Exploration and Exploitation in Visuomotor Prediction of Autonomous Agents

Laurens Bliek\textsuperscript{ab}

\textsuperscript{a} Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Mekelweg 4, 2628 CD Delft, The Netherlands
\textsuperscript{b} Almende B.V., Westerstraat 50, 3016 DJ Rotterdam, The Netherlands

Abstract

This paper discusses various techniques to let an agent learn how to predict the effects of its own actions on its sensor data autonomously, and their usefulness to apply them to visual sensors. An Extreme Learning Machine is used for visuomotor prediction, while various autonomous control techniques that can aid the prediction process by balancing exploration and exploitation are discussed and tested in a simple system: a camera moving over a 2D greyscale image.

1 Introduction

If robots are ever supposed to work in real, complex and uncontrolled environments, they should be able to learn autonomously rather than be preprogrammed with all the information necessary for their specific task. One way to achieve this is by using prediction: making use of the available sensor values and motor commands to predict the sensor data of the future. If this can be learned successfully, the robot can correctly predict the effect of its own motor commands on its sensor data. One consequence would be the ability to make a distinction between effects on a robot’s sensors that are caused by itself or the environment, and those that are caused by others. For example, if a robot mounted with a camera would move to the left, it is expected that the camera pixels will move to the right, and after performing this command it can choose to perform this command again or choose to do something else. In the field of computer vision, the focus lies mainly on extracting information from the sensor data alone (like in object recognition), while the focus in this paper is to let an agent extract sensorimotor information autonomously. This can be done by making use of the known motor commands and the ability to control them. In theory, this should be possible without providing any of this information to the agent.

Of course, one issue is how to control these motor commands. In control theory, the goal is to choose those motor commands that will move a system towards a desired state by using a reference signal. Prediction techniques can be used to aid in this process. But for the problem described above, there is no external reference signal, since only autonomous agents are considered. The control problem is reversed: prediction (as defined in the next section) is the goal, while control can be used as an aid in the prediction process. How to control the motor commands in such a way that prediction is optimised, is still an open question, especially for visual sensors.

The problem can be divided into two parts: the prediction part and the control part. The control part also contains two subproblems, which can be seen as an exploration vs. exploitation trade-off: exploitation in this case is equivalent to moving towards predictable parts of the sensorimotor space so that prediction is performed correctly, while exploration is equivalent to moving towards new parts of the sensorimotor space so that the predictor can learn to predict there as well.
There has been some progress in this area in the past years, like the playful machines of Ralf Der and Georg Martius [2], or the playground experiment of Kaplan and Oudeyer [7]. These techniques have been applied in both simulated and real robots, with impressive results. However, the sensors of these robots were usually low-dimensional, and the predictors were not always very complex. The contribution of this paper is to extend these techniques to agents with visual sensors and more complex predictors to cope with the dynamics of these systems. Visual sensors have the advantage that there is a direct effect of the motor commands on the sensor data, but a drawback is that visual data is in general high-dimensional.

In Section 2 the prediction problem and the used prediction technique will be presented, while Section 3 will focus on control techniques that are able to find a balance between exploration and exploitation in the context of prediction for autonomous robots. In Section 4 some of these techniques are applied in an experiment with a moving $32 \times 32$ pixel camera. Section 5 shows the results of these experiment, and Section 6 concludes this paper.

2 Prediction

2.1 Problem

Suppose there is an agent with no knowledge about itself or its environment. It contains sensors and motors to interact with the environment, and some computational power to make calculations and store information. The goal of the agent is to ‘understand’ its environment and the influence of its own motor commands on the sensor data. There is no clear definition of this understanding, but since prediction techniques will be used, the goal can be defined as follows: correctly predict the next sensor values $s_{t+1}$, given the current sensor values $s_t$ and motor commands $m_t$, for as much of the reachable sensorimotor space as possible. Here, $t$ is the timestep in discretised time, $S \subseteq \mathbb{R}^p$ is the $p$-dimensional sensor space, $M \subseteq \mathbb{R}^q$ is the $q$-dimensional motor space, and $s_t, s_{t+1} \in S$, $m_t \in M$. It is also assumed that $p >> q$.

Using this notation, prediction can be seen as a function $P : S \times M \rightarrow S$, and the mean-square prediction error $e_{t+1} = \frac{1}{p} \left| \left| P(s_t, m_t) - s_{t+1} \right| \right|^2$ should be minimised for multiple $(s_t, m_t) \in S \times M$. This section will discuss one technique to minimise this error, while being able to handle visual data.

The agent should not be told from the outside what is ‘good’ or ‘bad’: the sensor and motor data are all the information it has access to. Although no external signal is available, the prediction problem is still a supervised learning problem, since the desired output is available after one timestep. Not all supervised learning techniques are fit for this problem, because of the following properties:

- Nothing is known about the properties of the data (like the noise and structure, or the relation between motor and sensor values).
- The mapping from input to output can be complex (nonlinear, non-deterministic, etc.).
- Both input and output are high-dimensional.
- Good generalisation is necessary for dealing with noise or with new situations.
- Learning needs to be done online.

One technique that can deal with most of these properties, is a relatively new neural network technique.

2.2 Extreme Learning Machines

Extreme Learning Machines [3] (ELMs) are an efficient method for training a single hidden layer feed-forward neural network. Instead of tuning the output and hidden layer by backpropagating the error, the hidden layer is initialised randomly and remains fixed during the learning process, while the output weights are the only adaptable parameters. Since the output neurons are chosen to be linear, this leads to a linear least-squares problem, while the nonlinear hidden neurons still allow the network to approximate nonlinear functions. The output weights can be computed by using a pseudo-inverse of the hidden layer output.
The ELM algorithm can be summarised as follows:

1. Given a training set \((x_t, y_t)\), with \(x_t \in \mathbb{R}^n\) and \(y_t \in \mathbb{R}^m\), \(t = 1, \ldots, N\), initialise a single hidden layer feedforward neural network by choosing an activation function \(g\), the number of hidden neurons \(\tilde{N}\), and weight matrices \(W \in \mathbb{R}^{\tilde{N} \times n}\) and \(\beta \in \mathbb{R}^{m \times \tilde{N}}\) that consist of the connections between input and hidden layer and between hidden and output layer respectively. A threshold \(b \in \mathbb{R}^{\tilde{N}}\) can also be chosen for the hidden neurons.

2. Calculate the hidden layer matrix \(H \in \mathbb{R}^{\tilde{N} \times N}\) consisting of the values of hidden neurons for each training sample by using \(H_t = g(Wx_t + b)\) for the columns of \(H\).

3. Keep the weights in \(W\) fixed, but adapt the output weights \(\beta\) by using the Moore-Penrose pseudo-inverse \([8]\): \(\beta = YH^\dagger\), with \(Y\) the matrix consisting of desired output values \(y_t\) in each column.

The above algorithm gives the minimum norm least-squares solution to \(\beta H = Y\), where \(\beta H\) is the output of the neural network with fixed hidden layer parameters. Compared to traditional backpropagation algorithms, this algorithm has several advantages:

- Very large learning speed.
- Good generalisation performance because of the small output weights.
- Convergence to global minimum instead of possible local minimum entrapment.
- No need to adjust the learning rate.

Most of these advantages follow from the fact that there is a direct solution to the linear least-squares problem.

The current sensor values and motor commands of the agent can be used as an input of the ELM: \(x_t = \begin{bmatrix} s_t \\ m_t \end{bmatrix}\). If the next sensor value is chosen as the desired output of the ELM, i.e. \(y_t = s_{t+1}\), the ELM can be used for visuomotor prediction. This gives:

\[
P(s_{t+1}, m_t) = \beta g(W \begin{bmatrix} s_t \\ m_t \end{bmatrix} + b).
\]

3 Control

There are several possibilities for choosing a control that will aid the prediction process of the agent. The controller should make the agent explore its sensorimotor space while letting the predictor learn the relation between current sensor and motor values and the sensor values of the next timestep. As will be shown in this section, these are two contradictory goals, and several techniques for finding a balance between these goals will be discussed, as well as their applicability to agents with high-dimensional sensor data. The problem is closely related to the cognitive bootstrapping problem and the exploration vs. exploitation trade-off. The main difference with the latter is that exploration is a goal in itself in this problem, rather than a way to find out how to get to a certain goal.

Note that no external reference signals are available. The goal of the controller is not to move the system towards a desired state, but to aid the prediction process. Therefore, only functions of internal signals can be used in the controller. To be more precise, the motor command \(m_t\) at timestep \(t\) is allowed to depend on the current sensor values and the past sensor and motor values only:

\[
m_t = C(s_t, m_{t-1}, s_{t-1}, m_{t-2}, \ldots m_0, s_0).
\]

Since the prediction error \(e_t(s_{t-1}, m_{t-1}, s_t) = \frac{1}{p}||P(s_{t-1}, m_{t-1}) - s_t||^2\) also depends only on the current sensor values and the past sensor and motor values, it makes sense to use this error to choose the control.
Following the reinforcement learning framework \cite{12}, we use the prediction error to define a reward $r$ for the agent. The controller then chooses actions in such a way that the sum of future rewards is maximised: 

$$m_t = \arg \max_M \sum_{i=t+1}^{T} r_i,$$

for some final timestep $T$. The reward should be chosen in such a way that the controller indeed aids the prediction process of the agent. Even after having chosen the reward, this is not a trivial task, since the prediction errors of the future are not known. One way to maximise the sum of future rewards is discussed in Section 3.5, while several choices for the rewards are discussed in the following subsections.

### 3.1 Minimise prediction error

A first thought might be to let the controller choose those actions that minimise the prediction error. This way, the controller can cooperate with the predictor to make the best possible predictions. The reward could be chosen as $r_t = -e_t$, giving the agent a negative reward or penalty whenever the prediction error is high. Or the reward can be chosen as $r_t = c - e_t$ for some constant $c$, to include the possibility of positive rewards for low prediction errors.

This control technique should greatly increase the efficiency of the prediction process, and the prediction error is expected to decrease in a very short amount of time because the controller can choose those actions for which the prediction error is minimised. However, the sensorimotor space will in general not be fully explored, and the predictor might only specialise in a small part of it, for example by predicting what will happen when the agent does not move, or by staying in situations where sensor values do not change. This is a big drawback of this approach, so a different reward should be chosen. The controller should aid the predictor by providing exploratory behaviour.

### 3.2 Maximise prediction error

Instead of rewarding the agent for those actions for which the prediction error is low, causing the agent to choose predictable actions, the opposite approach might actually lead to exploratory behaviour. Choosing a reward equal to the prediction error, $r_t = e_t$, actually makes sure the agent explores different actions. After all, if it would choose the same action over and over again, the predictor would learn this situation and the prediction error would decrease, causing some other action to give a higher prediction error.

The controller that maximises this reward can be seen as an adversary of the predictor, choosing the actions that have not yet been fully learned to make sure the predictor keeps learning and the agent keeps encountering new situations.

While this approach might work in theory, several researchers \cite{10,11,4} pointed out that there are some drawbacks. Since the controller is actually an adversary for the predictor, it might find ways to prevent the predictor from learning. For example, it might provide such unstable motor behaviour that the sensor data becomes very noisy. Or it might move the system towards a part of the environment that is noisy. If it is possible to move the system towards a part of the sensorimotor space where no learning is possible, this approach can cause practical problems. The controller and predictor should not be adversaries, but complement each other, and there should be a balance between exploratory behaviour and predictability.

### 3.3 Maximise learning progress

Kaplan and Oudeyer \cite{4} used reinforcement learning to drive an agent to situations where the learning progress was maximised. The learning progress is defined as the difference between the average prediction error of a few timesteps ago and the current average prediction error, for example $r_t = \bar{e}_{t-d} - \bar{e}_t$, where $d$ is a delay value, and $\bar{e}_t$ is the average of the prediction errors between timesteps $t - w$ and $t$, for a certain sliding window $w$.

If the agent is learning, the prediction error should be decreasing, so there is a large learning progress. If the agent has learned its current situation, there is no progress, and if the agent cannot learn a specific situation there is also no progress. This solves the problem of staying in simple predictable or complex unlearnable situations.
3.4 Other approaches

The approaches mentioned above are by far not the only ones. There are many more approaches to the problem of finding internal rewards that lead to a balance between explorative and predictive behaviour. An example is the playful machine of Ralf Der and Georg Martius [2], where the concept of homeokinesis is introduced. Information-theoretic approaches also became popular over the past years [1]. Kaplan and Oudeyer improved the approach of maximising the learning progress by splitting the predictor in different experts [5]. And there are also examples of using random actions in combination with chaotic search [14, 13] or confidence [9]. An overview of both information-theoretic and predictive approaches concerning internal rewards is given in [6].

3.5 Chosen reinforcement learning algorithm

Regardless of the choice of reward, the problem remains of choosing those actions that maximise the unknown future rewards. Several algorithms exist for solving this problem [12], but only one of the simplest ones will be considered in this paper, since the focus of this section is mainly in the choice of the reward.

For every action \( m \in M \), we assign a value \( Q(m) \) that says something about the expected reward that follows from choosing this action. This value can change every timestep, but only for the action that was actually chosen, so \( Q_t(m) = Q_{t-1}(m) \) for all actions \( m \) that were not chosen at timestep \( t-1 \). For the action that was chosen at timestep \( t-1 \), the value is updated according to the following rule:

\[
Q_t(m_{t-1}) = (\alpha - 1)Q_{t-1}(m_{t-1}) + \alpha r_t, \tag{3}
\]

where \( \alpha \in [0,1] \) is some constant step-size parameter. This is a simple (myopic) version of the Q-learning algorithm, where only the reward of the next timestep is maximised and \( Q \) depends only on the action. The parameter \( \alpha \) determines the importance of the next reward, compared with the rewards of the past.

Now at each timestep \( t \), the action with the highest value is chosen: \( m_t = \arg \max_M Q_t(m) \). To encourage exploration, at each timestep there is a probability \( \epsilon > 0 \) with which the values are ignored and a random action \( m_t \in M \) is chosen. This is called the \( \epsilon \)-greedy method.

To further encourage exploratory behaviour, optimistic initial values are used: the initial values \( Q_0(m) \) are chosen equal to some constant \( c > 0 \), for all actions \( m \in M \). Choosing \( c \) large enough makes sure that all actions are tried out before the action that maximises the reward will be chosen, since the rewards encountered by the agent will be lower than the initial reward. Values of \( c = 1, \epsilon = 0.2 \) and \( \alpha = 0.9 \) were used in this paper.

4 Experiment

To test whether the techniques discussed above can indeed be applied to agents with visual sensors, an active camera moving over a 2D picture has been simulated (see Figure 1). The picture contains 512 \( \times \) 512 greyscale pixels, the camera 32 \( \times \) 32. The ELM technique from Section 2 has been used for the predictor, in a neural network with 30 hidden neurons. This number of neurons was found manually to give decent approximation capabilities while not requiring too much computation time, but it is not an optimal quantity. Training of the predictor had to occur each timestep, since the prediction error was used by the controller each timestep. The predictor has to predict the next camera pixel values, given the current camera pixel values (with some added white noise) and the horizontal and vertical speed of the camera.

For the control, the three techniques of Section 3 were compared, with a controller that could choose the horizontal and vertical speed of the camera. The three techniques that were compared are: Minimise prediction error (MinPE), Maximise prediction error (MaxPE), and Maximise learning progress (MaxLP). The motor space was discretised into 17 possible motor actions: move up, move down, move left, or move right, by either 1, 2, 4 or 8 pixels, or stand still. The camera was simulated for 4000 timesteps.
To show the drawbacks of some of the control techniques, some parts of the environment have been made noisy, while others have been made homogeneous. At the left and right boundaries of the image, the image was changed to random uniform noise that changed every timestep, to include an area where prediction is not possible. At the top and bottom boundaries of the image, all pixels were given the same grey-scale value, to create an uninteresting part of the environment where prediction is trivial. These noisy and homogeneous boundaries have been chosen to be 32 pixels wide. A good control policy should avoid both of these areas, while mainly exploring the ‘interesting’ part of the image.

Figure 1: Experimental setup: a simulated $32 \times 32$ pixel camera can move over a $512 \times 512$ greyscale image and has to learn the effects of its own movements on its sensor data using a neural network. The green rectangle on the left represents the part of the image that can be seen by the camera, the image on the right represents the output produced by the neural network after learning. Parts of the environment were made noisy or homogeneus to include situations for the agent that are either trivial or impossible to predict.

5 Results

Figure 2 shows the behaviour of the camera for one simulation for each of the three different control techniques. Different simulations caused similar results. As expected, in the MinPE and MaxPE approaches, the camera gets ‘stuck’ in parts of the environment where the reward is maximal. In the MinPE approach, the controller found out that staying in the homogeneous part of the environment gives the highest reward (lowest prediction error). Similarly, the MaxPE approach caused the controller to stay in the noisy part of the environment, where prediction is impossible and the prediction error is maximised. In both cases, the controller had no prior knowledge about the environment so it got in the specific parts of the environment by chance, but once it got there, the optimal strategy was to stay there.

The MaxLP approach shows different behaviour. It also arrived in the noisy or homogeneous parts of the environment by chance, but the reward was too low to stay in these parts of the environment. The reason for this is that the prediction error is low in the homogeneous part of the environment, but it does not decrease, therefore the learning progress is low in this part. In the noisy part of the environment, the prediction error is high, but also does not decrease, therefore the learning progress is low in this part as well. Learning is only possible in the other parts of the environment.

The goal of the controller was to stay clear from the noisy or homogeneous parts of the environment and explore the ‘interesting’ parts. Table 1 shows the percentage of time that the camera was in the noisy or homogeneous part of the environment, for the three different control policies. As expected, only the MaxLP approach gave good results.

Table 1: Percentage of timesteps that the camera was inside the homogeneous area (top/down), and the noisy area (left/right), for each of the three control policies.

<table>
<thead>
<tr>
<th></th>
<th>MinPE</th>
<th>MaxPE</th>
<th>MaxLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage in top/down area</td>
<td>92%</td>
<td>4%</td>
<td>9%</td>
</tr>
<tr>
<td>Percentage in left/right area</td>
<td>0%</td>
<td>91%</td>
<td>7%</td>
</tr>
</tbody>
</table>
These results are not new; letting an agent avoid unlearnable and too simple situations has been done before \cite{4}, but the main contribution of this paper is combining this control policy with a prediction technique in such a way that visual data can be handled. Figure\ref{fig:prediction_error} shows the prediction error for each of the three control policies. It turns out that the homogeneous parts of the environment can be predicted with an error of around 0.001, and the noisy parts with a maximum error of around 0.3. These values can be seen as minimum and maximum values for the prediction error in this experiment.

The MaxLP approach explored mainly the other, more interesting parts of the environment, causing a balance between exploring the environment and exploiting the prediction capabilities. This balance can also be seen in Figure\ref{fig:prediction_error} where the prediction error of the MaxLP approach is somewhere in between the prediction errors of the MinPE and the MaxPE approach.

![Figure 2: Movement of the center of the camera during 4000 timesteps of one simulation for the three control techniques: (a) Minimise Prediction Error (MinPE), (b) Maximise Prediction Error (MaxPE), (c) Maximise Learning Progress (MaxLP).](image)

6 Conclusion

In this paper, an Extreme Learning Machine was used for visuomotor prediction, and several control policies that can aid in the prediction process were discussed. It was shown experimentally that these techniques can be combined to make prediction possible for autonomous agents with visual sensors. The method of maximising the learning progress, combined with visuomotor prediction by an Extreme Learning Machine, causes a balance between exploration of the sensorimotor space, and exploitation of the prediction capabilities. This was showed by applying these techniques to a simulated 32 × 32 pixel camera that could
autonomously control its horizontal and vertical movement, without any prior information about itself or the environment. Parts of the environment that were too noisy or too simple to predict, were avoided, causing the camera to explore the more interesting parts of the environment. Further research will show whether this can be extended to more complex experiments and real-world applications.

Acknowledgment

This research was performed at Almende B.V. in Rotterdam, the Netherlands. The author would like to thank Anne van Rossum for his supervision during this research, and Giovanni Pazienza for providing valuable feedback on this paper.

References


