Learning and CTRNNs

Inman Harvey

Evolutionary and Adaptive Systems Group

EASy, Dept. of Informatics

University of Sussex

inmanh@cogs.susx.ac.uk

The Dynamical Systems approach

In contrast to GOFAI:-

The limbs of an animal, a human, or a robot – and their nervous systems, real or artificial – are physical systems with positions and values acting on each other smoothly in ***continuous real time.***

This is so even **without** nervous systems

Walking has a natural dynamics arising from the swing of limbs under gravity.

Passive Dynamic Walking

With upper and lower legs, and un-powered thigh and knee joints, a biped can walk down a slope with no control system

… in simulation …

… or in Reality

Collins,

Cornell.

Adding Nervous Systems

But then in animals, and typically in robots, the **Dynamical System** also includes a (real or artificial) **Nervous System** as part of the whole.

One popular robot/agent style of nervous system is the **CTRNN**

CTRNNs

CTRNNs

CTRNNs (continuous-time recurrent NNs), where for each node (i = 1 to n) in the network the following equation holds:

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{n} w_{ji} \sigma(y_j - \theta_j) + I_i(t)$$

y_i = activation of node i

τ_i = time constant, w_{ji} = weight on connection from node j to node i

$\rho(x)$ = sigmoidal = $(1/1+e^{-x})$

η_i= bias,

I_i = possible sensory input.

Why use CTRNNs?

1. They are **typical** DSs: arbitrary number of variables that vary over time in a lawful manner, depending on the current values of these same variables

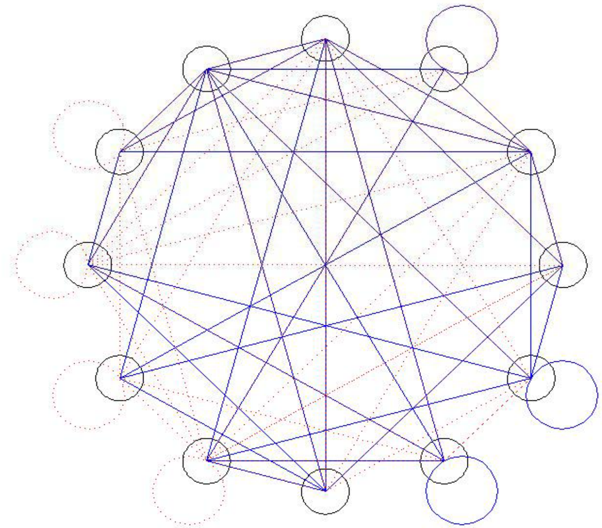
2. Not just typical, but **universal** in the sense that they can approximate arbitrarily closely any smooth DS (Funahashi & Nakamura)

3. Relatively **simple** family of DSs

4. A bit **reminiscent** of brains ….. but **careful!**

The Network view

The Network view



Each equation refers to one node in a network.

Fixed weights on connections

Biases **Sigmoids**

Time parameters = half-life of **leaky integrators**

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{n} w_{ji}\sigma(y_j - \theta_j) + I_i(t)$$

Looks a bit like a normal ANN

… except at least one strange thing – **the weights are fixed!?!?**

Doesn't that mean they **cannot learn??** Because surely learning in ANNs is all to do with weight-changing rules**??**

WRONG !!

Learning Ability ≠ Plastic weights !

The assumption that learning ability necessarily requires plastic weights is widespread and difficult to shake off – eg even Terry Sejnowski (editor-in-chief Neural Computation) is on record as saying just this.

Argument 1

Consider any standard ANN or real NN, with the ability to learn (eg with backprop built in)

This is a (smooth) DS, therefore (Funahashi and Nakamura) it can be approximated arbitrarily closely by some CTRNN – with fixed weights.

QED ! Mathematically open and shut case !!

Argument 2

People have been misled by the term CTR**NN**s, into thinking of them as just another type of neural network.

BUT think of it differently: each **node** is just a variable of the system, if it is modelling/emulating another brain/NN then some of the nodes would represent the weights, other nodes the activations.

It is **unfortunate** that they are pictured as ANNs; think of them as a system of differential equations instead.

Argument 3

What is Learning?

Learning is a **behaviour** of real/artificial/metaphorical organisms.

Actually a **meta-behaviour**, the **changing of behaviours over time under particular circumstances**

Learning to ride a bike

1. On Monday I sit on the bike, push the pedals and fall off
2. Tue, Wed, Thu …lots of practice and pain
3. On Friday I sit on the bike, push the pedals and ride away happily.

Change of behaviour, for the **better**, over **time**, through **experience**

Learning is a ***Behavioural*** term

I suggest that **learning** is best thought of, and limited to being used as, a **behavioural** term.

It has **no implications** at all about what **mechanisms** underlie it (eg plastic or non-plastic weights) – except that the system has to operate over at least 2 different timescales: **eg (a)** riding a bike and **(b)** learning to do so.

This may – or may not – imply different timescales operating within the mechanism.

Timescales

Typically in conventional ANNs (eg backprop) the faster timescale is that of **activations**; the slower timescale is that of **weights**.

In a CTRNN it may be that some nodes have short/fast time parameters (tau), and other have longer/slower ones. A long half-life on a leaky-integrator node implies that its current state is at least partially-dependent on what happened some time ago.

But actually long-term state can also be maintained by only fast nodes.

Examples of CTRNNs learning

A couple of examples of CTRNNs learning, despite weights being fixed:

1. Emulating Hebbian learning (Harvey unpublished – w.i.p.)
2. Study on Origins of learning (Tuci, Quinn, Harvey 2003) building on Yamauchi and Beer 1994.

Emulating Hebbian Learning

Emulating Hebbian Learning

A minimal version: a pre-synaptic node **A** and a post-synaptic node **B**, such that of both **A** and **B** are both activated together, the link between them is strengthened, otherwise weakened.

How can one make sense of this in **behavioural terms**, without any preconceptions as to the mechanism (…we are actually, as a proof of principle, choosing to do it with fixed weights CTRNN) ?

Hebb behaviour

We need a **test** for whether the **A-B link** is **strong** or **weak**.

Eg, input a sine wave of some randomly chosen period to **A**, compare with the resulting output from **B**.

Correlated implies strong link, uncorrelated implies weak.

OK, now we need a training regime such that, if everything is working as we want, this link gets strengthened/weakened appropriately

Training Regime

A CTRNN is designated as a Hebb-mechanism, with 2 nodes designated as **A** and **B**.

1. Randomise activations
2. Run with input sinewaves of **different** periods to **A,B**
3. Then apply sinewave to **A** only, see how correlated **B** is
4. Run with input sinewaves of **same** periods to **A,B**
5. Then apply sinewave to **A** only, see how correlated **B** is

Ideally (3) should be uncorrelated, (5) should be correlated

Results

Results

Evolve a population of CTRNNs with the fitness function being **correln-wanted^2 – correln-unwanted^2**

With just 3 nodes (**A, B** and one spare), get better than random but unimpressive.

With 6 nodes, get respectably good results (fitness > 0.8) – only preliminary work, room for more fine-tuning.

“Experimental evidence that in-principle it is do-able!”

Example 2: Origins of Learning

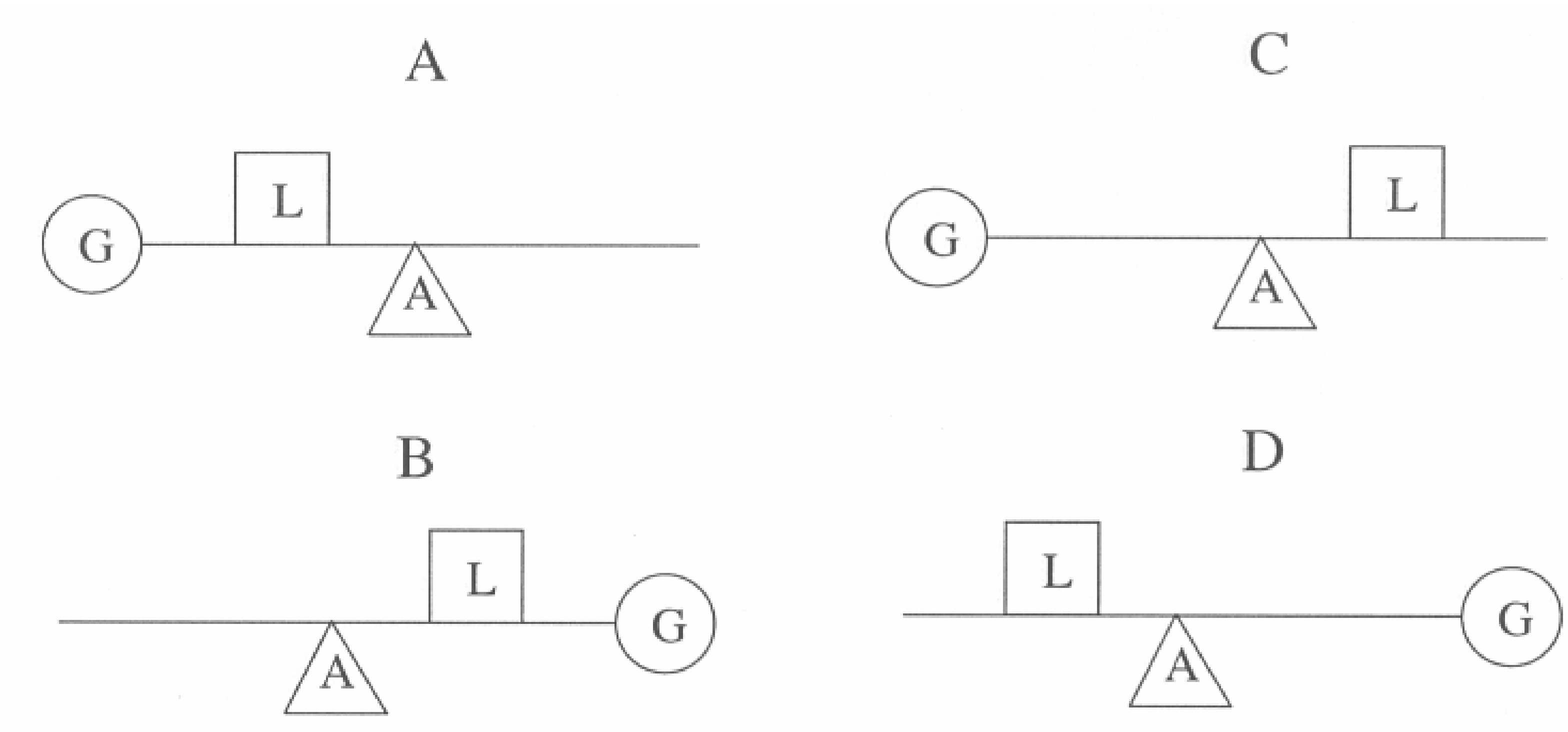
Work by Elio Tuci, with Matt Quinn.

Motivations:-

1. Evolution of learning, from an ecological perspective. The controller of an agent is supplied with **no** explicit learning mechanism, such as any automatic weight-changing algorithm

2. Modular behaviour without specifying any modules

The Model



Extension of work by Yamauchi and Beer (1994)

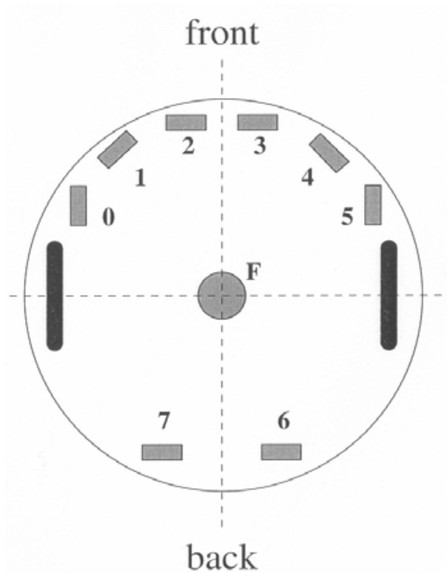
The task

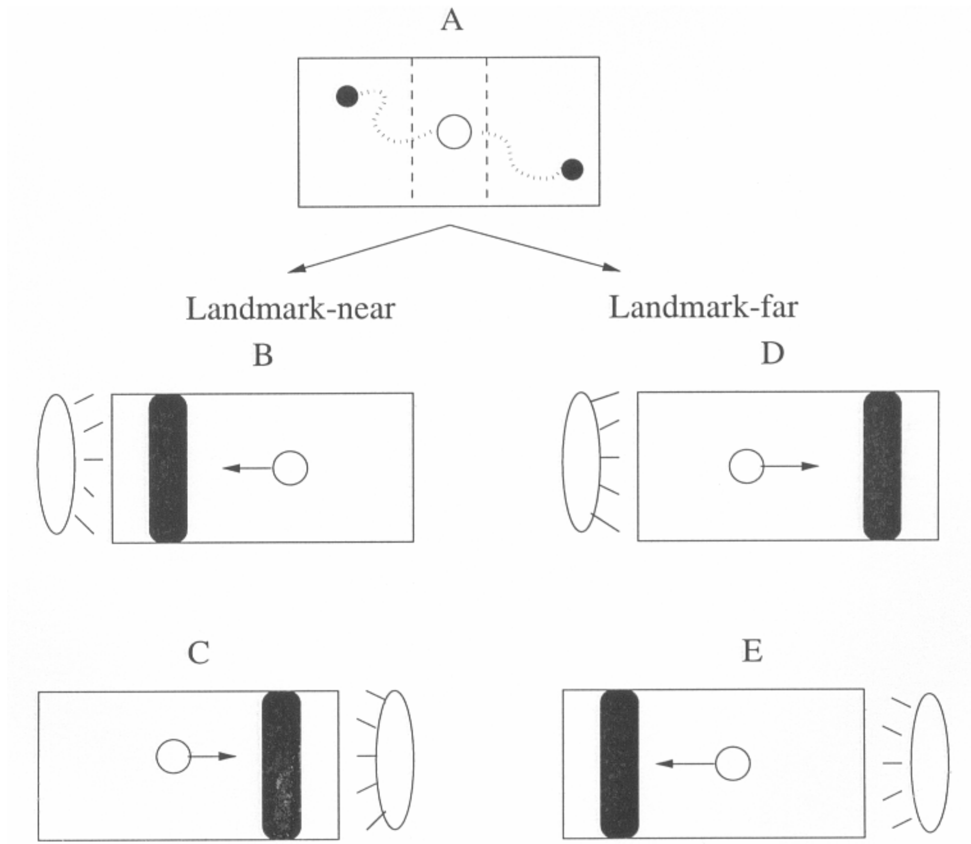
Y & B were trying to evolve the low-level, dynamical properties of control systems for whatever combination of reactive and learning behaviour was effective for the task.

Using CTRNNs – leaky-integrator neurons with fixed connection weights

Unsuccessful until explicit modules were introduced by the experimenters

The changes





A 2-D Khepera-like simulated agent

The problem

Starting from a blank slate, since it was 50/50 whether the light indicated the right or wrong direction, ‘one might as well ignore it’.

So typically a blind search strategy was evolved – and this was a strong local optimum in strategy-search-space.

Having ‘thrown away all vision’ there was no longer any visible cue left for learning with.

Modified fitness function

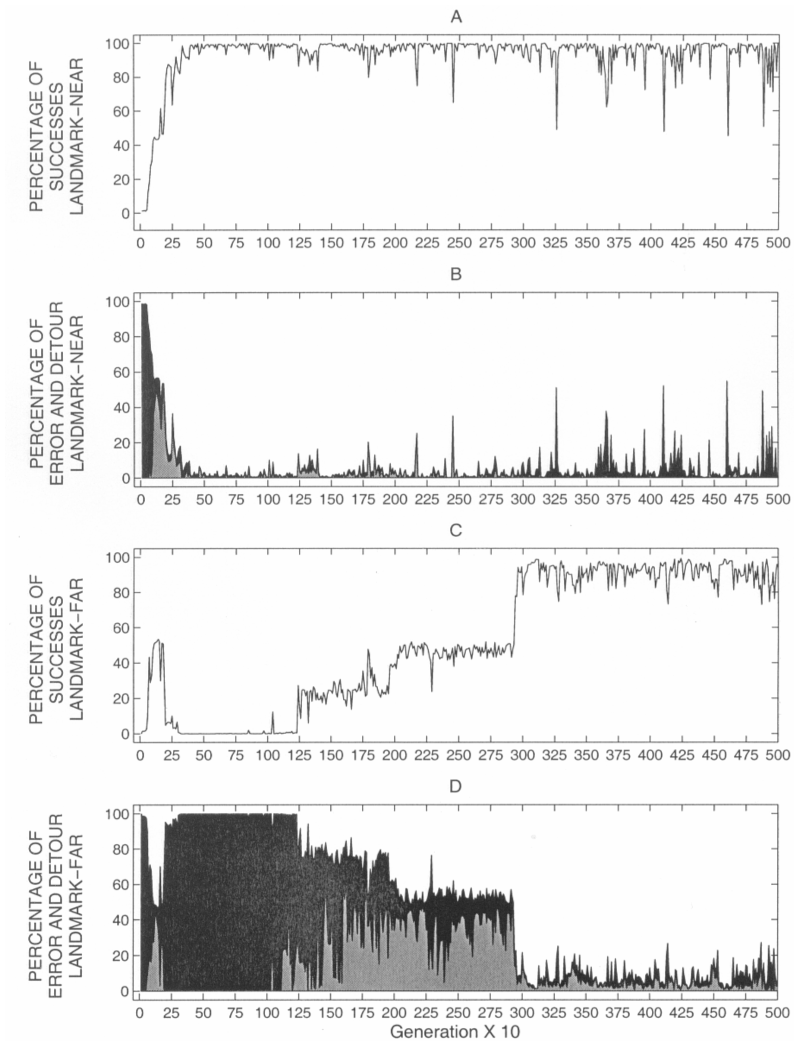
It seems to be essential to modify the evaluation function, so as to give selective pressure for the light to be a **salient** stimulus, *before* it has any value as a learning cue.

E.g. bias the experiments so that the light *is* a cue worth attending to. Here initially trials with light-goes-with-target were made worth 3 times the points of trials with light-opposite-to-target.

Success

Successfully evolved *integrated* CTRNNs with fixed connection weights to achieve this task

No hand-designed modules, no externally introduced reinforcement signal



Summary

From the theoretical arguments, and the two examples, it is perfectly possible to implement learning with a fixed-weight CTRNN.

If anyone tells you that it is impossible, they are foolishly wrong!

But are there **pragmatic** reasons for using plastic weights?

Pragmatic reasons not to use CTRNNs?

Pragmatic reasons not to use CTRNNs?

Maybe it is just **inefficient** to use CTRNNs, maybe Hebbian rules or, more generally, plastic weights make it much easier

It may well be easier to hand-design, does that mean also more evolvable?

Hebbian rules allow built-in multiplication, CTRNNs may have to work hard to do that?

Don't trust your Intuitions!

Don't trust your Intuitions!

To many people it is **obvious** that in principle CTRNNs cannot learn – but they are wrong.

To many people it is **obvious** that it is difficult for CTRNNs to learn – but what is the evidence?

Many have tried and failed – but that may be because the experiments have not been set up properly

Open Research Question

Beer (personal communication) that in at least one example, CTRNNs without plasticity were easier to evolve than those with.

Nice open research area !!!!

THE END