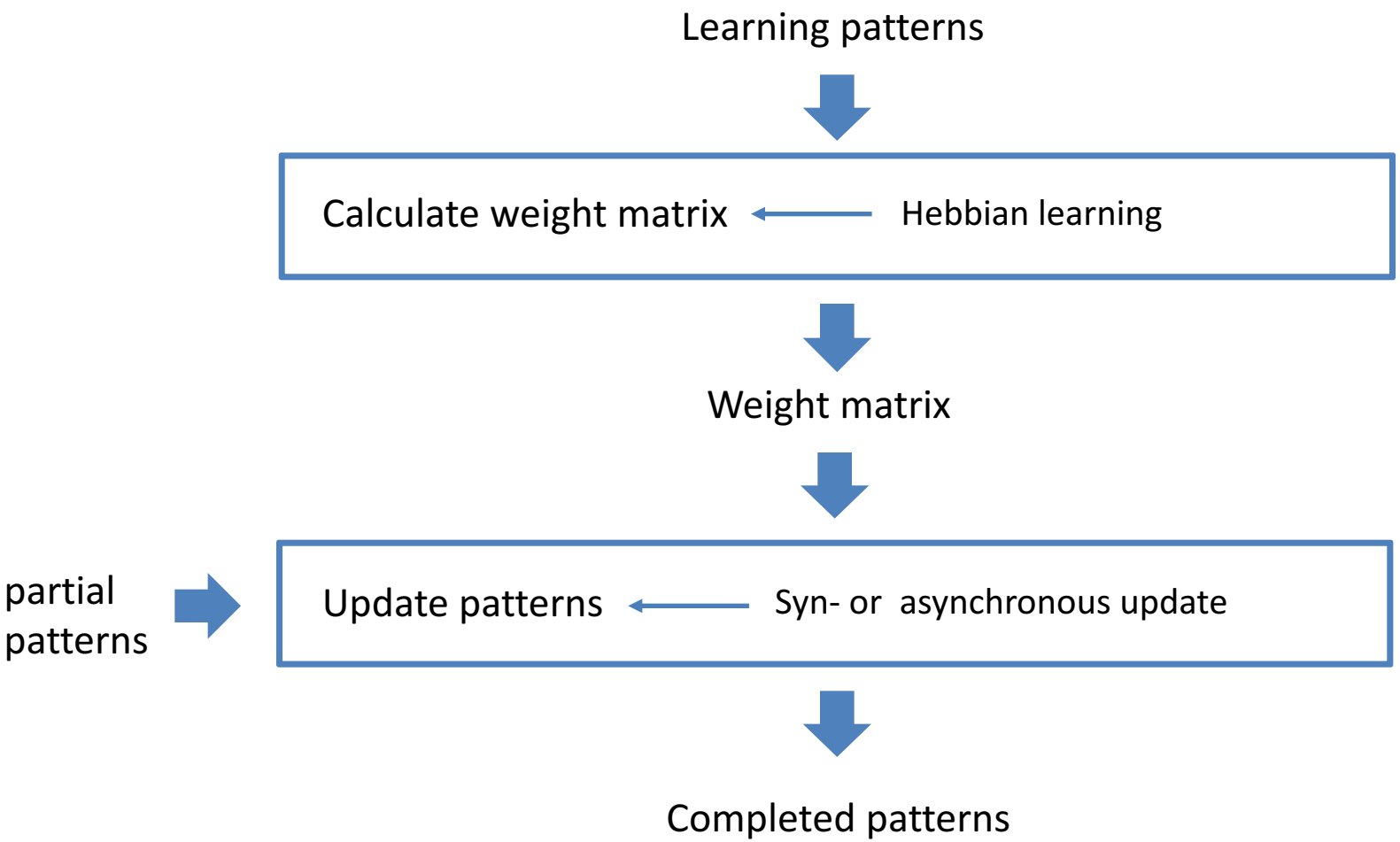
ANN - Lab 4
Hopfield Networks

Tianxiao Zhao
Suping Shi

Feb 22nd, 2017

How it works...


Learning patterns
Calculate weight matrix
Hebbian learning
Weight matrix
partial patterns
Update patterns
Syn- or asynchronous update
Completed patterns

Attractors and convergence

```
x1 = vm([0 0 1 0 1 0 0 1]);
x2 = vm([0 0 0 0 0 1 0 0]);
x3 = vm([0 1 1 0 0 1 0 1]);
```

```
iteration times:

a =

     3

patterns can converge towards stored patterns
```

Change the starting pattern even more dissimilar (more than half are wrong)

```
xd1 = vm([0 1 1 1 1 1 1 1]);
xd2 = vm([1 1 1 0 0 0 0 0]);
xd3 = vm([1 0 0 0 1 1 0 0]);
```

```
iteration times:

a =

     2

patterns can not converge towards stored patterns
```

Observation:

- Output pattern converges but not to the stored patterns
- Not any input can converge to stored patterns, only that have some similarities.

Attractors and convergence

Find all attractors

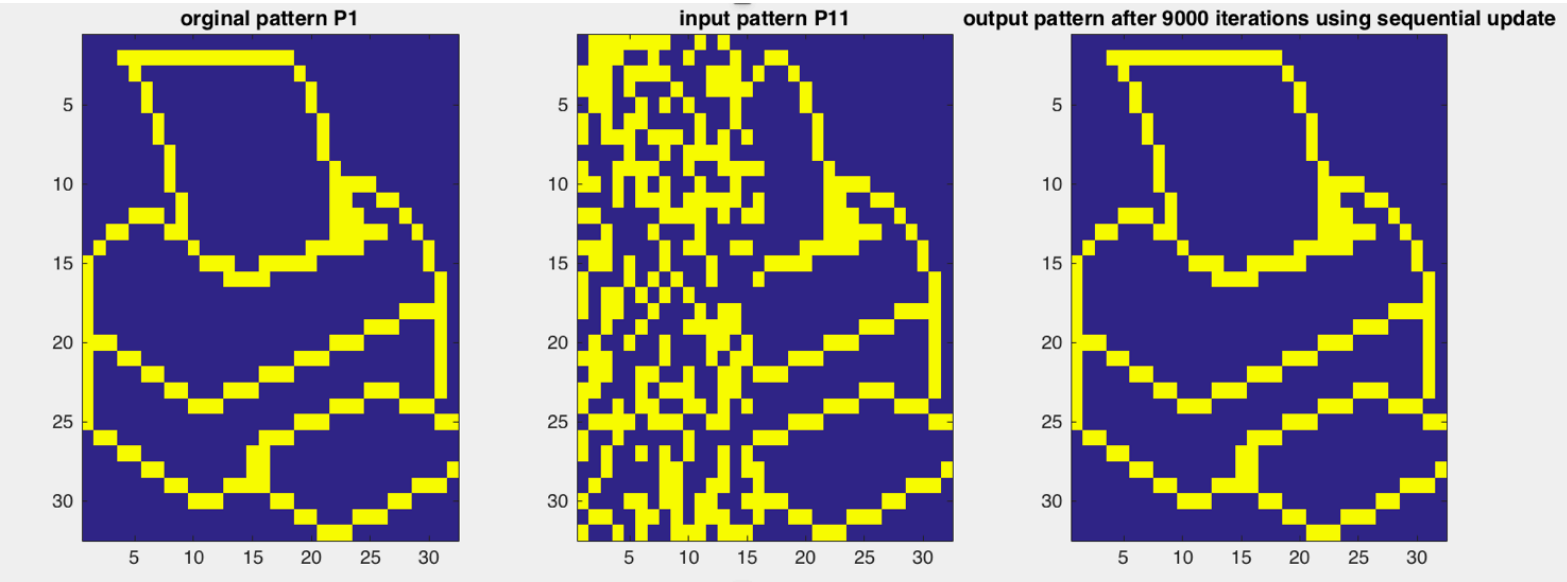
```
attractors =

     0     0     0     0     0     1     0     0
     0     0     0     0     1     0     0     0
     0     0     1     0     0     0     0     1
     0     0     1     0     0     1     0     1
     0     0     1     0     1     0     0     1
     0     1     1     0     0     1     0     1
     1     0     0     1     0     1     1     0
     1     0     0     1     1     0     1     0
     1     0     1     1     1     0     1     1
     1     1     0     1     0     0     1     0
     1     1     0     1     0     1     1     0
     1     1     0     1     1     0     1     0
     1     1     1     1     0     1     1     1
     1     1     1     1     1     0     1     1
```

14 attractors in total with 8 units

Sequential update

Only update one unit per iteration

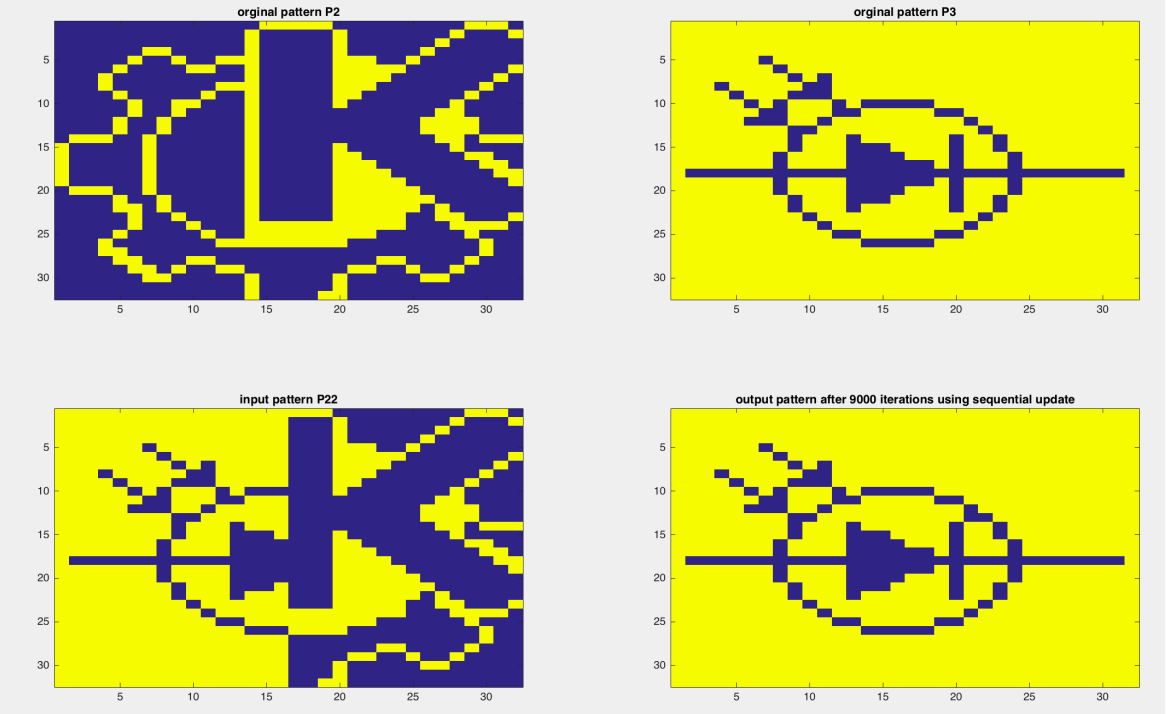
Using p1, p2, p3 to train Weight matrix; Input : p11 (degraded version of p1)



- The output has the same pattern as the stored pattern P1

Sequential update

Using p1, p2, p3 to train Weight matrix; Input : p22 (a mixture of p2 and p3)



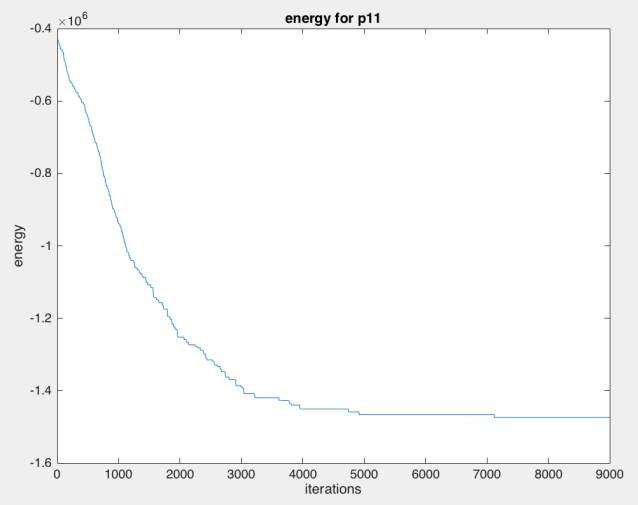
- The output pattern converge to the stored pattern p3

Energy

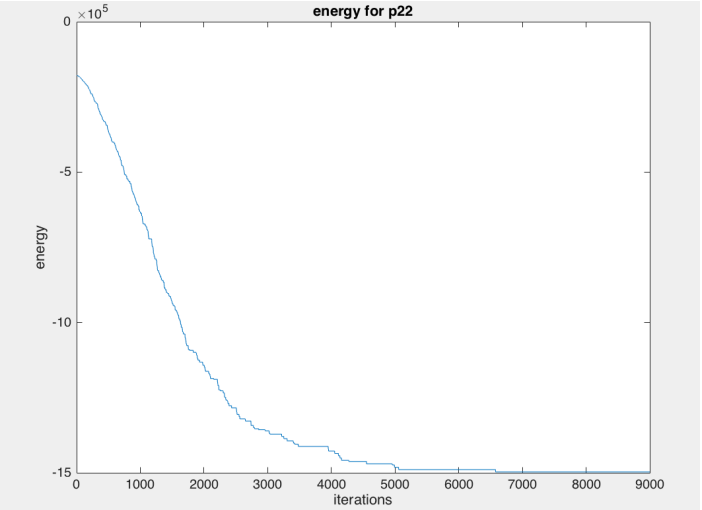
$$E = -\sum_i \sum_j w_{ij} x_i x_j$$

$$E = -x * w * x'$$

In MATLAB, we have



Energy for p11 with iterations



Energy for p22 with iterations

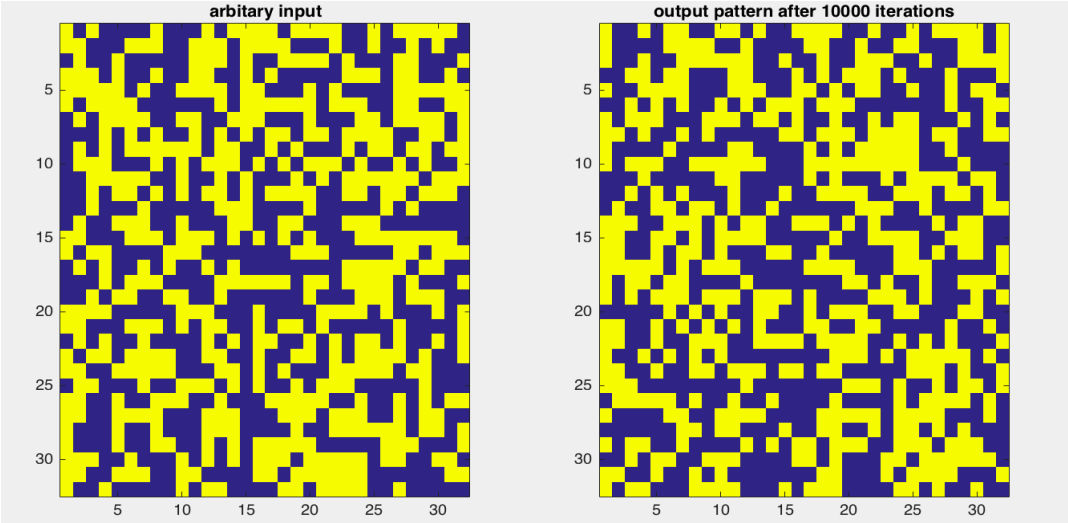
- With output converging, the energy also tends to converge to a lower level.
- Lower energy level, easier to reach the local minimal (attractors)
- Higher the input energy, more iterations needed

  (p22 has higher energy, more than 7000 iterations)

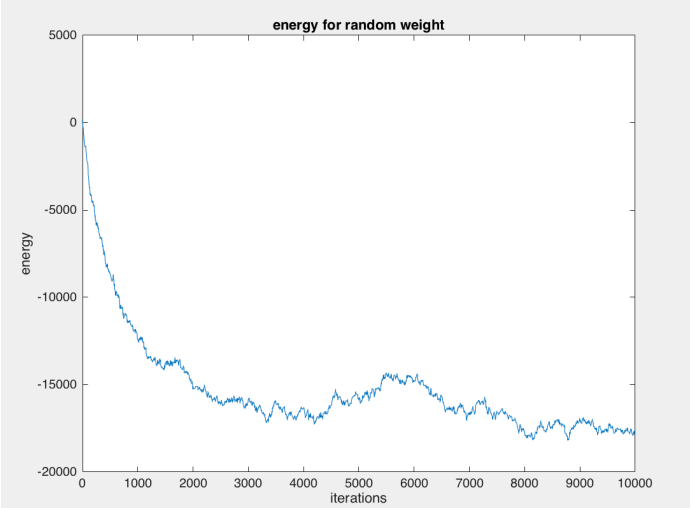
Energy

$$E = -\sum_i \sum_j w_{ij} x_i x_j$$

normally distributed random weight matrix:



With Random input

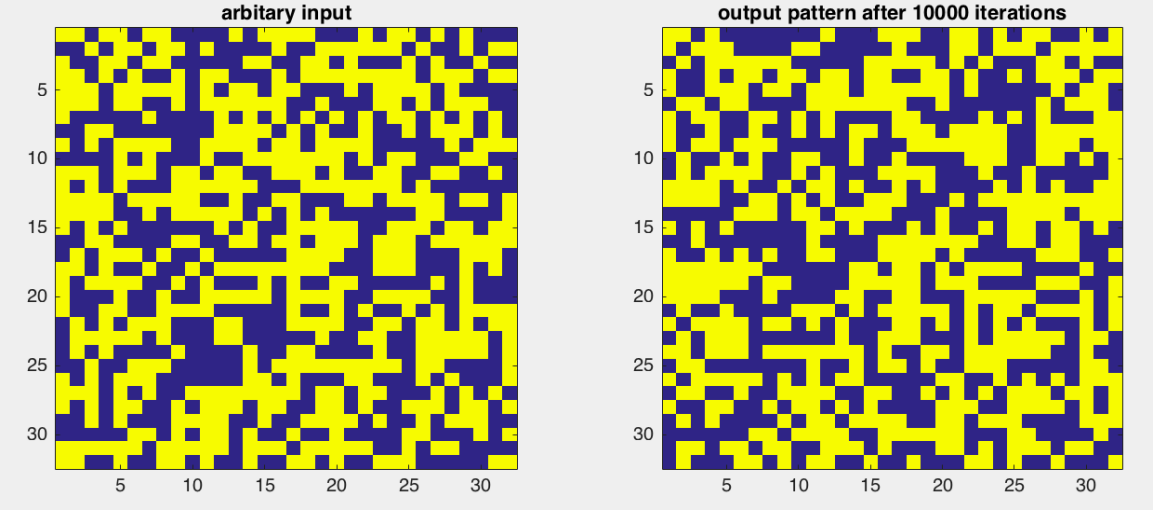


Hard to converge

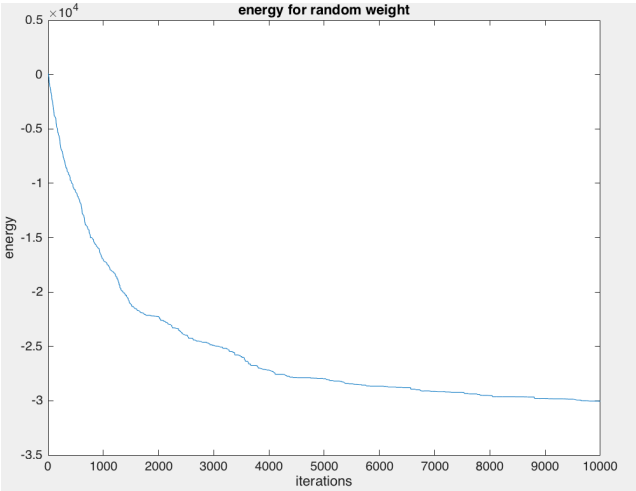
Energy

$$E = -\sum_i \sum_j w_{ij} x_i x_j$$

normally distributed random but still symmetric weight matrix:



With Random input

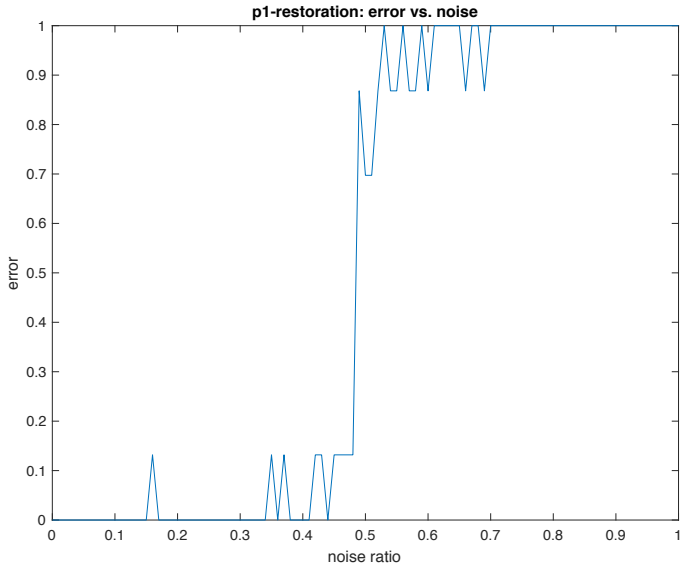


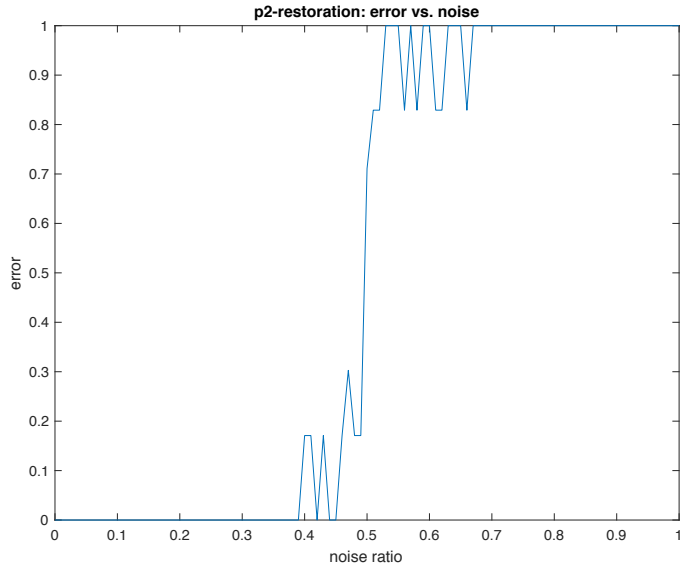
Energy converges

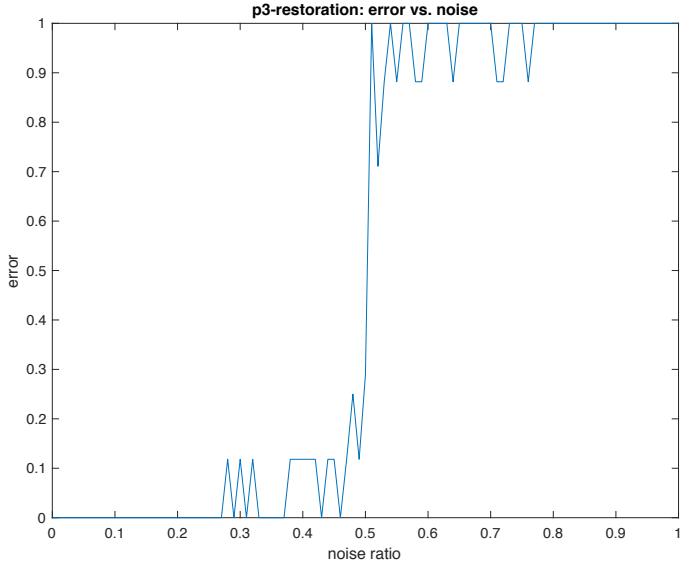
- **Weight element wij indicates the connection between xi and xj ,wji indicates the connection between xj and xi So $wij = wji$ → weight matrix must be symmetric**

Distortion Resistance

Train with p1-p3, also add noise to p1-p3





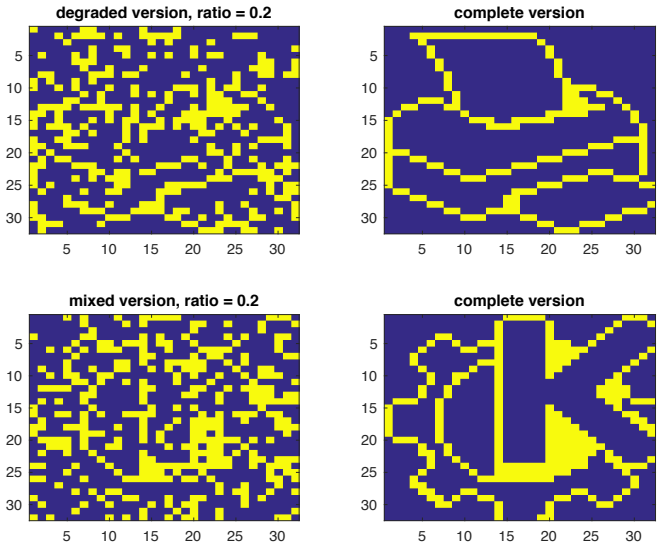


Observations:

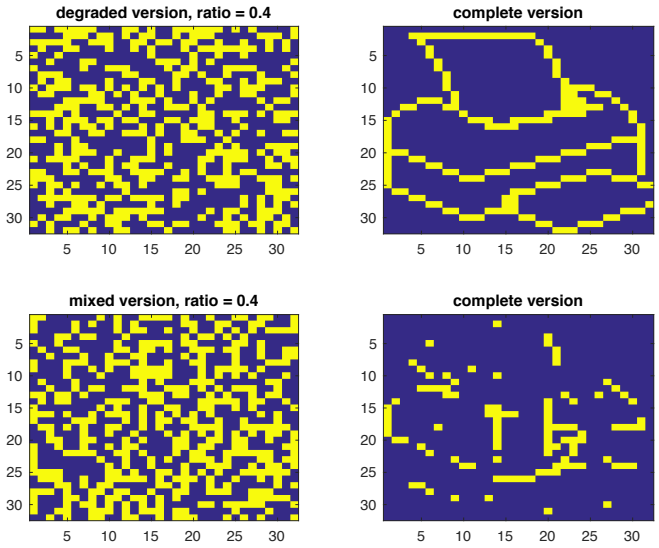
- Good restoration when noise ratio < 0.5
- Retrive inverse patterns when noise ratio > 0.5

Distortion Resistance

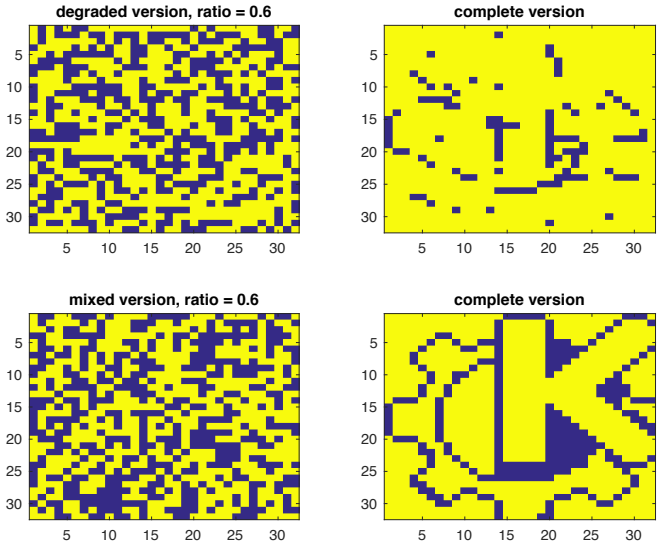
Train with p1-p3, also add noise to p1-p3



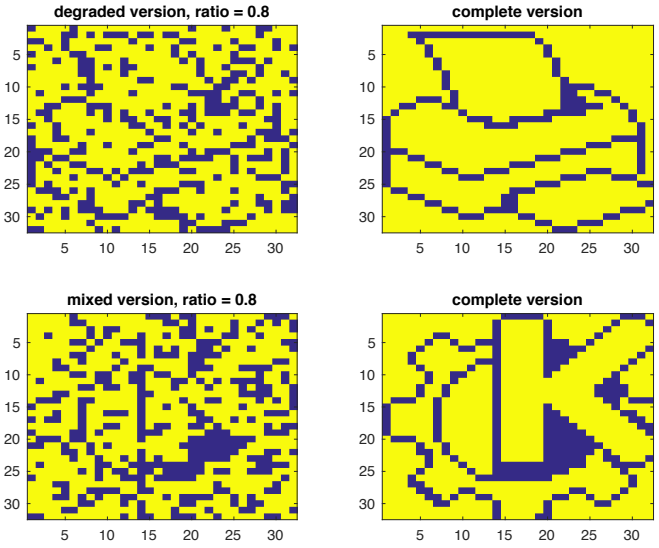
noise ratio = 0.2



noise ratio = 0.4



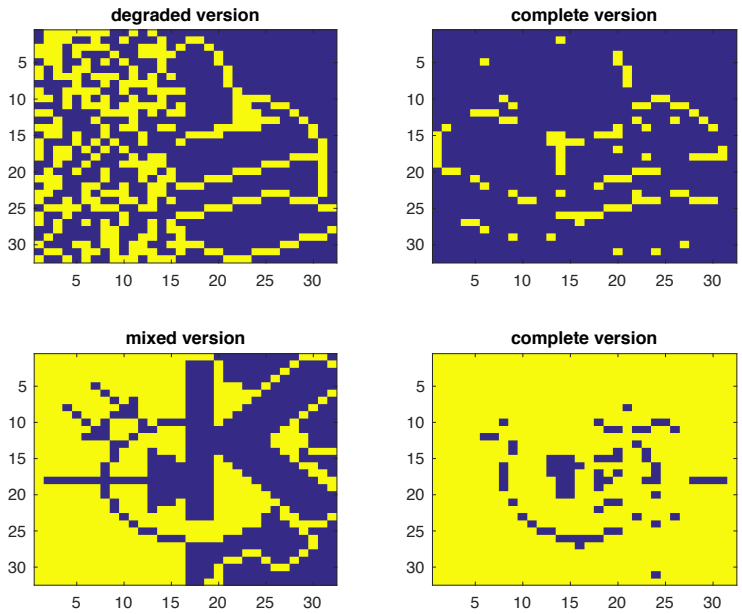
noise ratio = 0.6

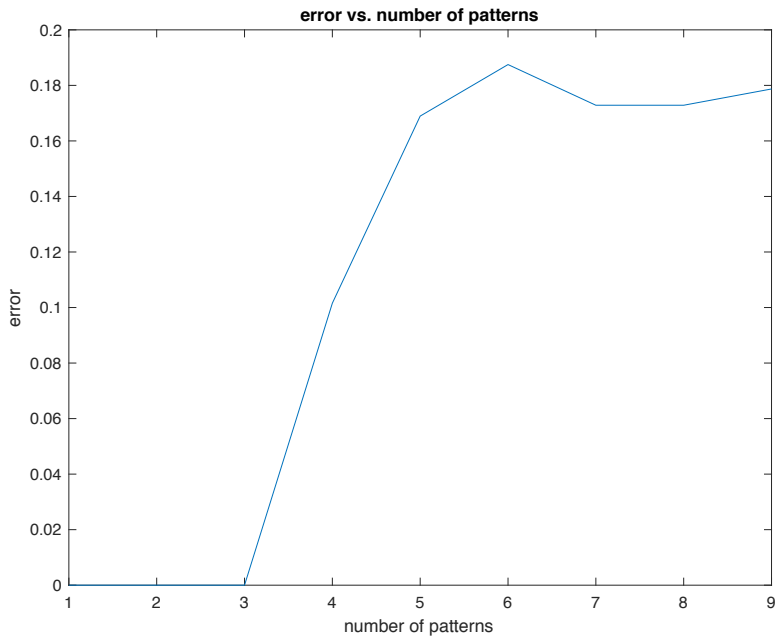


noise ratio = 0.8

Capacity

Train with p1-p4, no noise added



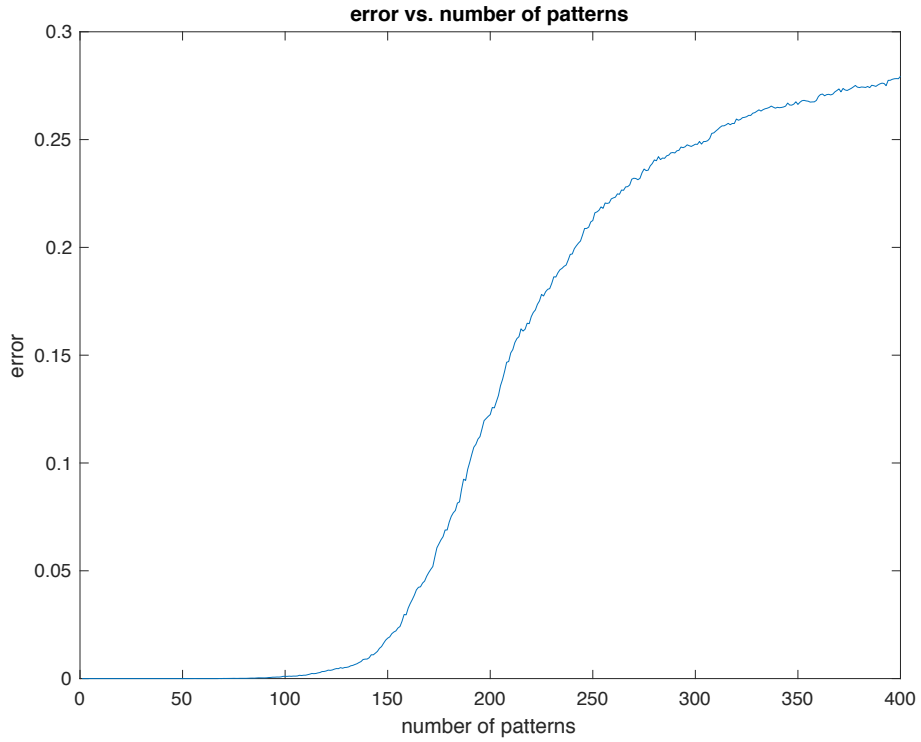


Observations:

- Three patterns could be safely stored
- Abrupt increase in errors

Capacity

Train with random patterns - error test

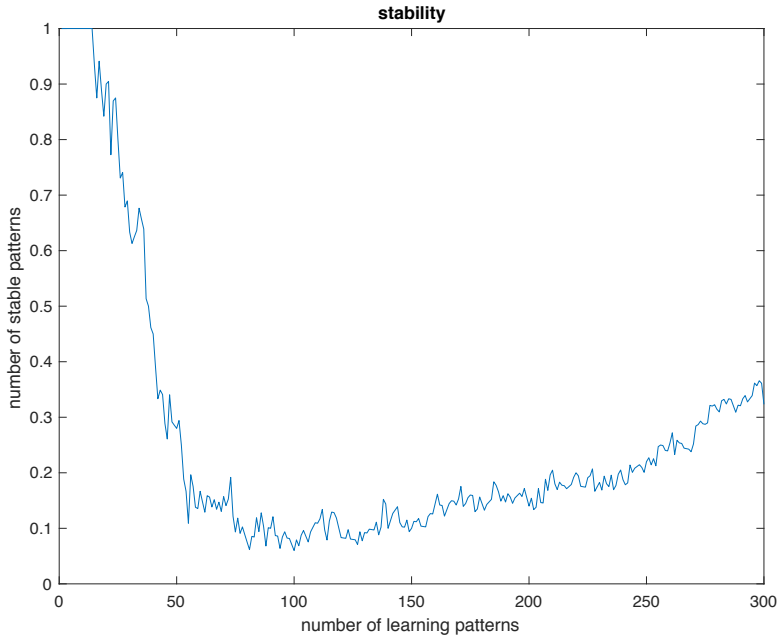


Observations:

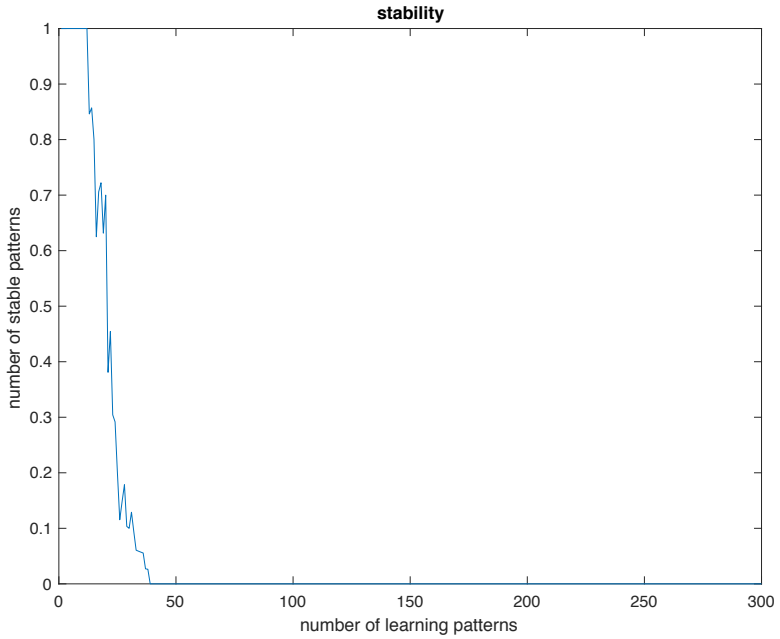
- Around 150 patterns could be safely stored (0.138N = 0.138*1024 = 141)
- Much more uncorrelated patterns, increased capacity

Capacity

Train with random patterns - stability test



with self-connections



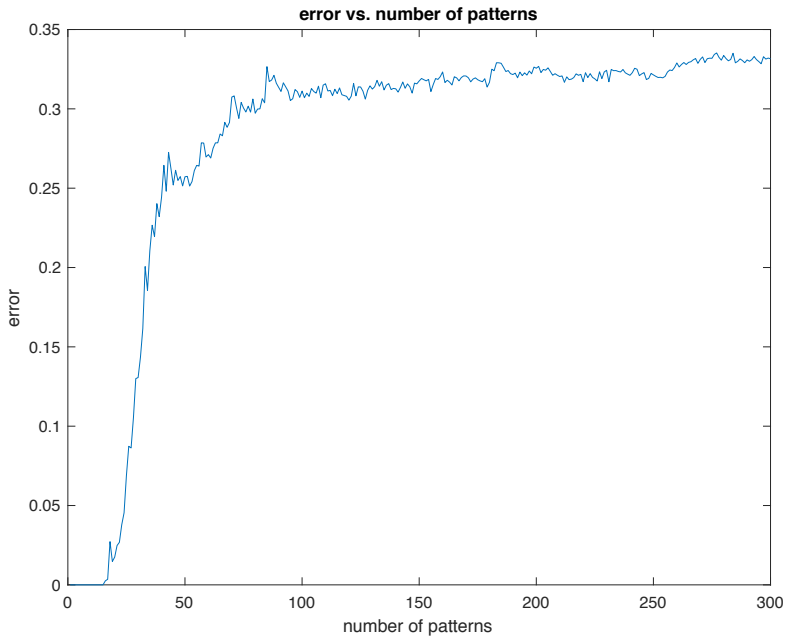
no self-connections

Observations:

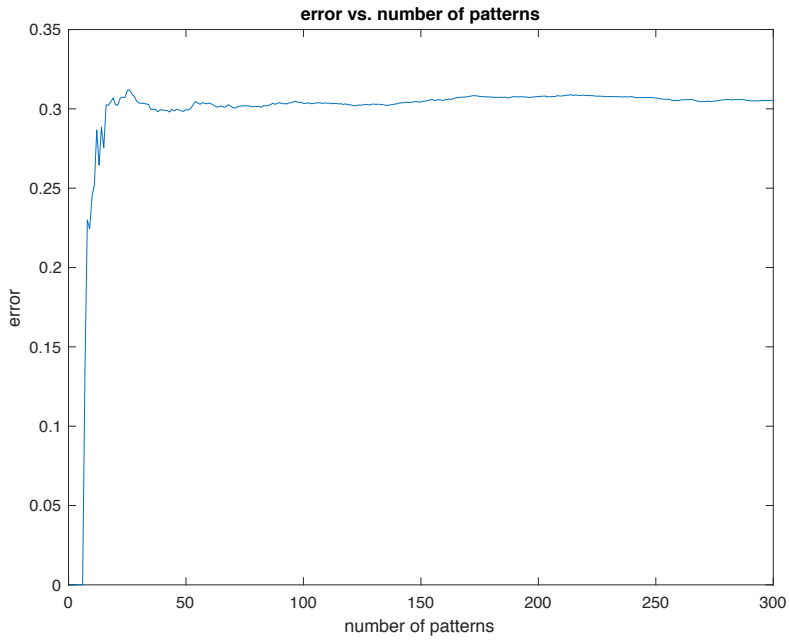
- Increase patterns -> decayed stability
- No self-connections -> hard to remain at current state -> even worse stability

Capacity

Train with random patterns - bias



No bias



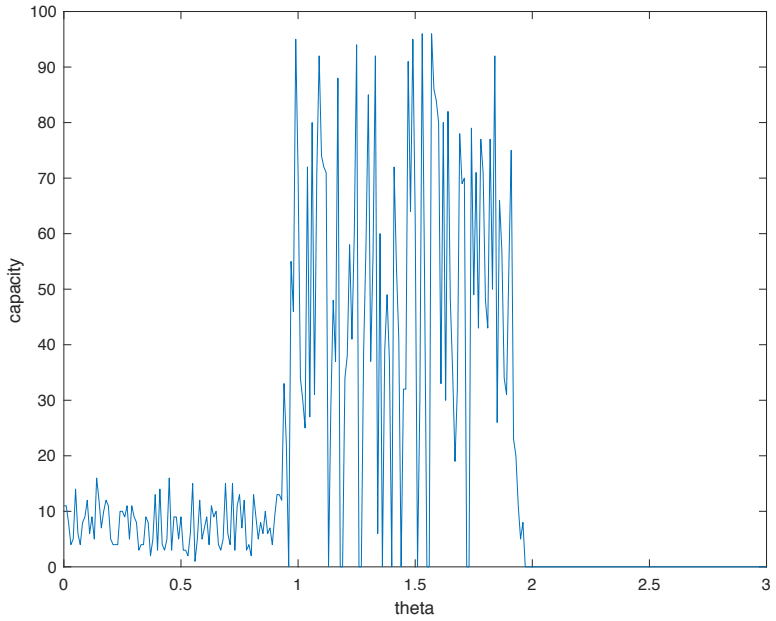
Bias = 0.5

Observations:

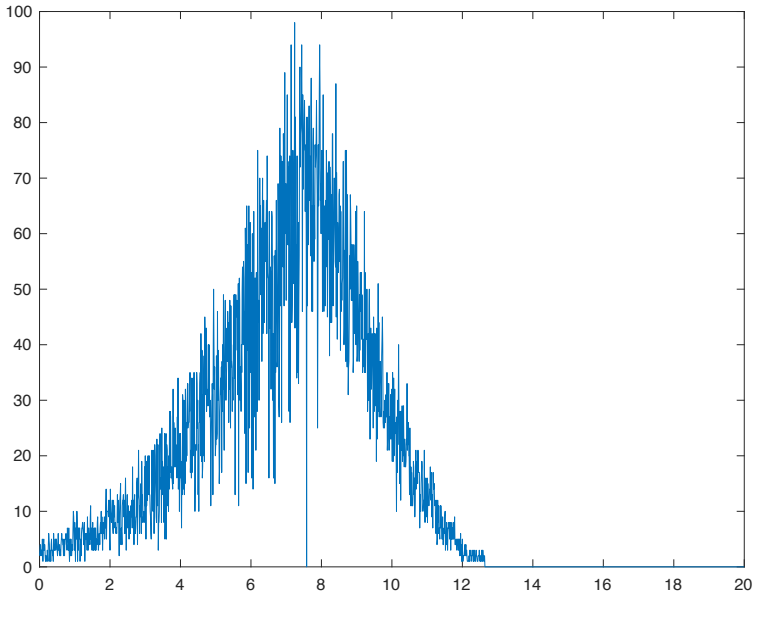
- Introduce bias -> worse capacity
- Bias -> patterns more correlated -> closer attractors -> easier to fall into spurious state -> worse capacity

Sparse patterns

Train with biased patterns



rho = 0.01



rho = 0.05

Observations:

- Small rho -> small optimal theta (positive link)
- Capacity first increases and then drops to zero as theta increases