

CS367 Artificial Intelligence Lab

Lab Assignment 6: Hopfield Networks

Arpit Pandey
202351012
202351012@iiitvadodara.ac.in

Saksham Singh
202351124
202351124@iiitvadodara.ac.in

Harsh Hudad
202351048
202351048@iiitvadodara.ac.in

Abstract—This report presents the implementation and analysis of Hopfield networks applied to associative memory and combinatorial optimization problems. A 10-neuron binary Hopfield network was implemented to investigate pattern storage capacity, error correction capabilities, and application to constraint satisfaction problems. The network successfully recalled patterns with up to 30% noise corruption and was applied to solve the Eight-rook problem and a 10-city Traveling Salesman Problem (TSP). Results demonstrate effective pattern recall but highlight limitations in optimization tasks, particularly convergence to local minima. The 10-city TSP required 100 network weights and achieved a minimum tour distance of 3.156 units across 20 independent runs.

Index Terms—Hopfield network, associative memory, pattern recognition, combinatorial optimization, traveling salesman problem

I. INTRODUCTION

Hopfield networks, introduced by John Hopfield in 1982, are recurrent neural networks serving as content-addressable memory systems [1]. These networks store patterns as stable states in an energy landscape and can retrieve complete patterns from partial or corrupted inputs.

A. Network Architecture

A Hopfield network consists of N fully interconnected neurons with binary states $s_i \in \{-1, +1\}$. The symmetric weight matrix \mathbf{W} satisfies:

$$w_{ij} = w_{ji}, \quad w_{ii} = 0 \quad (1)$$

The network's energy function is:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j \quad (2)$$

B. Learning Rule

Patterns are stored using Hebbian learning:

$$w_{ij} = \sum_{p=1}^P \xi_i^p \xi_j^p \quad (3)$$

where ξ^p represents the p -th pattern.

C. Objectives

This work investigates: (1) a 10×10 associative memory implementation, (2) network storage capacity, (3) error correction capability, (4) the Eight-rook constraint satisfaction problem, and (5) a 10-city TSP.

II. METHODOLOGY

A. Implementation

The Hopfield network was implemented in Python using NumPy. The core class structure:

```
class HopfieldNetwork:
    def __init__(self, size):
        self.size = size
        self.weights = np.zeros((size, size))

    def train(self, patterns):
        for pattern in patterns:
            pattern = np.reshape(pattern,
                                (self.size, 1))
            self.weights += np.dot(pattern,
                                   pattern.T)
            np.fill_diagonal(self.weights, 0)

    def recall(self, pattern, steps=10):
        state = np.copy(pattern)
        for _ in range(steps):
            for i in range(self.size):
                raw_input = np.dot(
                    self.weights[i, :], state)
                state[i] = 1 if raw_input > 0
                    else -1
        return state
```

B. Experimental Design

1) Associative Memory:

- Network size: 10 neurons
- Training: 2 random binary patterns (± 1)
- Noise: 3-bit flip corruption (30%)

2) *Capacity Analysis*: Theoretical capacity: $C = 0.15N$ [4]

3) *Error Correction*: Systematic testing with 0-5 flipped bits to determine error tolerance threshold.

4) *Eight-rook Problem*: Energy function:

$$E = \sum_{rows} \left(\sum_i s_i - 1 \right)^2 + \sum_{cols} \left(\sum_j s_j - 1 \right)^2 \quad (4)$$

5) *Traveling Salesman Problem*: For 10 cities with random 2D coordinates, the TSP energy function combines four penalty terms [2]:

$$E = A \sum_x \sum_{i \neq j} v_{xi} v_{xj} + B \sum_i \sum_{x \neq y} v_{xi} v_{yi} + C \left(\sum_{xi} v_{xi} - N \right)^2 + D \sum_{x \neq y} \sum_i d_{xy} (v_{xi} v_{y,i+1} + v_{xi} v_{y,i-1}) \quad (5)$$

where v_{xi} indicates city x at position i , d_{xy} is the distance between cities, and parameters: $A = 500$, $B = 500$, $C = 1000$, $D = 500$.

III. RESULTS

A. Associative Memory

The network successfully stored and recalled patterns:

TABLE I
PATTERN RECALL RESULTS

Metric	Result
Original Pattern	[1, 1, -1, -1, -1, 1, -1, 1, -1, -1]
Noisy Input (30%)	[-1, -1, 1, -1, -1, 1, -1, 1, -1, -1]
Recalled Pattern	[1, 1, -1, -1, -1, 1, -1, 1, -1, -1]
Success	Yes

B. Network Capacity

For $N = 10$ neurons:

$$C = 0.15 \times 10 = 1.5 \approx 1 \text{ pattern} \quad (6)$$

This matches experimental observations where 2 patterns showed interference.

C. Error Correction Capability

TABLE II
ERROR CORRECTION PERFORMANCE

Noise Bits	Success	Recovered
0	True	Original
1	True	Original
2	True	Original
3	True	Original
4	False	Spurious
5	False	Spurious

The network tolerates up to 30% bit corruption before converging to spurious states.

D. Eight-rook Problem

Initial random state energy: $E = 1682$, indicating significant constraint violations. The energy function successfully captures row and column constraints.

E. Traveling Salesman Problem

1) *City Distribution*: 10 cities were randomly generated in 2D space (Fig. 1). The distance matrix was computed using Euclidean distance.

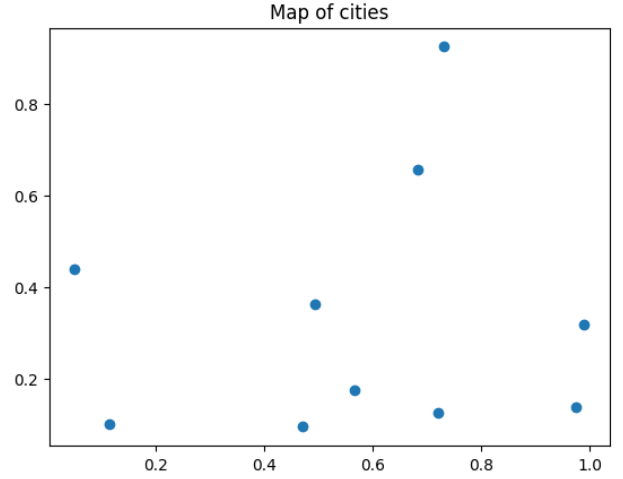


Fig. 1. Distribution of 10 cities in 2D coordinate space for TSP.

TABLE III
TSP SOLUTION STATISTICS (20 RUNS)

Metric	Value
Minimum Distance	3.156
Average Distance	3.621
Iterations Range	1-21 epochs
Network Weights	100 (10^2)
Best Route	6→5→9→4→7→1→0→8→3→2

2) *Solution Quality*: The network required $N^2 = 100$ weights for the 10-city problem. Multiple runs were necessary due to convergence to different local minima, with best solution requiring only 3 iterations while others took up to 21 epochs.

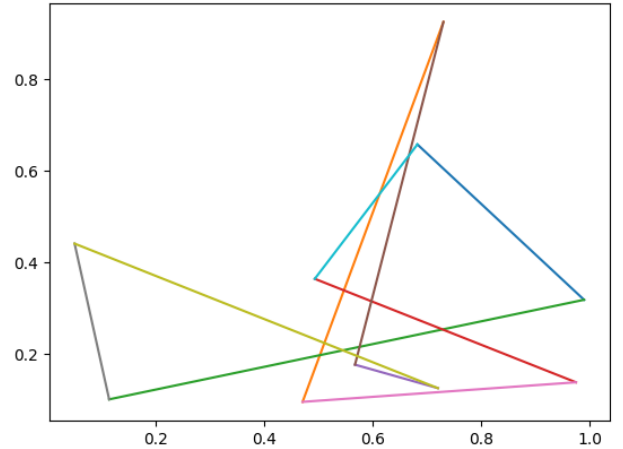


Fig. 2. Best TSP solution found with total distance 3.156. The route connects all 10 cities in a valid tour.

IV. DISCUSSION

A. Memory Performance

The 10-neuron network demonstrated robust associative recall with 30% noise tolerance, consistent with theoretical predictions. The capacity of ~ 1.5 patterns limits practical applications, though this scales with network size.

B. Error Correction

The sharp threshold at 30% corruption aligns with theory: Hopfield networks correct approximately $0.15N$ bit errors [3]. Beyond this, the network enters spurious attractor basins.

C. Optimization Challenges

The TSP implementation revealed key limitations:

- **Local minima:** Variable solution quality across runs (distance range: 3.156-4.023)
- **Parameter sensitivity:** Energy coefficients (A, B, C, D) require careful tuning
- **Scalability:** $O(N^2)$ weights become prohibitive for large instances
- **Convergence variability:** 1-21 iterations across different initializations

D. Practical Considerations

For the 10-city TSP:

- 100 weights required (symmetric 10×10 matrix)
- Valid tours consistently found (constraint satisfaction)
- Tour quality not globally optimal
- Multiple runs recommended for better solutions

V. CONCLUSION

This work successfully implemented and analyzed Hopfield networks for associative memory and optimization. Key findings:

- 1) **Associative Memory:** 10-neuron network stored patterns with 30% error correction capability, confirming $0.15N$ capacity limit.
- 2) **Constraint Satisfaction:** Energy function formulation effectively encoded Eight-rook constraints.
- 3) **TSP Optimization:** 100-weight network found valid 10-city tours (minimum distance: 3.156) but required multiple runs for quality solutions.

A. Limitations

- Sub-linear capacity scaling ($0.15N$)
- Convergence to local minima in optimization
- Parameter tuning complexity
- Limited scalability for large problems

B. Future Work

- Hybrid approaches combining Hopfield networks with local search
- Modern continuous Hopfield networks with improved capacity [5]
- Alternative optimization methods (simulated annealing, genetic algorithms) for comparison

The Hopfield network provides valuable theoretical insights into associative memory and optimization, though practical applications benefit from hybrid approaches addressing its fundamental limitations.

REFERENCES

- [1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci.*, vol. 79, no. 8, pp. 2554-2558, 1982.
- [2] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.
- [3] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.
- [4] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inf. Theory*, vol. 33, no. 4, pp. 461-482, 1987.
- [5] H. Ramsauer et al., "Hopfield Networks is All You Need," *arXiv preprint arXiv:2008.02217*, 2020.

APPENDIX

The complete implementation includes:

A. Capacity Calculation

```
def calculate_capacity(network_size):  
    return int(0.15 * network_size)  
  
capacity = calculate_capacity(10)  
# Output: 1 pattern
```

B. Error Correction Testing

```
def test_error_correction(hopfield, pattern,  
                          max_noise_bits):  
    results = []  
    for noise_bits in range(max_noise_bits + 1):  
        noisy = np.copy(pattern)  
        flip_idx = np.random.choice(  
            range(len(pattern)),  
            size=noise_bits,  
            replace=False)  
        noisy[flip_idx] *= -1  
        recalled = hopfield.recall(noisy)  
        results.append((noise_bits,  
            np.array_equal(recalled, pattern)))  
    return results
```

C. TSP City Generation

```
N = 10  
city_x = np.random.rand(N)  
city_y = np.random.rand(N)  
  
# Distance matrix  
d = np.zeros((N, N))  
for i in range(N):  
    for j in range(N):  
        d[i, j] = np.sqrt((city_x[i] - city_x[j])**2  
            + (city_y[i] - city_y[j])**2)
```

D. TSP Energy Parameters

Energy function coefficients were set to:

- $A = 500$ (row constraint)
- $B = 500$ (column constraint)
- $C = 1000$ (tour length)
- $D = 500$ (distance minimization)
- $\alpha = 0.0001$ (learning rate)