position) = \begin{cases} 1 \text{ if } KDo = KDc \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

Algorithm 1: LearnModel$(\text{maxIter}, \text{R}_{\text{Th}})$
---
1 From Domain Knowledge
2   Construct $G \equiv (V, E)$
3   Initialize Q, $p(\theta)$
4 SituationsObserved $\leftarrow \varnothing$
5 Initialize $\hat{P}$
6 $i \leftarrow 0$
7 **while** $i < \text{maxIter}$ **do**
8 $\quad i \leftarrow i + 1$
9 $\quad$ ATTIncluded $\leftarrow \{\}$
10 $\quad Query \leftarrow \textbf{BestQuery}(p(\theta), \text{Q})$
11 $\quad Attributes \leftarrow \textbf{SampleUniform}\big(\bigcup_{o \in \text{OUT}} \text{ATT}_o\big)$
12 $\quad Outcome,\ Situation \leftarrow \textbf{Experiment}(Query,\ Attributes)$
13 $\quad$ SituationsObserved $\leftarrow$ SituationsObserved $\bigcup Situation$
14 $\quad p(\theta), \hat{P} \leftarrow \textbf{Update}(p(\theta), \hat{P}, Situation, Outcome, Attributes)$
15 $\quad$ **for** $o \in \text{OUT}$ **do**
16 $\quad\quad$ ATTIncluded$[o] \leftarrow$
               $\quad\quad \textbf{IdentifyDependence}(\hat{P}, \text{ATT}_o, \text{SituationsObserved}, \text{R}_{\text{Th}})$
17 $\quad p(\theta), G, \text{Q} \leftarrow \textbf{Modify}(p(\theta), G, \text{Q}, \text{ATTIncluded})$
18 **return** $(G, p(\theta))$

This reference implies that a robot is expected to always kick in the commanded direction. $Score_T(Context)$, defined in Eq. (9), denotes how well a robot fares at a task T in certain $Context$. Lower score values indicate poor performance. We say a robot functions well in a $Context$, if the score is higher than a threshold.

$$Mismatch(Context) = \frac{\sum\limits_{Command} D_{KL}\big(P^T_{\text{ref}}(\text{OUT}|Command)||P_{\theta_T}(\text{OUT}|Situation)\big)^{5,6}}{|Domain(\text{COMM})|}$$

$$(8)$$

$$Score_T(Context) = \frac{1}{1 + Mismatch(Context)} \tag{9}$$

6 Results

We present results of applying our method in modelling a NAO robot (Fig. 3a) performing the task of kicking a ball and a Pepper robot (Fig. 3b) picking up different types of objects and clearing them off a table.
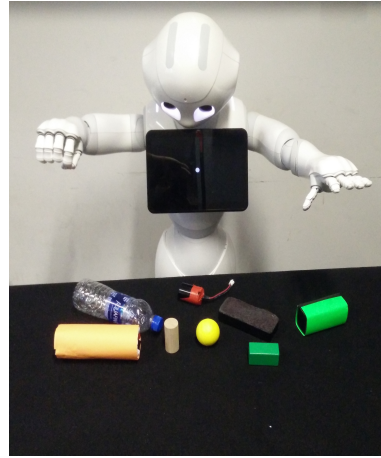
6.1 BallKick Task

Here we demonstrate advantages of the active experimentation approach and our model refinement algorithm. Our approach builds a model, but without knowing

---
[5] P_{θ_T} is the distribution parameterized by θ_T. For a task T, θ_T is defined in Eq. (4)
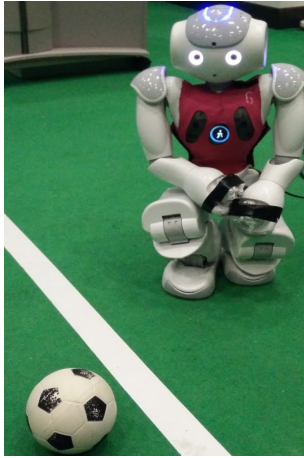
(a) NAO   (b) Pepper

Fig. 3: Subjects for experiments

the ground truth we cannot determine how good the learned model is. So we programmed the robot to behave according to a predefined model with parameters $\theta_{predefined}$. We use $D_{KL}(P_{\theta_{predefined}}(\text{OUT})||P_{\theta_{BallKick}}(\text{OUT}))$ to determine how close the learned and predefined models are. $\theta_{BallKick}$ is defined in Eq. (4), for the *BallKick* task. Fig. 4a shows the learner's initial guess of the Bayesian Network.

Passive vs Active Passively observing a subject, where you do not control the scenarios in which you witness it perform a task, is equivalent to randomly picking *Situation*s to build a model. We learned a model using our approach and another one by randomly picking *Situation*s. Fig. 4b depicts the Bayesian Network for the predefined model. We experimented with a single ball on a synthetic turf and thus variables *Turf* and *BallSize* were dropped in the predefined model. A subtle point to note, the experiments were noisy. The learner chose a particular *Situation*, sampled a direction from $P_{\theta_{predefined}}(KDo|Situation)$ and commanded the robot to kick in this direction. However, owing to noisy perception and actuation, sometimes the robot kicked in directions it wasn't commanded to. Fig. 5a shows that despite noisy experiments the active approach converged faster.

Model Refinement We programmed the robot to detect balls of any color but only a specific size. If presented with a ball of a different size, the subject would not detect and thus won't kick i.e. *KDo* would be *None*. Fig. 4c depicts the Bayesian Network for the predefined model and Fig. 4a shows the learner's initial guess. The learning curve for this model is shown in blue in Fig. 5b. The learner correctly identified the missing variable to be *BallSize*. Once identified, the algorithm resets the conditional probabilities (hence the jump in the trend) and restarts the learning process with the updated model. To show that incorporating relevant attributes yields a model better representative of the subject,
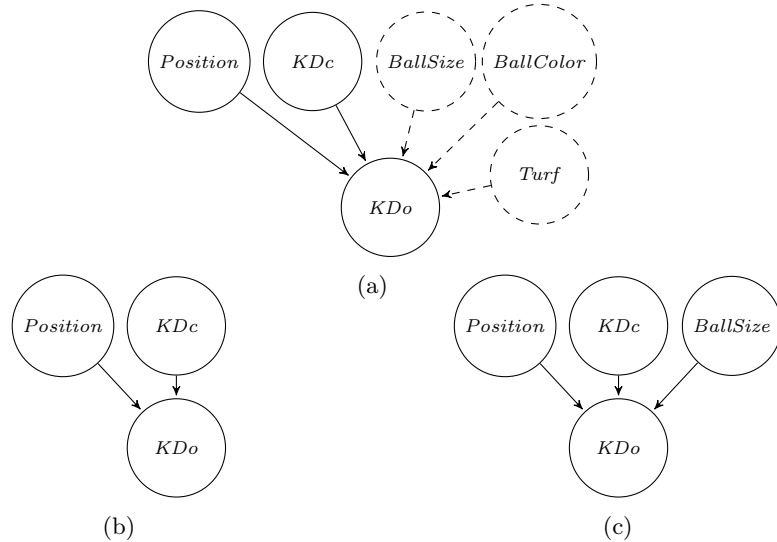
Fig. 4: (a) depicts the learner's initial guess of the model and, (b) and (c) depict the Bayesian Network of the predefined model. Nodes are as defined in Fig. 2c

we learned another model including *BallSize* from the start. As can be seen in Fig. 5b, the model that included *BallSize* from the start (in green) converged to a lower *KL Divergence* compared to the model that did not (in blue). Moreover, post refinement the trends for both models are almost same.

We tested in simulation, how our approach fares as the number of missing attributes increase. In these experiments the robot could only detect balls of a certain size and the kicked ball would randomly end up in any of the three possible directions i.e. *BallSize* and *Turf* were the missing relevant variables. To simulate noisy experiments we sampled *KDo* uniform randomly 20% of the time. We performed multiple runs and the results in Fig. 5c show that including all the relevant variables gives a better model. More the number of missing variables, more the number of experiments needed to identify them all. Moreover, it becomes progressively harder. As a variable gets included in the model, possible *Situation*s increase. While evaluating $R(o, A_j|S)$ (Eq. (5)), we only consider the distributions of o conditioned over values of A_j that have been observed more than a certain number of times in the *Situation* S. The active learning algorithm avoids repeating *Situation*s and thus it becomes incrementally harder to identify relevant variables.

6.2 Pickup Task

We programmed a Pepper robot to detect and pickup objects of three shapes viz. spherical, cubical and cylindrical. We experimented with two sets of weights and two sizes for each shape (in total 12 types of objects). Fig. 6 depicts the Capability Model for the *Pickup* task. A *Context* (*Size*, *Shape* and *Weight*)
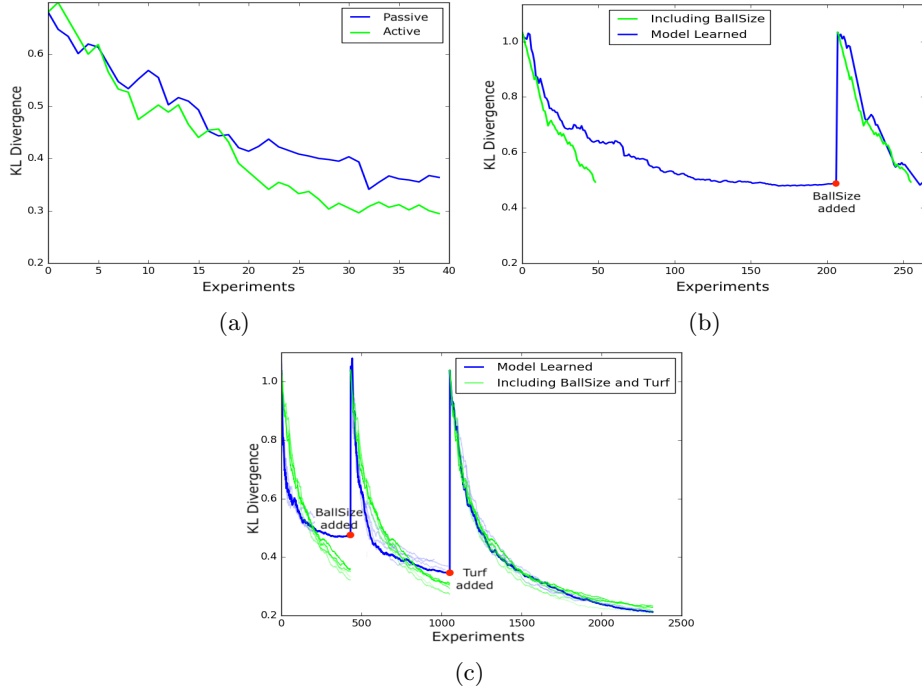
Fig. 5: (a), (b) and (c) depict the trend in $D_{KL}\big(P_{\theta_{predefined}}(KDo)\|P_{\theta_{BallKick}}(KDo)\big)$. (a) compares the trends when learned actively vs passively. In (b) and (c), curves in blue depict trends for the learner's model as it identifies relevant factors to include, red points mark the instances when a new variable is added and curves in green depict the trend if the model were initialized with the right variables.

denotes an object type. In every experiment Pepper was asked to pick up a particular type of object with one of its arms.

We performed 2 trials with 70 experiments each to learn the conditional probability tables associated with the Capability Model. We computed $Score_{Pickup}$, as defined in Eq. (9), for each object type using the reference P_{ref}^{Pickup} defined in Eq. (10). We identified object types with score higher than a threshold at the end of both trials, as favourable for Pepper to pickup.

$$P_{\text{ref}}^{Pickup}(Pick|Arm) = \begin{cases} 1 \text{ if } Pick = Success \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

Having knowledge of the scenarios a robot is well suited for could help in a collaborative task. To demonstrate this, we employed the robot along with a human to clear a cluttered table. In every experiment, the human cleared all but 4 objects off the table and, the robot had to clear the rest. The robot was allowed 3 tries per object (12 in total). We performed such experiments in two settings. In the first setting, the human randomly selected objects for the robot to pick

up and in the second, the human only selected objects of favourable types. We conducted 5 experiments in each setting. Table 1 summarizes the results.

Settings	Number of objects cleared	Number of tries
Objects of any type	1.8 ± 0.98	8.8 ± 1.7
Objects of favourable types	3.4 ± 0.49	6.6 ± 1.2

Table 1: Results of the clear-the-table task ($\mu\pm\sigma$) after 5 experiments per setting.

Performance at the task is better in terms of number of tries as well as number of objects cleared, when the robot is employed in a favourable scenario.

A video outlining our work can be found at `https://youtu.be/_9fm3U8OvHE`.

SIT = {*Shape, Size, Weight, Arm*}
CNTX = {*Shape, Size, Weight*},
COMM = {*Arm*}
OUT = {*Pick*}, ATT$_{Pick}$ = {}

(b)

Arm ϵ {*Left,Right*},
Shape $\in$ {*Ball,Box,Cylinder*},
Size $\in$ {*Small,Large*},
Weight $\in$ {*Light,Heavy*},
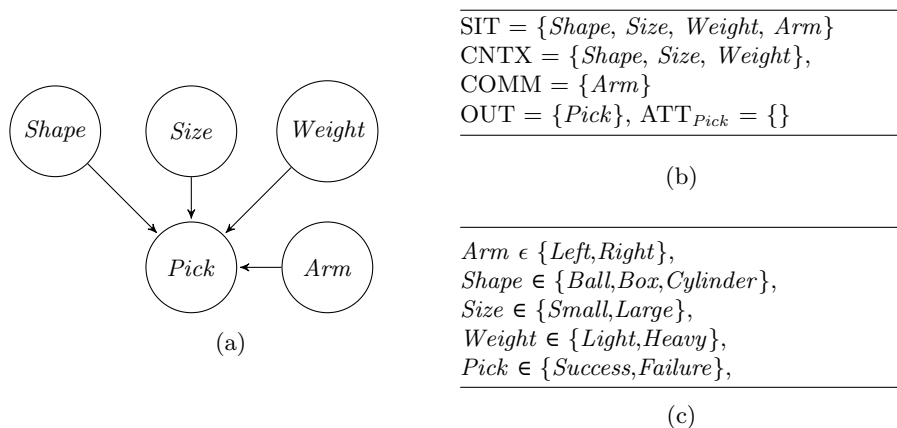*Pick* $\in$ {*Success,Failure*},

(c)

(a)

Fig. 6: Capability Model for the *Pickup* task. (a) depicts the Bayesian Network, (b) shows the type of each variable in the model and, (c) describes the values each variable can take.

7 Conclusion and Future Work

Building models of unknown robots becomes all the more relevant as robots become increasingly multifunctional. To the best of our knowledge, the problem of inferring capabilities from appearance and active experimentation is novel and yet unexplored. We presented an algorithm to build capability models from experiments and showed results with a NAO and a Pepper robot at two different tasks. Experimenting with a physical system requires an operator. However, we still need a systematic approach to design experiments. Our algorithm can serve as a tool for humans in determining, "What can this robot do?"

As an extension to this work we intend to develop an approach to draw inferences about the capabilities of a robot from its appearance and specifications. A robot's capabilities may have certain limitations which are intrinsic and others which it may learn to overcome. Enabling the learner to distinguish between the two and experiment accordingly is an interesting future direction. Moreover, we assumed that experimenting with the subject in any *Situation* has the same cost. Depending upon the task this may be quite far from reality. Learning models in such scenarios is another interesting problem.

8 Acknowledgements

References

1. T. M. Mitchell, P. E. Utgoff, and R. Banerji, "Readings in knowledge acquisition and learning," B. G. Buchanan and D. C. Wilkins, Eds., 1993, ch. Learning by Experimentation: Acquiring and Refining Problem-solving Heuristics.
2. P. D. Scott and S. Markovitch, "Experience selection and problem choice in an exploratory learning system," *Machine Learning*, 1993.
3. Y. Gil, "Learning by experimentation: Incremental refinement of incomplete planning domains," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, 1994.
4. J. J. Gibson, "The theory of affordances chapt," 1977.
5. E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, 2007.
6. A. Dearden, Y. Demiris, L. Kaelbling, and A. Saffotti, "Learning forward models for robots." IJCAI-INT JOINT CONF ARTIF INTELL, 2005.
7. L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation," *IEEE Transactions on Robotics*, 2008.
8. M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive science*, 1992.
9. A. Baranès and P.-Y. Oudeyer, "R-iac: Robust intrinsically motivated exploration and active learning," *IEEE Transactions on Autonomous Mental Development*, 2009.
10. A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, 2013.
11. C. Wang, K. V. Hindriks, and R. Babuska, "Active learning of affordances for robot use of household objects," in *IEEE-RAS International Conference on Humanoid Robots*, 2014.
12. S. Forestier and P. Y. Oudeyer, "Modular active curiosity-driven discovery of tool use," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.
13. S. Forestier, Y. Mollard, and P. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *CoRR*, 2017.
14. S. Hofer and O. Brock, "Coupled learning of action parameters and forward models for manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.
15. J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Transactions on Autonomous Mental Development*, 2012.
16. S. Tong and D. Koller, "Active learning for parameter estimation in bayesian networks," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, 2000.