A Robot Control Algorithm
Leture note for the CISP PhD school May 5-8 2003

Peter Johansen

29. april 2003

Each footstep is a question
Ron Cotterill

Abstract

A proposal is made for a robot control algorithm. It is initialized by a teacher who directs it through a series of example solutions of a problem. Left alone, the control chooses its next action by parsimony: its action attempts to keep the complexity of its interaction with the environment at a minimum.

Robot Learning

We use to think of a control program for a robot as explicitly programmed for each task. It would be preferable that the robot possess a general-purpose program to make it able to learn from its experience. A possible design is proposed which is based on a prediction algorithm. The algorithm attempts to increase the complexity of its experienced sequence of inputs and outputs as little as possible. It attempts to minimize the entropy of its dialogue with the environment. The algorithm searches for the longest uninterrupted sequence of symbol which has occurred previously. It then predicts the next symbols to be the same as the one which appeared in the previous most similar context.

Each action is a question to the environment, and the answer is the next input to the robot. Viewed this way, we group the sequence of alternating outputs and inputs into compound symbols, into output/input pairs. Based on the robot's previous experience - its sequence of compound output/input pairs - it makes a prediction using the lrs algorithm. Its action becomes the output component of the predicted compound symbol, and the input component is the expected next input. This way we hope (know?) that ultimately we will make as good decisions as if the solution had been explicitly programmed. A line of argument should follow the argument that Lempel-Ziv text compression [6] is asymptotically optimal and universal for finite state Markov state machines.

In this note the lrs algorithm is explained, and proposed as robot control algorithm. A problem solution is seen as an orbit in a symbolic dynamical system.

The lrs algorithm

```
variables: memory is initially a string from a finite input alphabet.
           time is initially the length of memory
           input holds next input

- determine the longest suffix of memory which occurs in memory in addition
           to its occurrence as a suffix.
- if there exists several occurrences, choose the latest occurrence.
- predict the symbol immediately following this latest occurrence(of
           the longest suffix)

repeat until EOF
  let suffixset(k)=all positions pos<time such that
                    memory(pos-k:pos)==memory(time-k:time)
  determine the largest k such that suffixset(k) is not empty
  output memory(pos+1)
  input=Read(environment)
  memory=memory followed by input
  time=time+1
end {repeat}
```

Outputs

The robot simultaneusly outputs a symbol and reads a symbol. The symbol that it reads is the predicted next sensor reading. Its output is a command to its effectors, for instance two parameters which are the speed of its left and right motor. If each parameter is quantified by a byte, the output alphabet O is all possible combinations of two bytes, $|O| = 2^{16}$. If we choose to quantify speed more coarsely for instance into $\{no\ motion, low, high\}$ we have $|O| = 3^2$. It is important to get as coarse a quantification as possible in order in decrease the number of possible actions. This in order to increase recognition of previous similar actions.

Inputs

In Tu and Terzopulos' work [9] their fish *focuses* its input, i.e it filters the input for the features relevant to it in its present state. Let us consider the robot in a similar way. At each moment of time it focuses on a particular spatial feature by turning the camera to extract maximal information. Its distance sensors' measurements are coarsened to tell if objects are close or far away. This way, the set of possible inputs is partitioned into equivalence classes as coarsely as possible for correct action to be taken. Next it selects an action that depends upon internal state variables, updates the variables, and subsequently executes the actions. Coarsening of inputs facilitates recognition of previous identical circumstances. Computation of input classes involves among other things extraction of image features.

For the Eyebot robot [1] the input is a combination of the three distance sensors and the image. Each sensor can be quantified into maybe $\{out\ of\ range, 10, 20, 30, 40\}$. The image is 60×80 bits without any feature extraction. This makes it virtually impossible to detect repeated occurrences of input symbols. Hence, the inputs should be described by features. A single feature can for instance be direction to goal. If this direction is quantified into $\{-90, -45,\ 0, +45, +90\}$ degrees, the input alphabet I has cardinality $|I| = 5^4$. This is still a very large number, and a further reduction can be performed for instance by the k-means algorithm.

Teaching

The lrs algorithm can be used to have the robot learn from examples. The robot is shown by a teacher what to do. This means that the teacher in an initial learning phase is telling the robot which actions to take for a given sequence of inputs, for instance by remote control. In order to reduce teaching time, the robot can be taught by a program in a simulator[7]. If teaching is repeated for many trials, one can let the robot determine its next move by the lrs algorithm.

Why is this successful? Remember that the robot, using the lrs algorithm, searches for the most similar previous situation and repeats its previous behavior. It seems reasonable to believe that if the teacher has been sufficiently careful, the correct behavior is induced.

Prediction as Controlling Principle

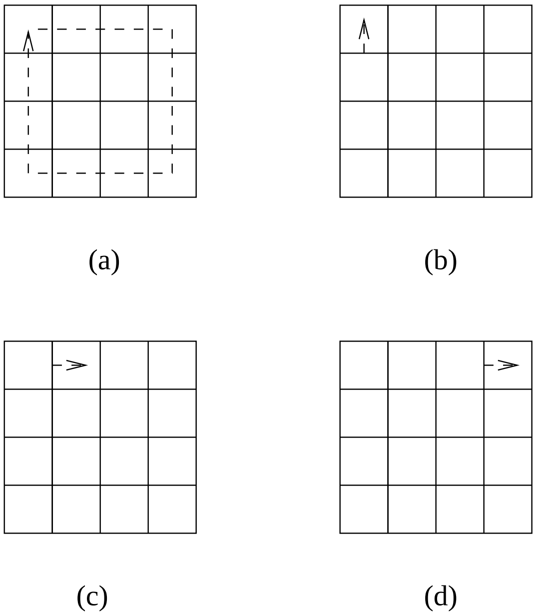
Let us return to the question of designing an uncommitted control algorithm for a robot. Let the robot predict the next compound symbol by the prediction algorithm. The prediction includes its own next output, which causes a change in the environment (for instance a right turn of the robot) together with prediction of the next input from the environment. If teaching has been complete, the environmental input will be predicted correctly.

We next have to decide a sensible memory update by the robot in the case that prediction of next input fails, indicating insufficient teaching. The suggestion is to let the next compound symbol in the sequence be the predicted action (which has allready been executed) together with the actual but unexpected input.

The lrs algorithm was proposed in [4]. In [5] computational experiments were performed with the lrs algorithm analysing a dialogue between a robot and the environment. The difference to that work is that in the present proposal the outcome of the dialogue is predicted based on the output/input pairs in the dialogue.

Wall Following Example

Consider a 4×4 pixel arena. The robot is supposed to find its way around the periphery of the arena.



Figur 1: (a) Intended periodic trajectory of the robot (b),(c),(d) Different initial positions and orientations of the robot

The robot is equipped with two binary sensors, a left and a front sensor. Value 1 indicates next to wall. It can move straight ahead one step S, and it can change its orientation 90° right and then move one step ahead R. Let the training data be a sequence of 4 compound symbols.

$${}^{1^1}S \quad {}^{1^0}R \quad {}^{1^0}S \quad {}^{1^1}S$$

The compound symbol ${}^{1^0}S$ means the left sensor signals next to wall, the front sensor signals no wall, and the output is straight ahead.

If the robot is positioned in first row, first column (figure 1 b), and oriented towards north, it will succesfully circle the inner periphery of the arena.

Let now the robot be positioned at a different initial square: at first row, second column oriented towards east (figure 1 c). It will turn right and its sensors will both show the value 0. This situation the robot has not met before, and the algorithm stops. In order to prevent stopping, the teacher must initially show an action or action sequence for each possible (input, output) pair.

Let finally the robot be positioned in row 1, fourth column oriented towards east (figure 1 d) and with training data

$$ {}^{1^0}S \quad {}^{1^1}S \quad {}^{1^0}R \quad {}^{1^0}S $$

The prediction is ${}^{1^1}S$ which has the effect that the robot attempts to go straight ahead. It is hindered by the wall and as a result does not move. The input component of the compound symbol was correct, and the memory becomes

$$ {}^{1^0}S \quad {}^{1^1}S \quad {}^{1^0}R \quad {}^{1^0}S \quad {}^{1^1}S $$

Next prediction is ${}^{1^0}R$, which correctly causes the robot to turn right and step ahead. The input component is unchanged and the situation is

$$ {}^{1^0}S \quad {}^{1^1}S \quad {}^{1^0}R \quad {}^{1^0}S \quad {}^{1^1}S \quad {}^{1^0}R $$

The robot is back on track and correctly circles the periphery.

Computation

The lrs-algorithm can be implemented by Ukkonen's suffix algorithm [10], but some modifications have to be made in order to compute the necessary updated information in the nodes. Each node has to know the most recent leaf inserted in its subtree. When following the suffix links, the visited nodes can be updated, and information can be stored for delayed updating of ancestor nodes. The problem is that a node may be visited by a suffix link before being updated. When a context is chosen, the safe way is the visit all nodes on the path to the root and choose the optimal context. Nodes can be updated at the same time. The amortized complexity of the suffix algorithm is $O(n)$. The complexity of the modification to keep track of the most recently inserted node is an open problem.

Appendix on Dynamical Systems

Consider the physical world in which the robot is embedded and which changes as result of the control algorithm. It is modelled as a dynamical system. We assume that the physical system is deterministic. We do not consider influence of random fluctuations on the system. Thus, the system's orbit exists as a function of the initial state of the control algorithm and the initial state of the world including the robot's initial position, but only a finite part is known to the control. Control of a continuous physical system would normally be modelled by a differential equation. In this case the model is discrete, like a difference equation, except that the "equation" is a string matching algorithm.

The parameter space – the initial state of the control and the state of the world including the initial pose of the robot – is partitioned into attractor basins. Each set of parameter values define an attractor, and two sets of parameter values are in the same basin if they define the same attractor.

We are now in a position to define what it means that the robot solves a problem. The problems which can be solved are the attractors, and a solution to a problem is an orbit which defines the problem's attractor.

Sampling and Poincaré sections.

The orbit described by a system is either a loop or a non self-intersecting curve in phase space. This is a consequence of the definition of phase space: its coordinates are all variables which decide the system's behavior. Since the system is deterministic, its fate is determined by a specific value of its state variables. When the orbit is measured, it happens with a finite sampling aperture. Thus, each measured value is the integral over a neighborhood in phase space. Let us idealize this situation by partitioning phase space into cubes. A measurement gives us the identity of the cube enclosing the ideal system state. The orbit intersects a face between two neighbor cubes. Generically it avoids the corners of the cubes. In other words, if the orbit did traverse a corner, one could change the segmentation into cubes slightly to avoid this situation. This way measurement gives us a sequence of cube identifiers, of which there is finitely many. The face between two cubes is called a Poincaré section. In physics, one often positions a single section at an appropriate position in phase space. An intersection point between the orbit and the section is by the system mapped into the next intersection point of the orbit and the section. This way a map is induced from intersection point to intersection points which is called the *first return map*. By finite measurement of a system, whose exact evolution we cannot specify analytically, we get a sequence of finite identifiers, which are called symbolic coordinates. The study of quantized dynamical systems is called *symbolic dynamics*.

Symbolic dynamics

Consider a sampled signal, that is a sequence $x_1^\infty = x_1 x_2 \ldots x_i \ldots,\ x_i \in B$, where $B = \{0, 1, \ldots, k-1\}$ is a finite set of k integers. We shall consider the set $\Sigma = \{x_i\, x_2\, \ldots\, x_j\, \ldots | x_i \in B\}$ of semi-infinite sequences. In the parlance of symbolic dynamical systems we consider a map σ from the set Σ of semi-infinite sequences into itself. The map σ removes the first symbol from its argument. σ is called the *shift map*. By an *orbit* we understand the sequence of iterated images of σ. If we want to emphasize the map we say *σ-orbit*

$$\sigma : B^\infty \to B^\infty$$

The orbit of x is the sequence

$$x\,\sigma(x)\,\sigma^2(x)\,\ldots\,\sigma^i(x)\,\ldots$$

We define a distance measure in σ by

$$d(x_1^\infty, y_1^\infty) = \sum_{n=1}^{\infty} |x_n - y_n| k^{-n}$$

We observe first that

$$d(x, y) \leq 1$$

for all $x, y \in \Sigma$.

Let $S_{\epsilon,z}$ be an ϵ-sphere with center z. Let 0 be the sequence of all zeroes. We observe that $\Sigma \subseteq S_{1,0}$ and that Σ is a closed set. Thus, A is compact. We know from mathematical analysis, that a sequence of points in compact set contains at least one point of accumulation. Hence, an orbit has at least one point of accumulation. Let $D(x)$ be the set of accumulation points for the orbit of x. $D(x)$ is called the *attractor* of the orbit of x.

Constructing the attractor

The orbit of a semi-infinite sequence can be computed using a tree as data structure. Elements of the orbit are inserted as paths from the root in a tree, in which the edges are labeled by symbols from B. The node which is reached by the path w is denoted $\overline{w}$. Let x and y be two elements of the orbit, and let n be the least integer such that $x_n \neq y_n$. The distance between x and y is bounded as a function of n as follows.

$$k^{-n} \leq d(x, y) \leq \sum_{i=n}^{\infty} k^{-i} = k^{-n+1} \tag{1}$$

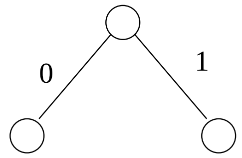
The two orbits then have the path from the root to the node $\overline{x_1\,\ldots\,x_{n-1}}$ in common.

Example 1

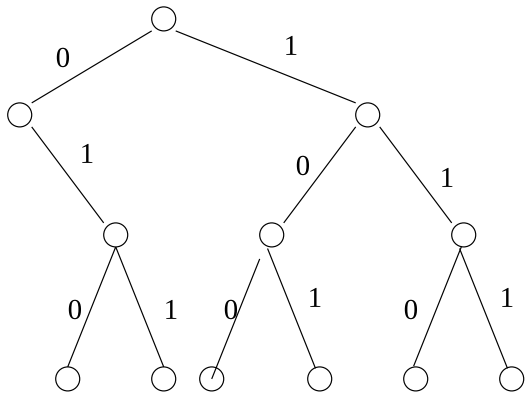
The sequence of alternating symbols $x = 01010101\ldots$ has an attractor that consists of two points, the sequences $010101\ldots$ and $101010\ldots$ and is represented by the tree in figure 2.

Another example

The first six elements of the orbit of the sequence $x = 01110100\ldots$ has been inserted in the tree representing the attractor in figure 3. By inserting still more elements, the attractor can be approximated still better. The *suffix* algorithm inserts the first n orbit elements in time $O(n)$[10].



Figur 2: The sequence of alternating symbols $x = 01010101\ldots$ has an attractor that consists of two points, the sequences $010101\ldots$ and $101010\ldots$



Figur 3: The first six elements of the orbit of the sequence $x = 01110100\ldots$ has been inserted in the tree representing the attractor

Accumulation points

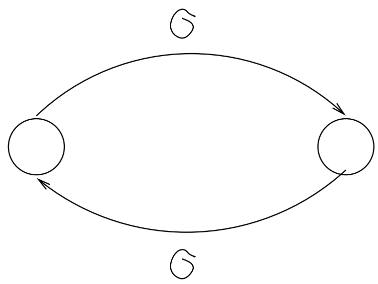
The accumulations points that constitute the attractor can be characterized in terms of the representing tree. First, consider the case that a sequence is eventually periodic, that is after a transient repeats itself periodically: $x = t\,p\,p\dots$ and t and p are shortest possible. Then the leaves of the tree fall into two classes, the period leaves and the transient leaves. The transient leaves represent orbit elements in which not all symbols of t have been removed, and the period leaves represent periodic orbit elements. The tree has finitely many nodes.

Second, the sequence is not eventually periodic. This implies that any two elements have a largest common prefix w. They are represented by paths which are descendants of the node $\overline{w}$. The representing tree has countably many leaves. A consequence is that it has at least one infinite path. In fact, each infinite path ω is an accumulation point of the orbit: It being infinite means that it contains infinitely many nodes. For each node $\overline{w}$, $|w| = n$ there exists at least one orbit element which agrees with ω for the first n elements, and therefore their distance is less than k^{-n}. We can conclude that ω is an accumulation point.

We can deduce a property of the attractor, we will show that σ maps the attractor into itself! It is quite easy to see. It follows from a more general property of the suffix tree. The subtree rooted at the node $\overline{w}$ is a subset of the entire tree. This follows since if wz belongs to the orbit, also z belongs to the orbit.

Example 1 continued

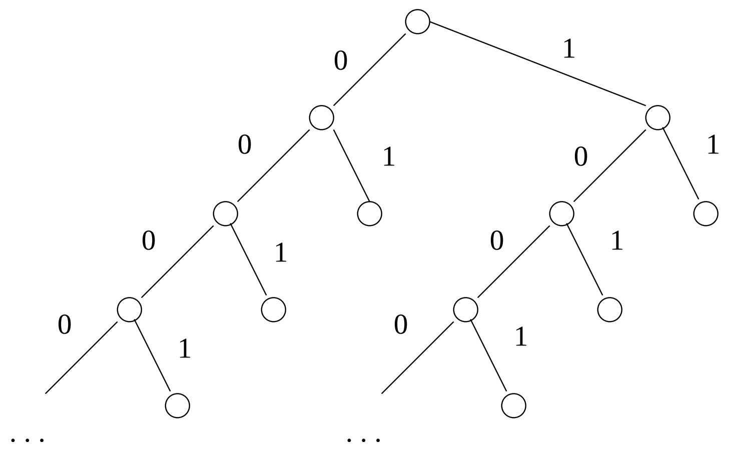
The 2-point attractor of example 1 is an autonomous finite state machine (FSA) as shown in figure 4.



Figur 4: The attractor for the orbit of the alternating sequence $0101010101\dots$ is an autonomous state machine

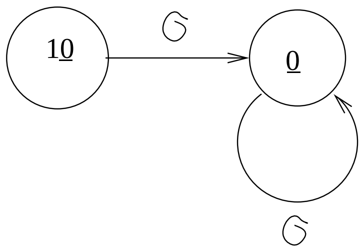
Example

The attractor for the orbit of the sequence $101001000100001\dots$ consists of the two accumulation points $0000\dots$ and $10000\dots$



Figur 5: The attractor for the orbit of the sequence 101001000100001... consists of the two accumulation points 0000... and 10000...

In this case the orbit approaches the attractor, but never reaches it.
The attractor's state graph is shown on figure 6.



Figur 6: State graph for the attractor for the orbit of the sequence 101001000100001... An underlined digit means periodic continuation. The graph has a transient state.

One may ask if an accumulation point can belong to the orbit if the sequence is not ultimately periodic. The answer is yes. The sequence spelled by an infinite path in the suffix tree of a non-periodic sequence is on the attractor. An example is Champernowne's sequence, in which the decimal numbers are listed, first of length 1, then of length 2, etc.

$$12345678901011121314\ldots19202122\ldots30\ldots99100100\ldots$$

The attractor has a countably infinite graph. The suffix tree is the full 10-ary tree. The attractor is dense in Σ: it comes arbitraily close to any point. Its fractal dimension is 1.

Dimension

The attractor can be covered by a set of ϵ-spheres since A is compact. Elements from an k^{-n+1}-sphere are all paths descending from a common node at level n. We can label the spheres by the nodes, hence by words from B. The covering consists of disjoint spheres if no path to a covering nodes passes another node of the cover. No infinite path can avoid passing through a cover node. In graph terminology one could call it a *front* of the tree.

Abstractly, when covering the orbit by spheres, and designating the spheres as states, one knows that σ maps an element of a sphere in a finite number of steps into a member of some sphere. This follows since an infinite tour would come closer than ϵ to an accumulation point. σ may map different elements of a sphere into different spheres. σ maps awz into wz.

If one chooses a cover of nodes at the same level of the tree, one obtains a non-deterministic finite state automaton with $N_n \leq 2^n$ states (Some levels may be empty) which recognizes a set that includes the orbit. It is not obvious that it does not include other paths.

The fractal dimension of the sequence is

$$D_0 = \lim_{n\to\infty} \frac{N_n}{n} \tag{2}$$

The fractal dimension is a special case of the generalized dimension

$$D_q = -\frac{1}{q-1} \lim_{n\to\infty} \frac{\log_2 \sum_{i=1}^{2^n} N_i^q}{n} \tag{3}$$

The entropy D_1 is another special case of the generalized dimension.

Dynamics

The fractal dimension is a property of the geometry of the attractor. The dynamics of the system is reflected by the orbit's behavior close to the attractor. A local linear approximation for an attractor in d dimensions has d eigenvectors. The logarithm of the numerical eigenvalues are called the Lyaponov coefficients. The numerically largest measures the speed by which the orbit is repelled by the attractor (positive Lyapunov coefficient) or attracted (negative Lyapunov coefficient). Assuming a finite dimension d, the Jacobian of the linear approximation can be estimated, and the largest eigenvalue found [3].

For symbolic dynamics, a simple and easily accessible measure would be how fast predictions deteriorate. We want to choose the context which has been most succesfull in the past.

Litteratur

[1] Thomas Bräunl and Birgit Graf. Autonomous mobile robots with on-board vision and local intelligence. In *Proc Second IEEE Workshop on Perception for Mobile Agents*, pages 51–57. Fort Colins CO, 1999.

[2] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner, and G. Vattay. *Classical and Quantum Chaos*. `http://www.nbi.dk/ChaosBook/`, Niels Bohr Institute, 2001.

[3] J-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange atractors. *Review of Modern Physics*, 57(3), 1985.

[4] Peter Johansen. Inductive inference of ultimately periodic squences. *BIT Computer Science Numerical Mathematics*, 28(3):573–580, 1988.

[5] Peter Johansen. On-line string matching with feedback. *Theoretical Computer Science*, pages 53–67, April 1995.

[6] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22:75–81, 1976.

[7] Henrik Hautop Lund. Evolving robot control systems. In *Proceedings of 1NWGA*. Vaasa University, 1995.

[8] R. Andrew McCallum. Hidden state and reinforcement learning with instance-based state identification. *IEEE Transations on Systems, Man and Cybernetics (Special issue on Robot Learning)*, 2000?

[9] Tu and Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH'94*. ACM Computer Graphics, 1994.

[10] E. Ukkonen. Constructing suffix trees on–line in linear time. In J. v. Leeuwen, editor, *Algorithms, Software, Architecture. Information Processing 92*, volume 1, pages 484–492. Elsevier, 1992.