A Generative Probabilistic Model for Learning Complex Visual Stimuli

Alexey Potapov\textsuperscript{1,2} and Vita Potapova\textsuperscript{2}

\textsuperscript{1}Department of Computer Photonics and Videomatics, ITMO University, St. Petersburg, Russia
\textsuperscript{2}Faculty of Liberal Arts and Sciences, St. Petersburg State University, St. Petersburg, Russia
pas.ai.cv@gmail.com, elokkuu@gmail.com

Abstract
The problem of representing and learning complex visual stimuli in the context of modeling the process of conditional reflex formation is considered. The generative probabilistic framework is chosen which has been recently successfully applied to cognitive modeling. A model capable of learning different visual stimuli is developed in the form of a program in Church (probabilistic programming language). NAO robot is programmed to detect visual stimuli, to point at selected stimuli in a sequence of trials, and to receive reinforcement signals for correct choices. Conducted experiments showed that the robot can learn stimuli of different types showing different decision-making behavior in a series of trials that could help arranging psychophysiological experiments.

Keywords: Cognitive robotics, Probabilistic programming, Visual stimuli, Conditional reflex

1 Introduction

Biologically inspired robotics pays attention to different aspects of living things including both bodies and behavior. The latter is studied on very different levels starting from neurons and ending with higher cognitive abilities. However, most works are focused either on low-level or high-level aspects. The latter are usually modeled with cognitive architectures \cite{1-3}, which capabilities, however, are highly dependent on low-level control and perception modules. At the same time, these low-level modules usually perform some fixed operations. This results in limiting intermediate level control to some set of standard tasks as picking up an object or navigating to a destination.

This can also be seen on the problem of semantic grounding in sensomotoric data. The basic idea consists in detection of co-occurring semantic categories and linguistic units \cite{4}. These investigations have high theoretical and practical significance. However, they have certain limitations, because of restricted perceptual information representations, which can capture restricted (and predefined) set of regularities in sensorial and linguistic input. The most common and strong restriction consists in consideration of only objects (ball, cup, etc.) and their features (shape, color, etc.) as semantic categories or visual concepts \cite{4, 5}. In more recent works \cite{6}, spatial relations are also used to ground...
adverbs (in addition to verbs grounded in actions), but this possibility is defined a priori. It is interesting to point out that utilized learning techniques do not exceed those used in early classical cybernetic models of conditional reflexes [7, 8]. Here, a semantic category and a linguistic unit are just two stimuli, between which a conditional link is established.

Capabilities of animals to form conditional reflexes are far beyond that of robots. Without these capabilities, robots will be highly limited in their higher cognitive functions. Surprisingly, detailed models of general mechanisms of conditional reflex are rare. A t the same time, some authors believe [9] that classical (Pavlov) and operant conditioning are key mechanisms in the learning of any adaptive behavior of animals, and robot training methods are still very far from this.

Models of operant (not classic) conditioning are developed in more details in robotics. However, they mostly address the problem of complex structure of motor acts [7]. Traditionally, robotic conditioned reflex systems are implemented in supposition that stimuli (or events such as ‘hearing a bell’, ‘seeing a light’, etc.) are already extracted and identified (in different trials) [9]. At best, visually observed objects are used as stimuli [10].

At the same time, experiments with animals reveal their ability not only to respond to any circle or red object, but also to learn to select the darkest object, object with certain number of dots, and so on [11, 12]. Interestingly, chickens (just as most cognitive robots) cannot learn to choose stimulus based on its relative brightness. How can we supply robots with abilities similar to that of higher animals? At least, such diverse stimuli should be representable in sensory systems of robots.

In order to investigate this problem, we placed a robot in conditions similar to those used in some experiments on animal’s conditioning. The robot repeatedly observes several “feedboxes” with different marks (we use geometric figures). It should point at a correct one to receive reinforcement. To do this, it should learn a correct rule, which can be based on relative or absolute size, position, shape, or brightness of stimuli.

This task states such questions as how to represent these rules and how to search for them. Previously [13] we used logic rules, which were inferred by genetic programming. This solution is technically admissible, but is neither flexible, no biologically plausible. More appropriate approach is to use probabilistic generative models, which are successfully applied in cognitive modeling [14].

The main contribution of our work consists in application of the framework based on generative probabilistic models to intermediate-level perception (involved in conditioning) in addition to high-level cognition.

2 Methodology

We considered the task of selecting “feedboxes” with rewards on the base of marks (stimuli) placed on them. In each trial, three “feedboxes” were given, and strictly one of them “contained” a reward. Marks on “feedboxes” with rewards followed the same rule within a series of trials.

We programmed a NAO robot to select a stimulus from several visually observed stimuli and to point at the selected stimulus after its head is touched, and then to interpret the next touch as the positive feedback. Processing of images was carried out by our previously developed algorithms [14]. Derived descriptions of each stimulus contain coordinates (x, y) within its bounding box, brightness (b), size (sz), position of the bounding box in the image (pos = left, middle, right), and shape (sh = circle, triangle, square). Thus, on each trial, the robot has a history of previous trials including the correctness of the selected stimuli, and it should determine the most probably rewarded stimulus.

Representing rules for selecting correct stimuli in predicate logic is straightforward [13], however inductive inference of these logic rules is not supported by deductive inference engines of such logic languages as Prolog, and is needed to be separately implemented, e.g. on the base of genetic algorithms or some other metaheuristic search techniques. Probabilistic programming allows for representing rules for selecting stimuli and inferring these rules from.
PPLs usually extend existing languages preserving their semantics as a particular case. Programs in these languages typically include calls to (pseudo-)random functions. PPLs use an extended set of random functions corresponding to different common distributions including Gaussian, Beta, Gamma, multinomial, etc. Evaluation of such a program with random choices is performed in the same way as evaluation of this program in the base (non-probabilistic) language.

However, programs in PPLs are treated as generative models defining distributions over possible return values, and their direct evaluation can be interpreted as taking one sample from corresponding distributions. Multiple evaluation of a program can be used to estimate an underlying distribution.

PPLs go further and support programs defining conditional distributions. Such a program contains a final condition indicating whether the result of program evaluation should be accepted or not (in some languages an "observe" statement can be placed anywhere to impose conditions on intermediate results). Conditional probabilities specified by probabilistic programs correspond to posterior probabilities of generative models given some data, thus their estimation is very useful and can be directly applied to machine learning problems and probabilistic inference.

We will use Church [14] – a probabilistic language based on Scheme, in which several inference methods are implemented including basic rejection sampling, and Metropolis-Hastings sampling.

3 Model

We developed a probabilistic model capable of estimating probabilities of stimuli to be rewarded basing on the information from previous trials. This model is a program in Church containing approximately 30 lines of code. In this model, geometric figures extracted from images were represented as lists of feature values (list x y b sz pos sh), e.g. (list 40 125 200 20 0 0). A task on each trial was defined as a list of three such figures (define task (list fig1 fig2 fig3)), where each fig is a list of features.

After pointing at the selected figure for the current task, the robot received reinforcement if the choice was correct. Then, this task was transformed to the form (list correct? fig1 (list fig2 fig3)), where correct? is #t or #f depending on presence of reinforcement, fig1 is the selected figure, fig2 and fig3 are two remaining figures. This list extended data list (as its new element using cons), which was empty on the first trial and increased its size per one element after each trial. Given data and task, the model estimated posterior probabilities of fig in task to be the correct answer. To do this, rules were stochastically generated. These rules consist in conjunction of elementary tests.

In the model, each test is represented as a list (list n op r v), where n is the index of the feature to check, op is the index of the operation (>, <, = are encoded with 0, 1, 2), r indicates if the feature value of the figure should be compared to the feature values of other figures or some absolute value, v is the absolute value of the feature to be compared with. The following function generates a random test.

(define (gen-test-params)
  (let* ((n (random-integer 6))
         (op (random-integer 3))
         (r (flip))
         (v (random-integer (list-ref '(100 200 256 100 3 3) n))))
    (list n op r v)))

A random rule consists of a random number (up to 3 in our experiments) of random tests:

(define n-test (+ (random-integer 3) 1))
(define rule-params (repeat n-test gen-test-params))

Generated rules are transformed by rule-proc into functions that check if some stimulus in a task is selected correctly and function check is used to test if the given rule is valid for the selected stimulus with respect to all other figures.
Instead of selecting one best rule as was done in [13], we can directly infer posterior probabilities of figures in task to be solutions. To do this, one should hypothesize, which figure is the rewarded stimulus, convert task to the entry of data using this assumption, and perform conditional inference using this extended data. If this hypothesized stimulus is used as the query expression, its marginal probabilities will be calculated (generated rules as random variables will be summed out).

If a hypothesis contradicts former trials in data, its probability will be zero. If it doesn’t contradict, but requires complex rules (with small prior probabilities), its posterior probability will be lower. If it doesn’t contradict many different rules, its probability will be higher. Thus, this approach helps to distinguish between hypotheses, for all of which correct rules exist.

The overall model will be as follows.

```lisp
(executing this will yield a list of 1000 samples, which can be passed to hist function to construct a histogram.
```

4 Experiments

At first, let us consider and discuss several characteristic series of trials, which were presented to the robot. Consider (fig. 1) the following sequence of trials with the assigned probabilities (stimuli are depicted in a refined form for better perception).

On the first trial, data is empty, and probabilities are nearly equal. Difference in these probabilities originates in estimating probabilities via sampling. Here, the rightmost figure (circle) was occasionally assigned the highest probability and selected by the robot. And it appeared to be the correct choice.

On the second trial, the difference in probabilities is not random. Again, the highest probability was assigned to the rightmost circle, which has many common features with the previous correct figure. Consequently, more rules satisfying the first trial also point at this figure. However, many other rules are possible, which don’t contradict the first trial, so the difference in probabilities is not very large. Again, this choice appeared to be correct.

On the next trial, the highest probability is assigned to the rightmost figure (triangle). Evidently, since there is no figure with the color of the previous successful figures, there are two simplest rules remaining – choose the rightmost figure, and choose the circle. Apparently, the figure in the middle
received lower probability since it is neither circle nor rightmost (but there are still many more complex rules that can account for this figure). Difference in probabilities for the rest two figures is due to random sampling. It can be seen by executing mh-query once again (one can obtain such probabilities as 46%, 23%, 31% instead of 33%, 26%, 41%). This choice appeared to be incorrect. However, this incorrect choice effectively rules out alternative explanations, and the robot selects correct figures (circles) during the following trials.

This behavior of the model seems quite reasonable. However, it resembles rather behavior of humans than of animals, but this can be ‘fixed’ if we require the model to follow the rules not strictly, but with some probabilities. Then, the model will be tolerant to outliers, but will require more trials to form the reflex. Possibility to change the model’s behavior in a desirable way by such very simple modifications makes the generative probabilistic framework so powerful.

Let us consider another not very often, but quite typical situation (fig. 2). Here, dark grey figures should be chosen. They are in the middle in the first two trials, but incorrect leftmost and rightmost figures are occasionally chosen.

The third trial looks strange – 100% probability is assigned to the incorrect answer. Moreover, after including this answer as negative into data, the model didn’t interpret the situation as contradictory, but assigned different probabilities to different figures, and chose the correct answer in the next trials.

This is the result of the local search strategy applied by mh-query in the situation, in which possible solutions constitute disconnected subsets in the space of random choices. Such behavior cannot be considered a critical bug, but one should aware of it. Moreover, the problem of search in inductive or deductive inference is very difficult, and no method (possessing limited computational resources) can solve it perfectly. Any other method will have its own peculiarities.

It should also be noted, that natural intelligence can also demonstrate such behavior. Indeed, humans can overlook some alternative hypotheses. More detailed analysis of conditioning is necessary to see, if similar effects are observable in animals learning. Again, replacing strict conditions with stochastic conditions can greatly reduce this effect.

Consider now the case of more difficult rules consisting of the conjunction of elementary tests (e.g. choose the figure that is not too high and not too bright). Here, more trials (~10) are needed to infer such rules. Normally, the model makes good and bad guesses consequently increasing its confidence in choices being made. After some point, the model starts to assign 100% probability to correct choices. More considerable difference than in the number of necessary trials is in mh-query execution time. Apparently, this difference is due to objective difficulty of such tasks. However, depending on search strategies, there can be very different changes in behavior (in decision-making time and in decision randomness). Again, it is interesting to analyze, how different animals solve tasks of different complexity, with the help of the model.

5 Conclusion

We programmed NAO robot to learn to select rewarded visual stimuli - geometric figures recognized by absolute or relative features. The generative model in Church language was developed to infer posterior probabilities for stimuli to be rewarded conditioned by results of previous trials.
On the qualitative level, some similarities and differences between behavior of the model and animals can be seen. The model shows different behavior for different types of stimuli (based on qualitative or quantitative features, absolute or relative features, simple and compound features).

Apparently, whether the robot can form a conditional reflex based on corresponding stimuli depends on the possibility to represent these stimuli by the model. However, more detailed analysis is possible within this framework.

For example, stimuli determined by several conditions require more trials to be learnt and more time to infer. Or the model can ‘overlook’ a correct rule for stimuli and confidently make an incorrect choice. Such peculiarities caused by the objective properties of tasks and by intrinsic properties of the model (e.g. inference and decision-making strategies) would be interesting to compare with abilities of animals in detail. We believe that such analysis can help to guide psychophysiological experiments.

This work was supported by Ministry of Education and Science of the Russian Federation, and by Government of Russian Federation, Grant 074-U01.

References


