Hopfield Networks and Applications

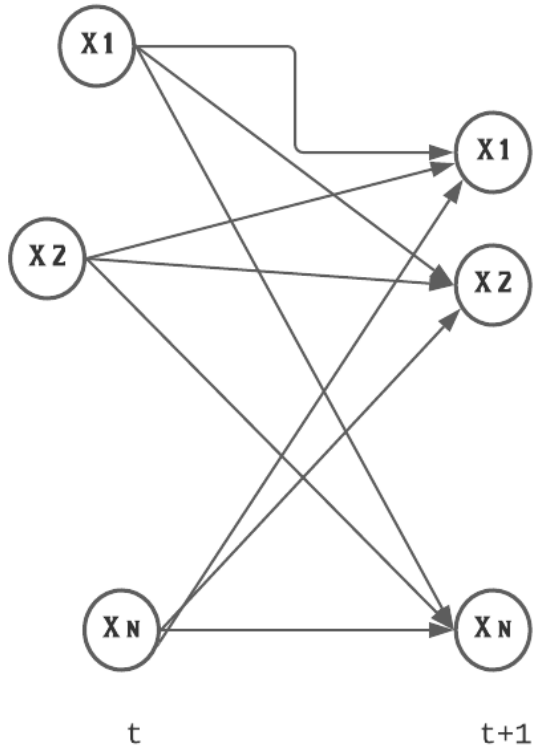
Rajarsi Pal

Indian Institute of Technology Madras // rajarsi14p@gmail.com

Mini-Project Presentation
@ Machine Learning for Physicists

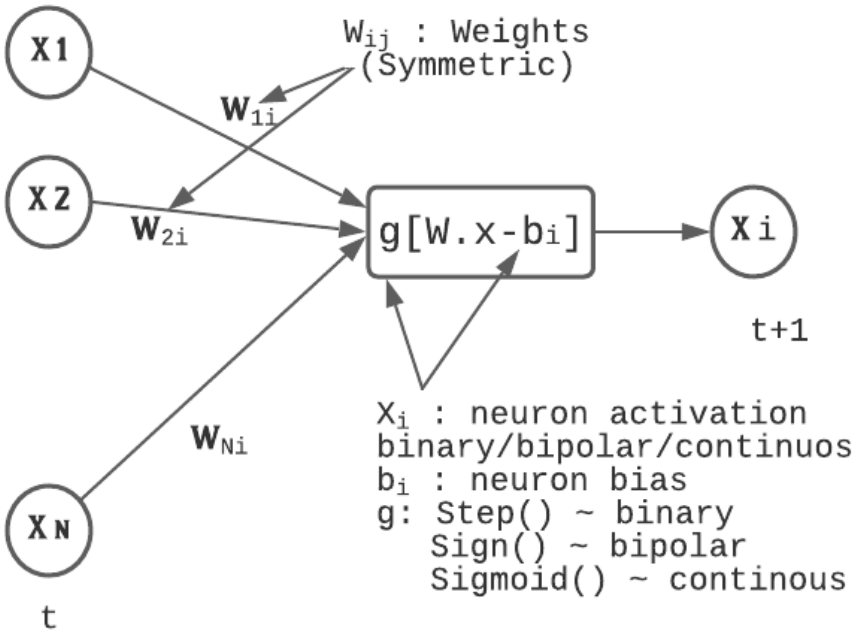
But what is a Hopfield network ?

A Hopfield Network is a type of recurrent neural network. They were introduced by J.Hopfield in 1982 as a model for associative memory.
Network architecture :



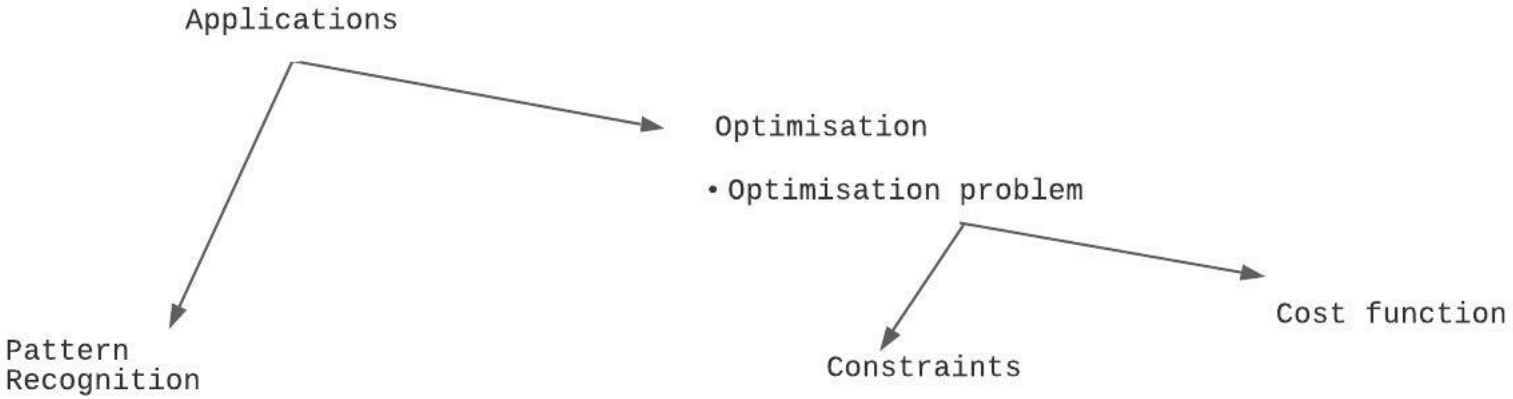
How does it work ?

- define **Energy** as, $E(X) = -\frac{1}{2N}\sum_{i,j} X_i W_{i,j} X_j + \sum_i b_i X_i$
- Neuron activation ;
  $X_i(t+1) = g_\beta(-\frac{\partial E}{\partial X_i}) = g_\beta(\frac{1}{2N}\sum_j W_{i,j} Xj - b_i))$, g_β is the activation function



- *Claim:* Energy decreases monotonically with iterations !

Applications



- Regenerate pattern from corrupted pattern
- Encode a given pattern into the weight matrix, such that stored patterns are at the Energy minima
- Since Energy decreseas monotonically to the minima, arbitrary initial configuration must converge to the stored pattern

- Reformulate the Cost function and Constraints into the Energy function
- By iteratively updating configurations Energy is minimised as Energy => Cost function, Cost function is minimised!

But how to store patterns ?

- Pattern to be stored, P_0 , P_0 is bipolar
- Then set Weight Matrix as, $W = P_0 P_0^T$, fix Bias $b = 0$, activation function : Sign()
- Note that if $X(t) = P_0$,

$$E(P_0) = -\frac{1}{2N} P_0^T P_0 P_0^T P_0$$

$$X(t+1) = Sign(P_0 P_0^T P_0) = P_0 = X(t)$$

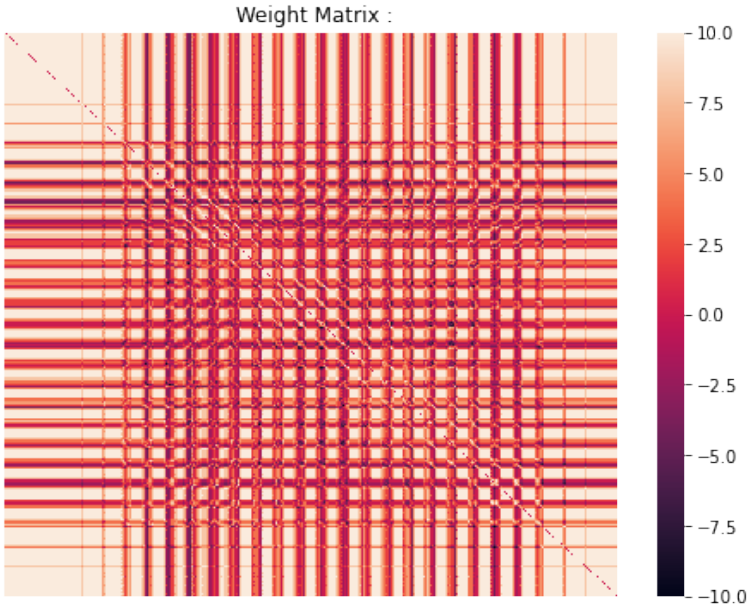
P_0 is a **fixed point** !

- for a set of patterns $P_0, P_1, P_2, ..$
$W = P_0 P_0^T + P_1 P_1^T + P_2 P_2^T + ..$

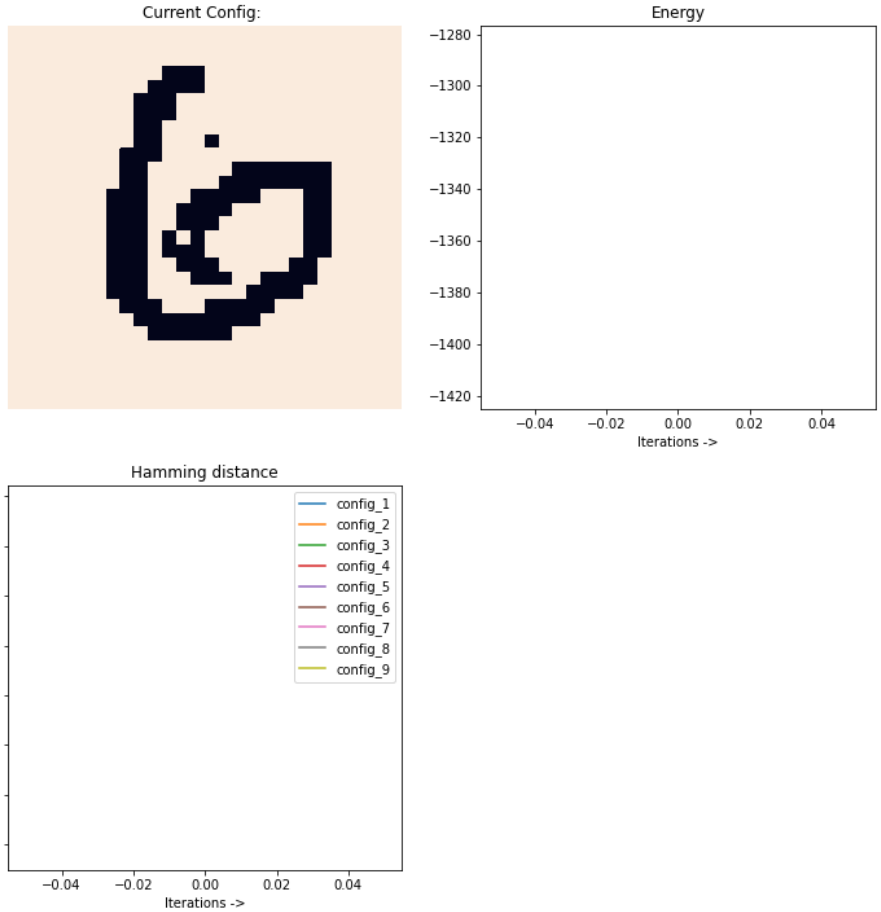
Storing MNIST data

- MNIST figures as stored patterns: $(28 \times 28 \text{ pixels}) = 784$ bipolar neurons.

Figure: Stored patterns



Retrieving a pattern



[?]

Does it really work ?

Why does it happen ?

- Overlap of stored configurations ! Leads to **Spurious** Energy minima sites off the original configurations.

Can we fix it ?

- Choosing configurations far apart i.e Hamming Distance (C1, C2) $\approx$ 0.5*N.
- Using Probabilistic Update routines, such that lower energy $\Rightarrow$ higher probability.

Hamming distance $D_h \equiv$ count of bits different.
Inner product $\equiv C^T X = N - 2D_h(C, X)$, E(X) $\approx (N - 2D_h(C, X))^2$

with some Random Patterns :

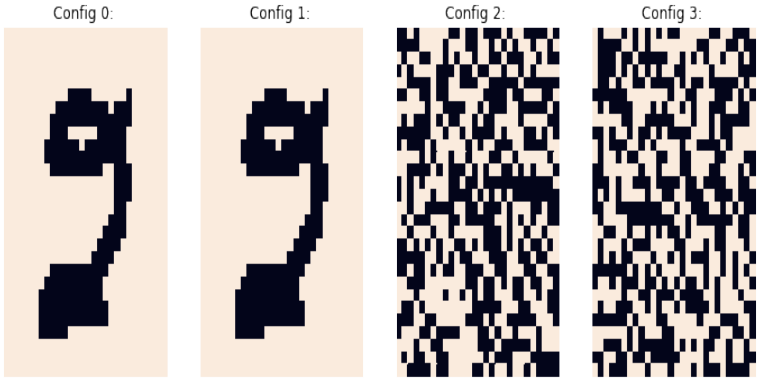
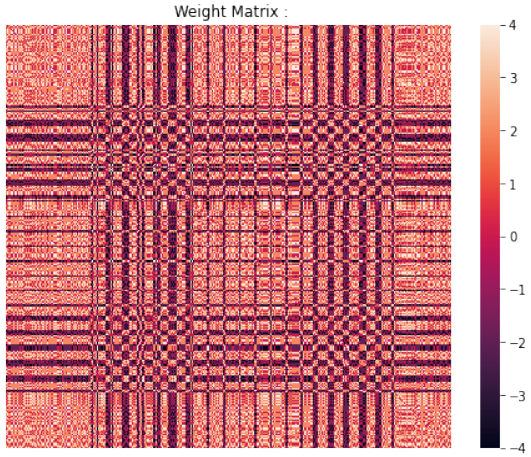
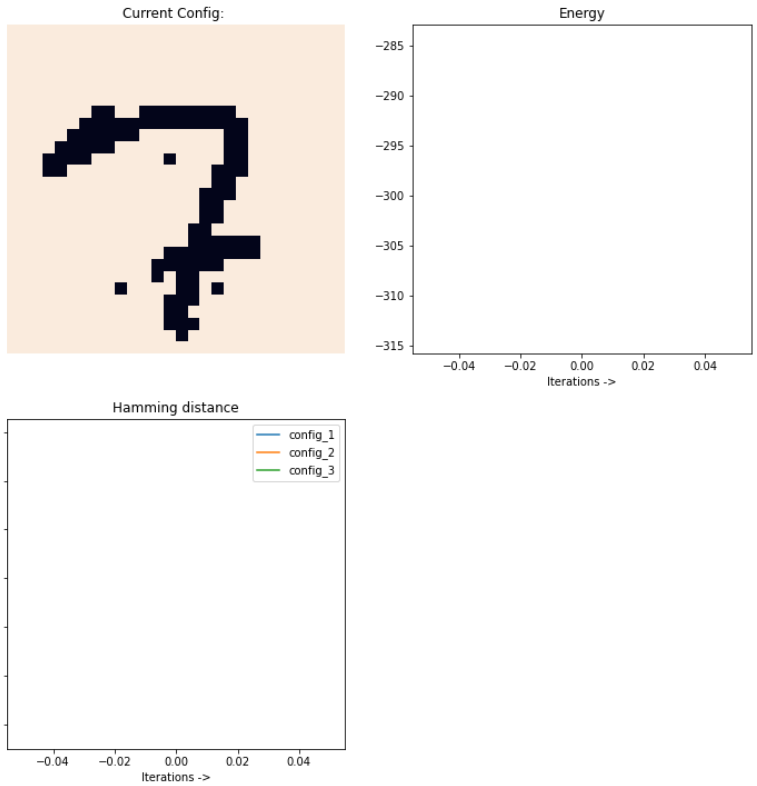


Figure: Stored patterns




Current Config:
Energy
−285
−290
−295
−300
−305
−310
−315
−0.04
−0.02
0.00
0.02
0.04
Iterations ->
Hamming distance
config_1
config_2
config_3
0.50
0.45
0.40
0.35
0.30
0.25
0.20
−0.04
−0.02
0.00
0.02
0.04
Iterations ->
[?]

- Hopfield's original paper 1982