Slide 0

Lecture 5.2

The Hopfield Model For Optimization

Jonathan Shapiro

Department of Computer Science, University of Manchester

February 6, 2003

Slide 1

Hopfield Model Applied To Optimization Problems

References: "Neural computation of decisions in optimization problems", J. Hopfield and D. Tank, Biological Cybernetics, Vol 52, pp 141 – 152, 1985. "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank", G. Wilson and G. Pawley, Biological Cybernetics, Vol 58, pp 63 – 70, 1988. The second paper shows that the method proposed by Hopfield and Tank usually produces solutions which violate the constraints. *Introduction to the Theory of Neural Computation*, J. Hertz, A. Krogh, R. Palmer, Addison Wesley, 1991, Chapter 4.

Slide 2

Content-addressable memory for difficult problems

Hopfield model relaxes to minima of the Liapunov function. Thus, it can be used to solve hard optimization problems.

- Image Restoration: (Geman and Geman, 1984).
  - Make patterns with edges stable states.
  - Starting state a noisy image.
  - Network relaxes to pick out the edges.
- Optimization Problems: (Hopfield and Tank, 1985).
  - e.g. the travelling salesman problem.
  - make near optimal solutions stable states.
  - starting state relaxes into some nearby, good solution.

Slide 3

- Notes:
  - Weights are designed to do a specific task. No learning.
  - Problem solved by network relaxing into good solutions, not search (exactly).
  - Requires casting the problem into finding minima of a Liapunov function *of the same form* as Hopfield's Liapunov function.
  - Can be implemented in hardware using analog VLSI.

Slide 4

Hopfield reasoned,

- Minds do more that make associations; they have to solve *hard* problems.
- Humans solve hard problems in a characteristic way.
- Since the Hopfield network progresses to minima of a Liapunov function under the dynamics, the Hopfield model is solving an optimization problem. Other problems can be solved by the dynamics of the model *if* the problem can be converted into the problem of finding minima of a function *of the same form as* the Hopfield Liapunov function.
- First applied to find solutions of the Travelling Salesman Problem.

Slide 5

Extensions of the Hopfield Model

References: "Neurons with Graded Responses Have Collective Computational Properties like Those of Two-State Neurons", J. Hopfield, Proceedings of the National Academy of Sciences USA, Vol 84, pp 8429 – 8433, 1984. *Introduction to the Theory of Neural Computation*, J. Hertz, A. Krogh, R. Palmer, Addison Wesley, 1991, Chapter 3.3.

Slide 6

Problem: As described above, Hopfield dynamics goes to nearest minimum of the Liapunov function. For solving hard optimization problems, these can be local minima which might have a large value of the function to be minimized.

Solutions:

The continuous model This can be implemented in analogue hardware, and Hopfield was very dedicated to building special purpose hardware for solving optimization tasks using this approach.

Simulated annealing: If the neurons update rule is probabilistic and of a particular form, the problems of local minima can diminished. This is useful for conventional computer implementations of the model.

Slide 7

The Continuous Model

To get a model with continuous neurons, two changes are made to the model described above:

1. The neuron takes a continuous range of values, between 0 and 1, say.
2. The output of a neuron is a continuous function of its input.

$$x_i = g(u_i),$$

where u_i is a quantity which determines the output and g is some function which varies smoothly from 0 to 1. See figure **??**.

Slide 8

3. The neuron cannot respond to the input immediately. The quantity u_i takes some time to ramp up to the total input to the neuron. This is expressed mathematically by letting u_i obey the following equation

$$\tau\frac{du_i}{dt} = -u_i + \sum_j w_{ij}x_j + I_i.$$

This says that the change in time of this variable u_i is proportional to the difference between it and the total input. In time, u_i will be equal to the total input, it will just take some time for it to get there. Here τ is the time constant which determines how long it takes for u_i to go to the value $\sum_j w_{ij}x_i + I_i$. How u changes over time is shown in figure **??**.

Slide 9

The output function for a continuous neuron

Slide 10

The total input to the neuron takes time to reach the weighted sum of its inputs.

Slide 11

Results

- This model behaves like the binary Hopfield model, it ends in similar final states.
- The network changes smoothly and continuously, unlike the binary model which goes through discrete changes at discrete time intervals.
- There is a Liapunov function for this model, so the dynamics always ends in stable states.

Slide 12

Hardware Implementations

The model is important because it can be implemented in hardware.

- An early circuit for a neuron and the network as a whole is shown in figure 1.
- The neuron's input-output function is produced by an amplifier which takes an input voltage V and produces as output $g(V)$. The connection weights are resistors, and the time delay is produce with an RC circuit.
- A network of these neurons can be produced in analog VLSI. Networks of up to a few hundred neurons have been produced.
- **Result:** Massively parallel, special purpose, analog computers using collective computation to solve TSP, linear programming problems and other hard problems were built by Hopfield and his collaborators.
- Problems: High connectivity (like that in brains) is extremely hard to get

Slide 13

in VLSI.

Slide 14

Figure 1: (a) Circuit for implementing an analog neuron. (b) A network of neurons.

Slide 15

Analog Electronic Vision Systems

References: *An Introduction to Neural Electronic Networks*, S. Zornetzer, editor, Academic Press, 1990. *Analog VLSI and Neural Systems*, C. Mead, Addison Wesley, 1989. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", S. Geman and D. Geman, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol 6, pp 721 – 741, 1984.

Slide 16

Motivation

Special purpose analogue parallel computers have been build for vision tasks.

Image restoration — given noisy, blurred, or incomplete information about an image, reproduce the image.

- An ill-posed problem — not enough information to solve.
- Must put in assumptions — smoothness, few edges, etc.

Slide 17

Simple smoothing model

Here is a very simple example.

Given:

1. There may be noise in the input
2. Resolution of eye is finite

Task: Reconstruct image intensities such that

1. image is smooth,
2. intensities are near the data (the inputs)
3. edges may also be present (ignore for now).

Slide 18

- Let d_i be input at retina cell i. Let V_i be hypothesised intensity at the corresponding point in the image.
- Use continuous model with linear neurons $g(u) = u$.
- Liapunov function treating conditions 1) and 2) above

$$L = \frac{1}{2}\sum_i \sum_{j \text{ neighbors } i} (V_i - V_j)^2 + \frac{K}{2}\sum_i (V_i - d_i)^2 . \quad (1)$$

Network of linear elements smoothes images. (See figure)

Slide 19

Dealing with discontinuities(edges)

Natural images contain discontinuities in the form of edges. Different methods have been proposed.

1. Add addition nodes which hypothesize the existence of an edge at location i. Penalize hypotheses which have many edges,

$$\begin{aligned} L &= \frac{1}{2}\sum_i \sum_{j \text{ neighbors } i} (1 - S_i)(V_i - V_j)^2 + \frac{K}{2}\sum_i (V_i - d_i)^2 \\ &\quad + \mu \sum_i S_i . \end{aligned} \quad (2)$$

   where $S_i = 0, 1$ for the absence or presence of an edge.
2. See also Geman and Geman, 1984.
3. Edge detection *emerges* from implementation constraints (Mead).

Slide 20

How edge detection emerges from implementation constraints

- Implemented in a network of linear resistors.
- Resistors not available in all CMOS processes and expensive.
- Can be emulated with an amplifier with similar transfer characteristics (with a few transistors)

  Resistor: Current = voltage/R

  Amplifier: Cannot produce unbounded current; must saturate.

Slide 21

Slide 22

- This saturation gives you segmentation for free.

Slide 23

- You get appropriate response to edges from the physics of the device. (Edge-detection is emerges from the properties of the implementation devices.)
- This constraint must be true for real eyes as well (argues Mead).
- We learn something about biology by building these types of computers to perform similar computational tasks to those done by the biological system.

Slide 24

A Silicon Retina

Described in Mead (1989).

Three types of cells:

1. Photoreceptor — takes log of image intensity.
2. Resistive network — does smoothing and capable of sharp edge segmentation.
3. “Bipolar cells” — measures difference between two cells above.

Network consisted of cells required for 5000 pixels images.

Slide 25

Experimental results

- Edge-detection — bipolar cells do edge detection too.
- Automatic gain and sensitivity control.
- Experimental comparisons with biological retinal cells suggest that this engineered artifact is a good model.
- Shows illusions such as Mach bands.

Slide 26

Conclusions

- Computation is done in the eye; the optic nerve does not have sufficient bandwidth to transmit raw, high fidelity images.
- This computation can be carried out by attractor neural networks; dynamical neural networks whose dynamics leads to fixed points.
- Engineering systems can teach us about biology when the biology system and the engineering system share similar constraints.