Hopfield model

Instructor: Mark Kramer

Today

Describe and simulate the Hopfield model

Original paper

Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

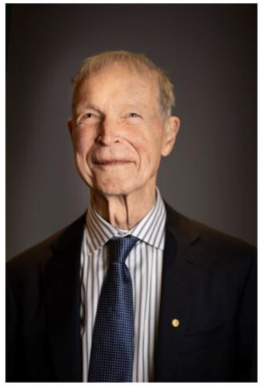
J. J. Hopfield

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

John J. Hopfield

Facts



John J. Hopfield
Nobel Prize in Physics 2024

Born: 15 July 1933, Chicago, IL, USA

Affiliation at the time of the award: Princeton University, Princeton, NJ, USA

Prize motivation: “for foundational discoveries and inventions that enable machine learning with artificial neural networks”

Prize share: 1/2

Work

When we talk about artificial intelligence, we often mean machine learning using artificial neural networks. This technology was originally inspired by the structure of the brain. In an artificial neural network, the brain’s neurons are represented by nodes that have different values. In 1982, John Hopfield invented a network that uses a method for saving and recreating patterns. He found inspiration in physics' models of how many small parts in a system affect the system as a whole. The invention became important in, for example, image analysis.

https://www.nobelprize.org/prizes/physics/2024/hopfield/facts/

Motivation

We design circuits to perform computational tasks

Evolution does not

… but, the brain performs computational tasks.

How?

“a spontaneous collective consequence of having a large number of interacting simple neurons.”

Motivation

A “simple” model of memory

… ignore biological details

Memory should

- work with partial information
- be robust to errors

Complete item

Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci U S A. 1982 Apr;79(8):2554-8.

Partial information

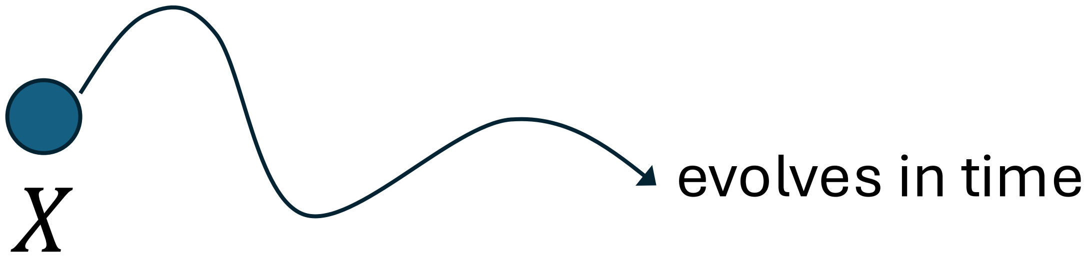
Hopfield Proc Natl Acad Sci. 1982

Partial information with error

Hopfield Proc Natl Acad Sci. 1983

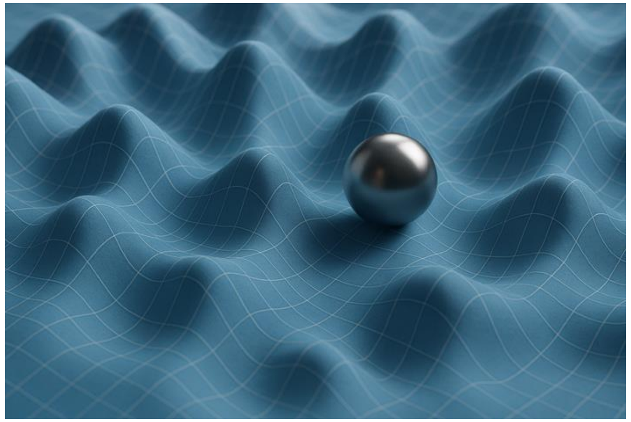
Motivation

Physical systems can act to error correct



A physical system described by general coordinates X

Example: Marble rolling on a surface



1-dimension: position x; velocity v

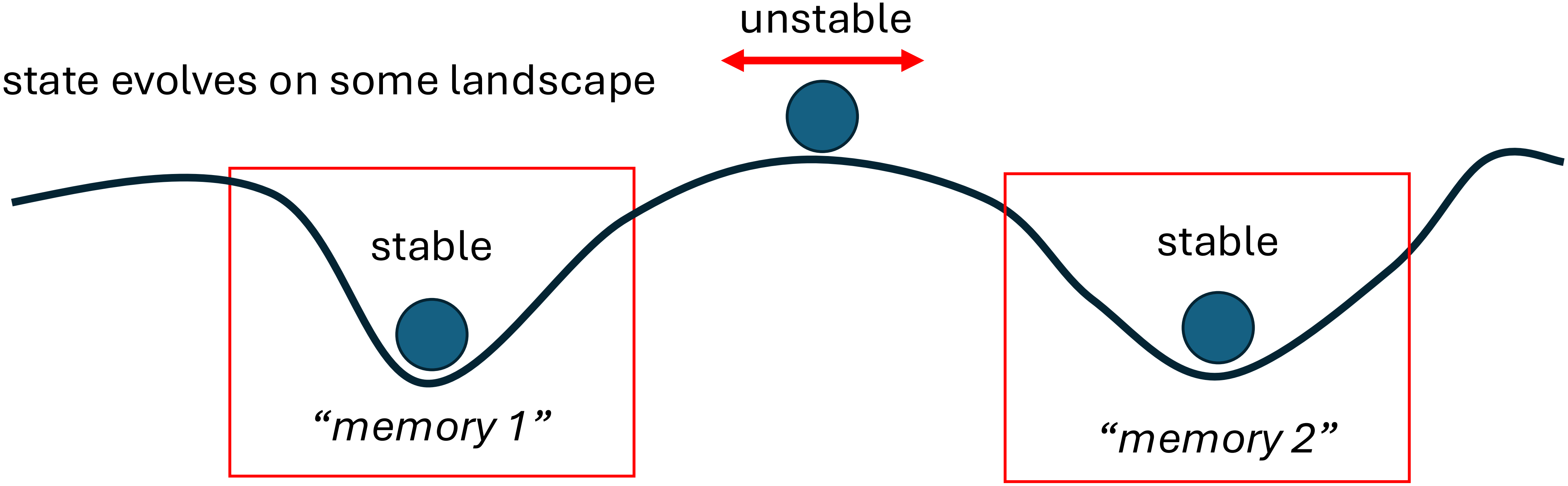
2-dimension: positions x, y; velocities v_x, v_y

Motivation

X is a point in <u>**state space**</u>

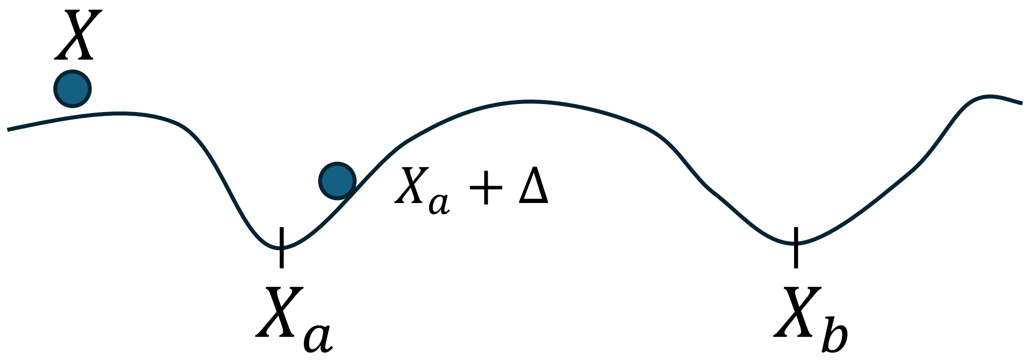
memory → locally stable points

The state evolves on some landscape



near the stable point → flow to it

Motivation



X a point in state space

$X_a, X_b, \ldots$ locally stable points

Start near a locally stable point $X = X_a + \Delta$ partial knowledge of item

As time goes on $X \approx X_a$ system approaches correct memory

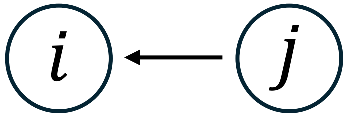
"Any physical system whose dynamics in [state] space is dominated by a substantial number of locally stable states to which it is attracted can therefore be regarded as a general content-addressable memory."

What that motivation, now … the model system

Model system

Consider neuron i

$$V_i = \begin{cases} 0 \text{ if neuron is “not firing”} \\ 1 \text{ if neuron is “firing at maximum rate”} \end{cases}$$



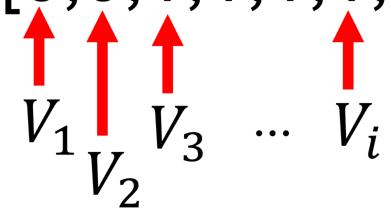
T_{ij} = connection to neuron i from neuron j

= 0 if not connected

<u>Note</u>: reciprocal connections are <u>allowed</u>

Q: What is T_{ij}?

The state of a system with N neurons is a **<u>binary word</u>** [0,0,1,1,1,1,0,1, ...]



Model system

<u>Dynamics</u>: V_i (the state of neuron i) … changes in time

$$V_i \to 1 \text{ if } \sum_{j \neq i} T_{ij} V_j > U_i$$
$$V_i \to 0 \text{ if } \sum_{j \neq i} T_{ij} V_j < U_i$$

$U_i =$ fixed threshold
$= 0$

matrix (T_{ij})

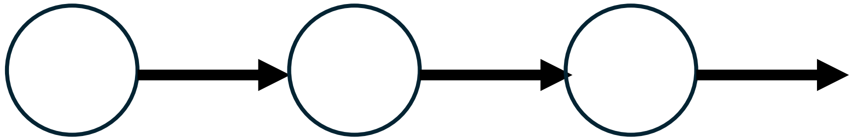
vector (V_j) *(we'll review <u>linear algebra</u> soon …)*

Each neuron i adjusts its state V_i <u>randomly in time</u> (at some rate)

Model system

Compare to the Perceptron

Feedforward



Perceptron	**Hopfield**
feedforward	allows backward coupling
sequential updates	random updates

Python **1. Simulate Hopfield network**

Model system

Define a new variable

$$S = 2V - 1$$

S: [-1, 1]

V: [0, 1]

$$\text{sign}\{x\} = \begin{cases} +1 \text{ if } x > 0 \\ -1 \text{ if } x \leq 0 \end{cases}$$

Define new dynamics

$$\begin{matrix} V_i \to 1 \\ V_i \to 0 \end{matrix} \quad \text{if} \quad \sum_{j \neq i} T_{ij} V_j \begin{matrix} > U_i \\ < U_i \end{matrix}$$

Original Hopfield 1982

$$S_i = \text{sign}\left\{\sum_{j \neq i} T_{ij} S_j\right\}$$

New (simpler) notation

[Kleinfeld 2024]

Q: What is T_{ij}?

Information storage

We want to store a state $S = \xi$

<u>Ex</u>: $\xi = [1,1,-1,1,1,-1,1,1,1,1]$

Storage prescription: **Choose $\boldsymbol{T_{ij}}$**

Q: What is T_{ij}?

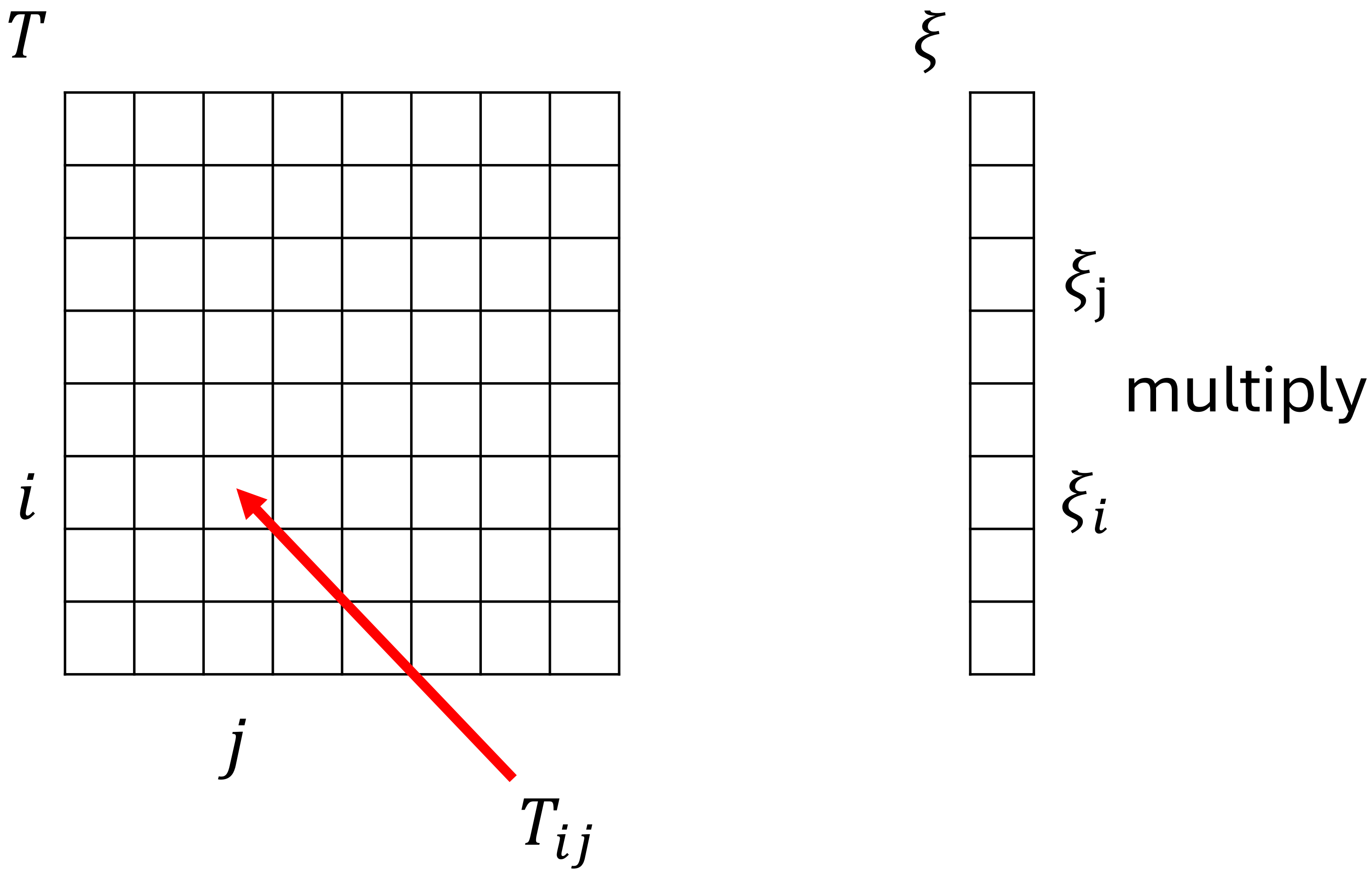
$$T_{ij} = \xi_i \xi_j$$ where $T_{ii} = 0$

<u>**What**</u> is this?

Information storage

What is this?

$$T_{ij} = \xi_i \xi_j$$



<u>Why</u> use this T_{ij}?

Information storage

Case 1: Consider the update rule for the stored stable state ξ

update rule

$$S_i = \text{sign}\left\{\sum_{\mathrm{j \neq i}} T_{ij} S_j\right\}$$

How does ξ evolve due to dynamics?

$$\xi_i = \text{sign}\left\{\sum_{\mathrm{j \neq i}} T_{ij} \xi_j\right\}$$

$$\xi_i = \text{sign}\left\{\sum_{\mathrm{j \neq i}} (\xi_i \xi_j)\, \xi_j\right\} = \text{sign}\left\{\sum_{\mathrm{j \neq i}} \xi_{\mathrm{i}} \xi_{\mathrm{j}}^2\right\}$$

$[(-1)^2, 1^2] = 1$

$$= \text{sign}\left\{\sum_{\mathrm{j \neq i}} \xi_{\mathrm{i}}\right\}$$

Information storage

Case 1: Consider the update rule for the stored stable state ξ

There are N neurons

So

$$\xi_i = \text{sign}\left\{\sum_{j \neq i} \xi_i\right\} = \text{sign}\left\{\xi_i \sum_{j \neq i} 1\right\} = \text{sign}\{\xi_i\,(N-1)\}$$

If $\xi_i = 1, \quad \text{sign}\{\xi_i\,(N-1)\} = 1$

If $\xi_i = -1, \quad \text{sign}\{\xi_i\,(N-1)\} = -1$

So $$\text{sign}\left\{\sum_{j \neq i} T_{ij}\xi_j\right\} = \xi_i$$

→ the update rule does not change the state ξ

It's **stable**

Information storage

Case 2: Consider the update rule for another state S *(not necessarily stable)*

How does S evolve due to dynamics?

$$S_i = \text{sign}\left\{\sum_{j \neq i} T_{ij} S_j\right\}$$

T_{ij} stores the state ξ

$$S_i = \text{sign}\left\{\sum_{j \neq i} \xi_i \boxed{\xi_j S_j}\right\}$$

What is this?

Information storage

<u>Case 2</u>: Consider the update rule for <u>another state</u> S *(not necessarily stable)*

$$S_i = \text{sign}\left\{\sum_{j \neq i} \xi_i \xi_j S_j\right\}$$

What is this?

Assume S <u>mostly matches</u> ξ

What is the product?

... mostly +1 with a few -1

→ sum > 0

$$S_i = \text{sign}\{\ \xi_i\ (\text{positive number})\ \}$$

Information storage

Case 2: Consider the update rule for another state S *(not necessarily stable)*

$$S_i = \text{sign}\{ \xi_i \text{ (positive number) } \} = \text{sign}\{ \xi_i \}$$

depends on the sign of ξ_i

So, $S_i \rightarrow \xi_i$ State S moving towards the stable, stored state ξ

This works and $S_i \rightarrow \xi_i$ as long as $\sum_{j \neq i} \xi_j S_j > 0$

i.e., S_j has at least 50% overlap with ξ_j

Information storage

Case 2: Consider the update rule for another state S

update rule

$$S_i = \text{sign}\left\{\sum_{j \neq i} T_{ij} S_j\right\}$$

How does S evolve due to dynamics?

$$S_i = \text{sign}\left\{\sum_{j \neq i} \xi_i \xi_j S_j\right\}$$

What is this?

Python

2. Pseudoorthogonality

Information storage

Consider

$$S_i = \text{sign}\left\{\sum_{j \neq i} \xi_i \xi_j S_j\right\} = \text{sign}\left\{\xi_i \boxed{\sum_{j \neq i} \xi_j S_j}\right\}$$

What is this?

Conclusions

$$\boxed{\sum_{j \neq i} \xi_j S_j}$$

$\text{mean} \approx 0$ when S & ξ are random (i.e., different patterns)

$\text{mean} > 0$ when S & ξ are similar (i.e., nearby patterns)

pseudoorthogonality

<u>Aside</u>: Linear Algebra (Part 1)