Improved Robustness and Hyperparameter Selection in the Modern Hopfield Network

Hayden McAlister

School of Computing, University of Otago

Supervisors: Anthony Robins, Lech Syzmanski

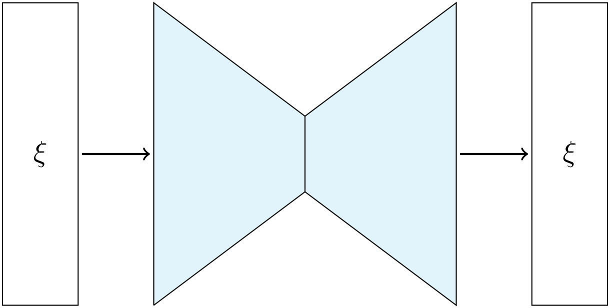
Overview

Autoassociative Memories

- Learn to associate a state with itself.
- Relax probe towards a learned state.

Autoassociative Memories

- Learn to associate a state with itself.
- Relax probe towards a learned state.



Autoassociative Memories

- Learn to associate a state with itself.
- Relax probe towards a learned state.

Autoassociative Memories

- Learn to associate a state with itself.
- Relax probe towards a learned state.

Classical Hopfield Network

- Association by Hebbian learning.
  - Biological inspiration.
  - Easy to analyze.
- Relax by matrix multiplication.
  - Mean field approximation.
  - Nonlinearity keeps states in bipolar domain.
  - Energy guaranteed to achieve a minima (under sensible conditions).

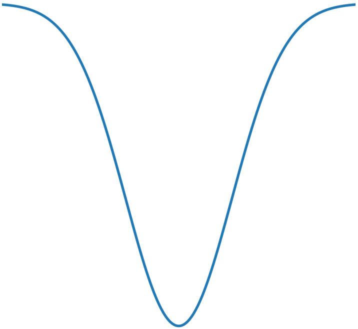
$$W = \sum_k \xi_k \otimes \xi_k \tag{1}$$

$$\xi_{t+1} = \text{Sign}\left(W \cdot \xi_t\right) \tag{2}$$

$$E\left(\xi\right) = -\frac{1}{2}\xi^T W \xi \tag{3}$$

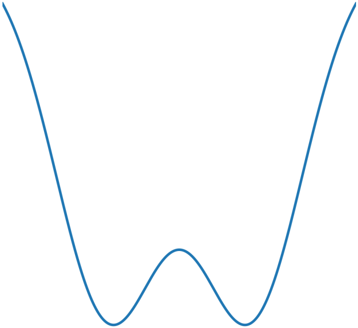
Modern Hopfield Network

- Classical energy wells are too shallow.



Modern Hopfield Network

- Classical energy wells are too shallow.



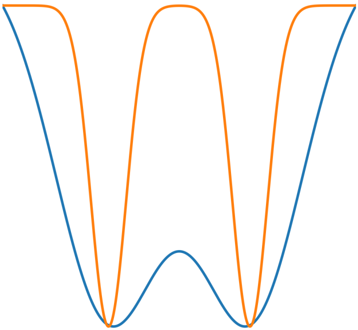
Modern Hopfield Network

- Key trick: Replace quadratic energy with general polynomial.
  - Heck, anything with a vaguely polynomial shape.

$$f_n(x) = x^n$$

$$f_n(x) = \begin{cases} x^n \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

$$f_n(x) = \begin{cases} x^n \text{ if } x \geq 0 \\ -\epsilon x \text{ if } x < 0 \end{cases}$$



Modern Hopfield Network

n – The Interaction Vertex

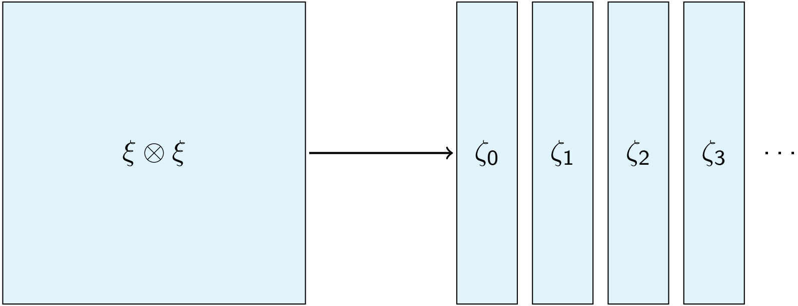
- Controls the range of influence that memories have.
- However, also radically alters the network architecture.

Modern Hopfield Network

n – The Interaction Vertex

- Controls the range of influence that memories have.
- However, also radically alters the network architecture.

Memory matrix replaced by list of memory states – vectors of same dimension as data.



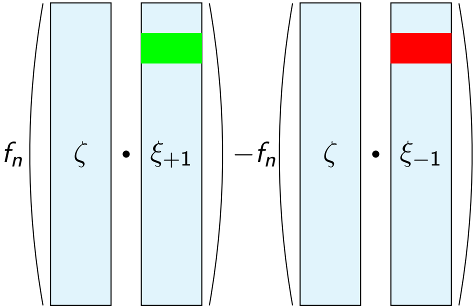
Modern Hopfield Network

n – The Interaction Vertex

- Controls the range of influence that memories have.
- However, also radically alters the network architecture.

Relaxation no longer uses mean field – now a contrastive difference.

- Negative energy no longer means "stable" – the energy *difference* between a neuron clamped on and off indicates stability.



Modern Hopfield Network

n – The Interaction Vertex

- Controls the range of influence that memories have.
- However, also radically alters the network architecture.

Learning no longer supports Hebbian – now requires gradient descent.

$$W = \sum_k \xi_k \otimes \xi_k$$

$$\text{Loss}\left(\xi\right) = \tanh\left[\beta \sum_\mu \left(f_n\left(\zeta_{\mu,i} + \sum_{j\neq i} \zeta_{\mu,j}\xi_j\right) - f_n\left(-\zeta_{\mu,i} + \sum_{j\neq i} \zeta_{\mu,j}\xi_j\right)\right)\right]$$

Properties – Network Capacity

Larger network capacities with higher interaction vertices:

$$K_{\max} = \frac{1}{2\,(2n-3)!!}\frac{N^{n-1}}{\ln(N)} \tag{4}$$

Notably, super-linear for $n > 2$.

Properties – Training Times

Faster training times with higher interaction vertices:

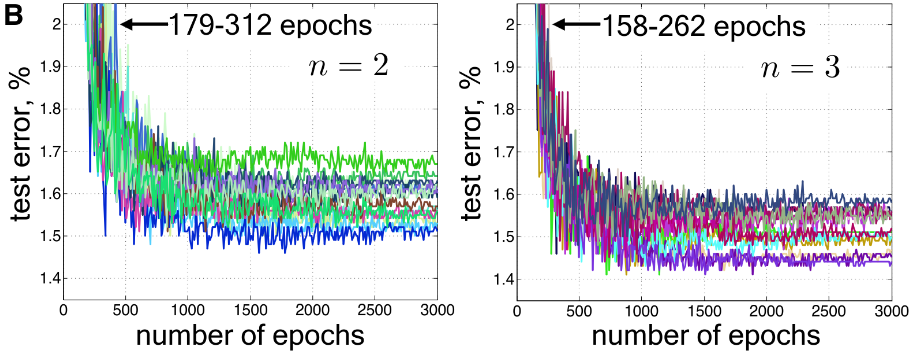


Figure: Krotov and Hopfield 2016, Figure 01

Properties – Feature to Prototype Transition

Low interaction vertices result in memories that look like features, while higher interaction vertices result in memories that look like prototypes:

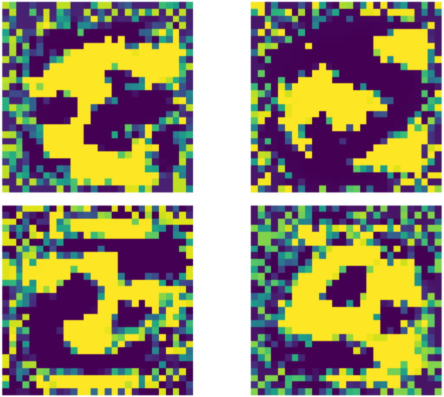


Figure: Feature-like Memories, $n = 2$

Properties – Feature to Prototype Transition

Low interaction vertices result in memories that look like features, while higher interaction vertices result in memories that look like prototypes:

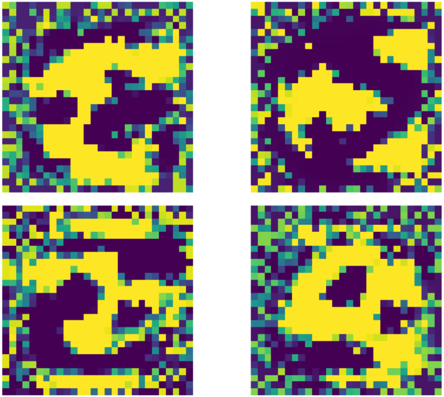


Figure: Feature-like Memories, $n = 2$

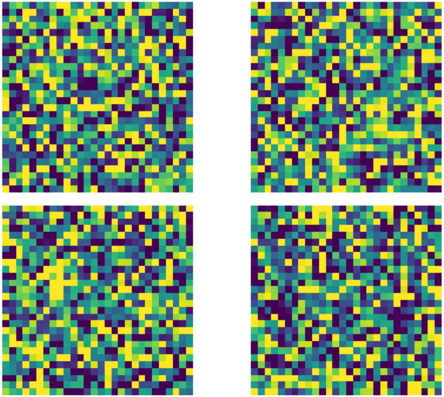


Figure: Prototype-like Memories, $n = 20$

Why is our implementation broken?

- Other implementations online...

Why is our implementation broken?

- Other implementations online. . .
    - Most use a feed-forward architecture that isn't as general.

Why is our implementation broken?

- Other implementations online...
  - Most use a feed-forward architecture that isn't as general.
  - An example of the autoassociative memory exists, and works!

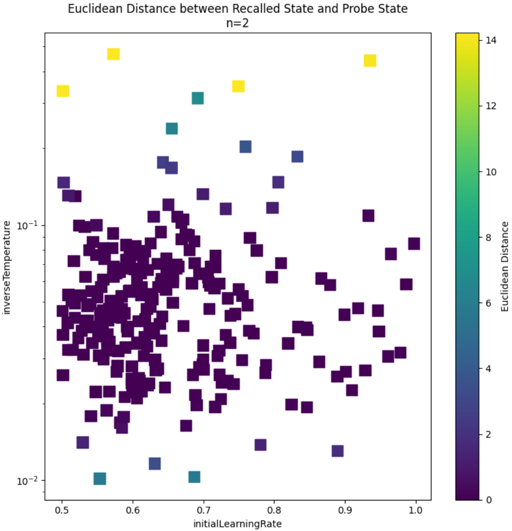
Why is our implementation broken?

- Other implementations online...
  - Most use a feed-forward architecture that isn't as general.
  - An example of the autoassociative memory exists, and works!
  - When translated line by line to PyTorch, still broken...
- Hyperparameters are numerous and "magic".

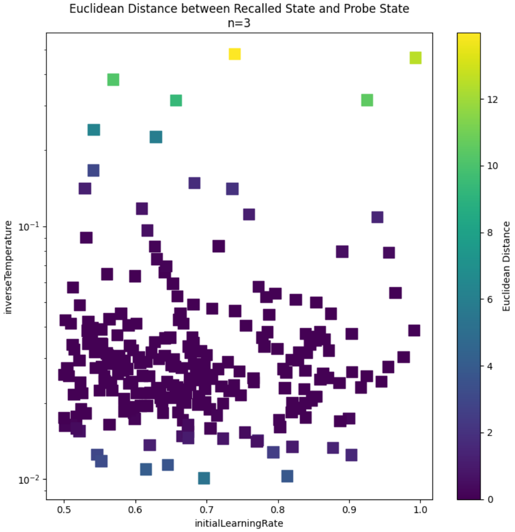
Hyperparameter Search

Training our network on autoassociative tasks of dimension 100.
We measure the Euclidean distance between learned state and recalled state. Lower is better, as this corresponds to a well recalled memory.

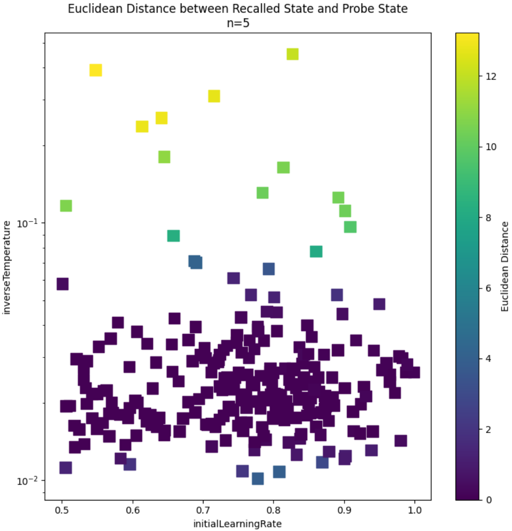
Hyperparameter Search: $n = 2$



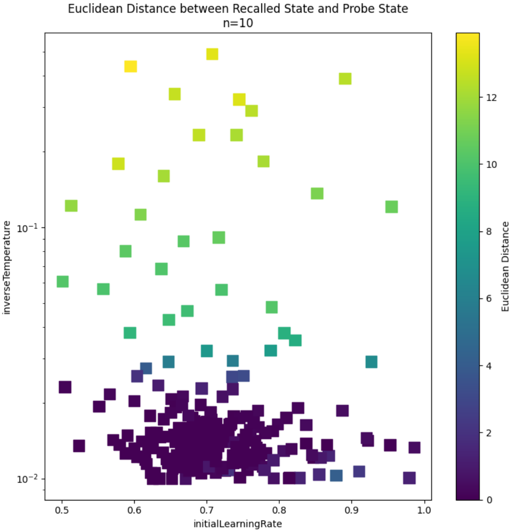
Hyperparameter Search: $n = 3$



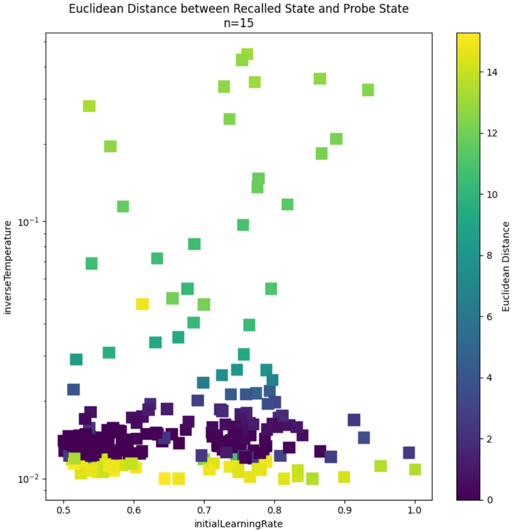
Hyperparameter Search: $n = 5$



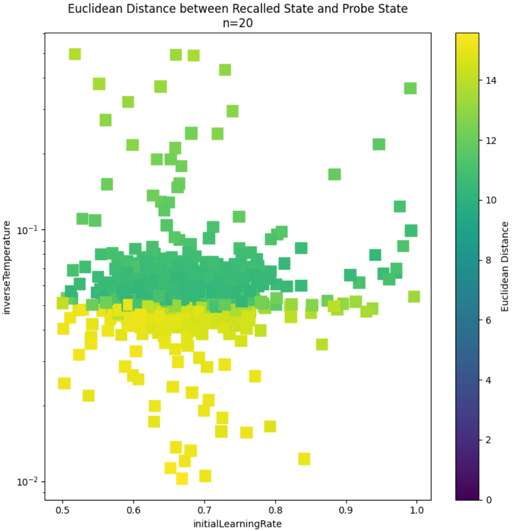
Hyperparameter Search: $n = 10$



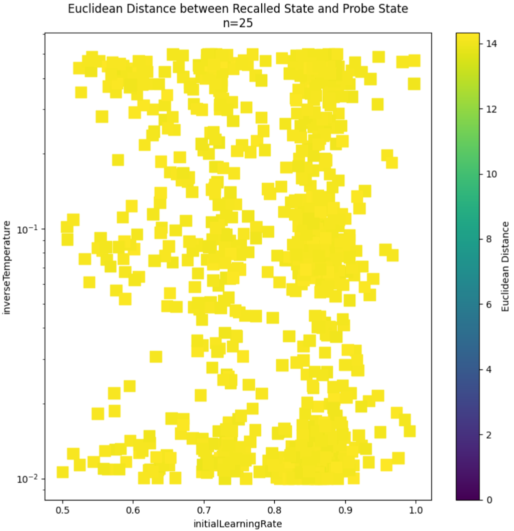
Hyperparameter Search: $n = 15$



Hyperparameter Search: $n = 20$



Hyperparameter Search: $n = 25$



Network Dynamics Step by Step

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

1. Calculate similarities $\zeta\cdot\xi_{+1},\ \zeta\cdot\xi_{-1}$
2. Pass similarities through interaction function f_n
3. Sum the result over all memories $\sum_\mu$
4. Multiply by a scaling factor β
5. Pass through activation function (e.g. Sign or tanh)

Network Dynamics Step by Step

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

1. Calculate similarities $\zeta\cdot\xi_{+1},\ \zeta\cdot\xi_{-1}$
2. **Pass similarities through interaction function f_n**
3. Sum the result over all memories $\sum_{\mu}$
4. Multiply by a scaling factor β
5. Pass through activation function (e.g. Sign or tanh)

$$f_n(x)=x^n$$

Network Dynamics Step by Step

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

1. **Calculate similarities** $\zeta\cdot\xi_{+1},\ \zeta\cdot\xi_{-1}$
2. **Pass similarities through interaction function** f_n
3. Sum the result over all memories $\sum_\mu$
4. Multiply by a scaling factor β
5. Pass through activation function (e.g. Sign or tanh)

$$\zeta,\xi\in[-1,1]^N \implies \zeta\cdot\xi\in[-N,N]$$

Network Dynamics Step by Step

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

1. **Calculate similarities** $\zeta\cdot\xi_{+1},\ \zeta\cdot\xi_{-1}$
2. **Pass similarities through interaction function** f_n
3. Sum the result over all memories $\sum_{\mu}$
4. Multiply by a scaling factor β
5. Pass through activation function (e.g. Sign or tanh)

$$\zeta,\xi\in[-1,1]^N \implies \zeta\cdot\xi\in[-N,N]$$
$$f_n(\zeta\cdot\xi)=(\zeta\cdot\xi)^n$$

Network Dynamics Step by Step

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

1. **Calculate similarities** $\zeta\cdot\xi_{+1},\ \zeta\cdot\xi_{-1}$
2. **Pass similarities through interaction function** f_n
3. Sum the result over all memories $\sum_{\mu}$
4. Multiply by a scaling factor β
5. Pass through activation function (e.g. Sign or tanh)

$$\zeta,\xi\in[-1,1]^N \implies \zeta\cdot\xi\in[-N,N]$$
$$f_n(\zeta\cdot\xi)=(\zeta\cdot\xi)^n$$
$$=N^n$$

How large is too large?

Network parameters	Interaction function value
$N = 100, n = 2$	10^4
$N = 100, n = 5$	10^{10}
$N = 100, n = 10$	$10^{18.89}$
$N = 100, n = 20$	10^{40}

How large is too large?

Network parameters	Interaction function value
$N = 100, n = 2$	10^4
$N = 100, n = 5$	10^{10}
$N = 100, n = 10$	$10^{18.89}$
$N = 100, n = 20$	10^{40}

Maximum value of a float32 is $\approx 3.4 \cdot 10^{38}$. Default data type of PyTorch.

How large is too large?

Network parameters	Interaction function value
$N = 100, n = 2$	10^4
$N = 100, n = 5$	10^{10}
$N = 100, n = 10$	$10^{18.89}$
$N = 100, n = 20$	10^{40}

Maximum value of a float32 is $\approx 3.4 \cdot 10^{38}$. Default data type of PyTorch.
Maximum value of a float64 is $\approx 1.8 \cdot 10^{304}$. Default data type of NumPy.

Stabilizing the Network

- The optimal hyperparameter region shrinks as n grows.

Stabilizing the Network

- The optimal hyperparameter region shrinks as n grows.
- The optimal region also shifts considerably as n grows.

Stabilizing the Network

- The optimal hyperparameter region shrinks as n grows.
- The optimal region also shifts considerably as n grows.
- When n reaches some critical threshold, the optimal region is disrupted and the network does not learn at all.

Stabilizing the Network

- The optimal hyperparameter region shrinks as n grows.
- The optimal region also shifts considerably as n grows.
- When n reaches some critical threshold, the optimal region is disrupted and the network does not learn at all.
  - Large n makes prototype memories.
  - Prototype memories have high similarities.
  - High similarities and large n causes overflow.

Fixing the Problem

$$\tanh\left[\beta\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

Fixing the Problem

$$\tanh\left[\boldsymbol{\beta}\sum_{\mu}\left(f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

Fixing the Problem

$$\tanh\left[\sum_{\mu}\left(\beta f_n\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)-\beta f_n\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right]$$

Fixing the Problem

$$\tanh\left[\sum_{\mu}\left(f_n\left(\boldsymbol{\beta}\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)-f_n\left(\boldsymbol{\beta}\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right)\right]$$

If f_n is homogenous: $f(\alpha x)=\alpha^k f(x)$.
Weaker than linear – includes Polynomial and Rectified Polynomial.

Fixing the Problem

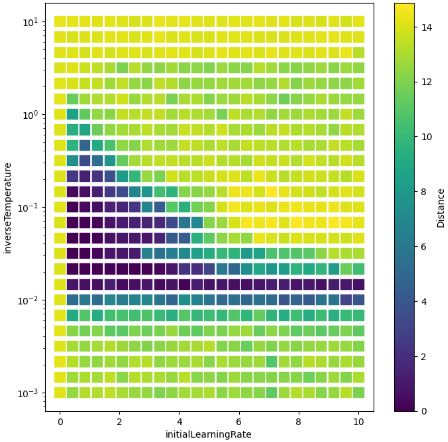
$$\tanh\left[\sum_{\mu}\left(f_n\left(\frac{\beta}{N}\left(\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)-f_n\left(\frac{\beta}{N}\left(-\zeta_{\mu,i}+\sum_{j\neq i}\zeta_{\mu,j}\xi_j\right)\right)\right)\right]$$

If f_n is homogenous: $f(\alpha x)=\alpha^k f(x)$.

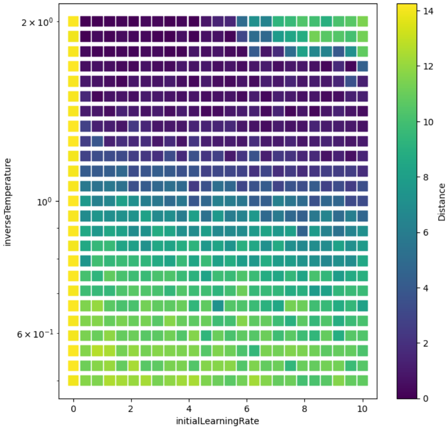
Weaker than linear – includes Polynomial and Rectified Polynomial.

Results of Modifications – Autoassociative Memory

$n = 2$



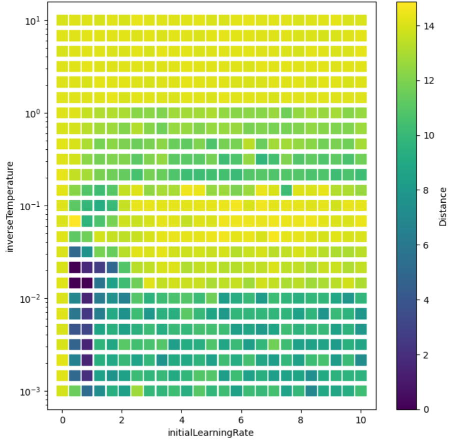
Original



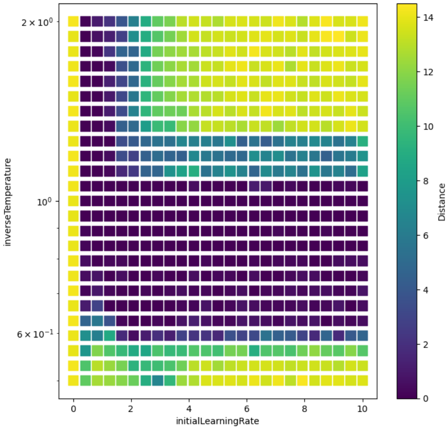
Modified

Results of Modifications – Autoassociative Memory

$$n = 10$$

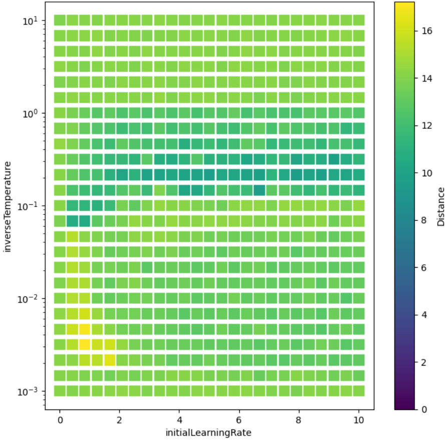


Original

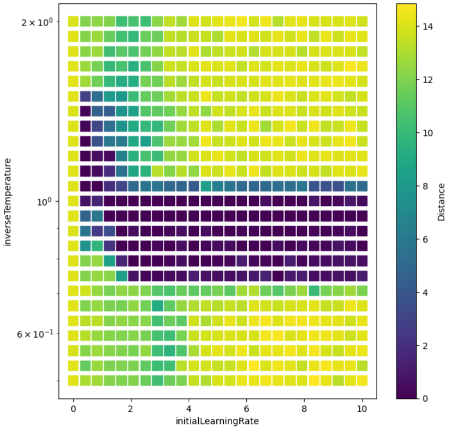


Modified

$n = 20$



Original



Modified

Results of Modifications – MNIST Classification

$n = 2$

Original

Modified

Results of Modifications – MNIST Classification

$n = 10$

Original

Modified

Properties – Feature to Prototype Transition

Low interaction vertices result in memories that look like features, while higher interaction vertices result in memories that look like prototypes:

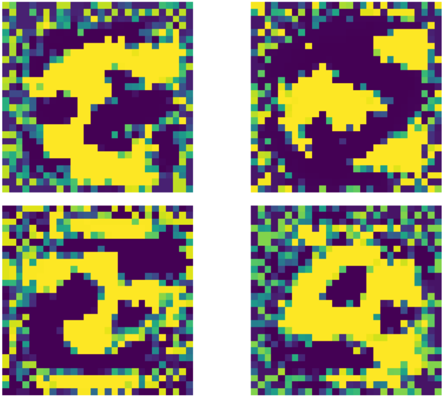


Figure: Feature-like Memories, $n = 2$

Properties – Feature to Prototype Transition

Low interaction vertices result in memories that look like features, while higher interaction vertices result in memories that look like prototypes:

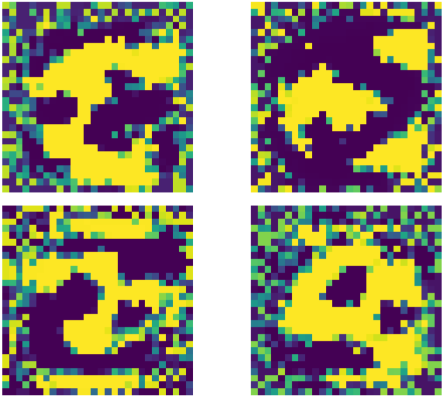


Figure: Feature-like Memories, $n = 2$

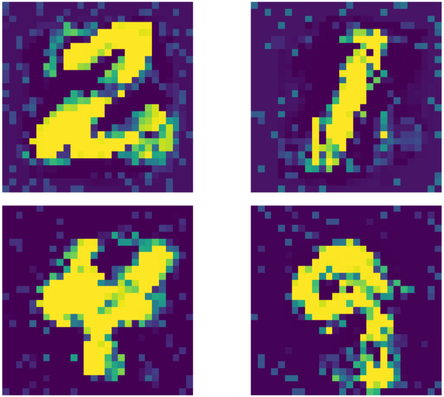


Figure: Prototype-like Memories, $n = 20$

Conclusion

- Modern Hopfield Network generalizes Classical Hopfield Network.
- Network capacity increases with the interaction vertex, even super-linearly.
- The original network (Krotov and Hopfield, 2016) has very unstable behavior for larger interaction vertices.
- The original network also has wildly shifting optimal hyperparameter regions.
- Our modifications solve both the instability and shifting hyperparameter regions at no additional cost.