A Hopfield Network for Digits Recognition

Gianluca Barbon

Università Ca'Foscari Venezia

gianluca.barbon@gmail.com

January 15, 2015

Overview

Aim of the project

This project consists in the implementation of a Hopfield Network using python. The aim of the implementation is to perform digit recognition. The network will be trained with different algorithms, in such a way to analyse performances and thus identify the best solution.

Hopfield Network

- recurrent artificial neural network
- proposed by John Hopfield in 1982
- content-addressable memory systems
  - main application: associative memories
- use of binary neuron units with sign activation function
- network converge to a local minimum (or lowest energy state).

Hopfield Network II

Main features:

- **sigle layer recurrent networks** in which each neuron is connected to all the others, with the exception of itself (so no cycles are admitted)
- **symmetric:** the synaptic weight matrix is symmetric, so $W = W^T$. This means that the weights is the same in both direction between two neurons.

The implemented Hopfield Network version uses *asynchronous update*. This means that neurons are updated one by one and picked randomly.

Formulas

Activation function

$$V_i = \begin{cases} +1 & \text{if } H_i > 0 \\ -1 & \text{if } H_i < 0 \end{cases}$$

Energy

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} V_i V_j - \sum_{i=1}^{n} I_i V_i$$

Energy variation

$$\Delta E = E(t+1) - E(t) \leq 0$$

Learning Algorithms features

Learning rule have some characteristics:

- **locality:** update of a given weight depends only on informations available to neurons on either side of the connection
- **incremental:** an incremental rule modifies the old network configuration to memorize a new pattern without needing to refer to any of the previous learnt patterns
- **immediate:** an immediate update of the network allows faster learning

Hebbian rule

Proportional weights are used in the activation between a pre and a post synaptic neurons:

Hebbian rule

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} x_i^{\mu} x_j^{\mu}$$

where N is the number of binary units with output $x_1, \ldots, x_N$.

Pseudo-inverse rule

Pseudo-inverse rule

$$w_{ij} = \frac{1}{N} \sum_{\substack{u=1 \\ v=1}}^{m} x_i^u (Q^{-1})^{uv} x_j^v$$

where m is the total number of patterns and q is:

q computation

$$q_{uv} = \frac{1}{N} \sum_{i=1}^{N} x_i^u x_i^v$$

This rule allows to improve the retrieval capability of the net with respect to Hebb rule, bringing the maximum number of pattern to $0.5N$. However, in this rule we have no local computation and no incremental updates, because it involves the calculation of an inverse.

Storkey's rule

Storkey proposes an algorithm to increase the capacity of the Hebbian rule without losing locality and incrementality.

Storkey's rule

$$w_{ij}^{\nu} = w_{ij}^{\nu-1} + \frac{1}{N} x_i^{\nu} x_j^{\nu} - \frac{1}{N} x_i^{\nu} h_{ji}^{\nu} - \frac{1}{N} h_{ij}^{\nu} x_j^{\nu}$$

where h_{ij} is:

h computation

$$h_{ij}^{\mu} = \sum_{\substack{k=1 \\ k \neq i,j}}^{n} w_{ij}^{\mu-1} x_k^{\mu}$$

Network Capacity and spurious patterns

- the maximum number of patterns that can be stored depends on neurons and connections.
- recall accuracy between vectors and nodes was 0.138 (Hertz et al., 1991)
- perfect recalls and high capacity, > 0.14, can be loaded in the network by Hebbian learning method.
- **Spurious patterns** can occur:
  - sometimes the network converges to spurious patterns, that are not in the set of training patterns
  - the energy in these spurious patterns is also a local minima.
  - also the negation of stored patterns is considered spurious

Implementation I

The Hopfield network is implemented with the HopfieldNet class:

- the net is immediately initialized with the given input and the given training algorithm at object creation
- class is provided with a test function
- the behaviour of the net is governed by the energy function

Implementation II

- **Hebbian rule**:
  - three versions
  - last one is chosen: dot products substitutes all loops
- **Pseudo-inverse rule**:
  - algorithm exploits dot product in the function for the computation of the Q matrix.
- **Storkey's rule**:
  - improvements where not possible because of complexity of the formulas
- both Pseudo-inverse and Storkey avoid the computation of half of the matrix, by taking advantage of the fact that the matrix is symmetric.

Testing with Courier Font Data Set

- Three different learning algorithms:
  Hebbian, Pseudo-inverse and Storkey
- **Courier Font Digits** dataset:
  - 10 tiff images of $44x70$ pixels in RGB colormap, with white background
  - each image contains a different digit written in black Courier font, from 0 to 9
  - lateral white spaces have been removed in order to improve algorithm efficiency
- image dimension of $14x9$ that leads to a network of 126 units

Testing with Courier Font Data Set II

Accuracy:

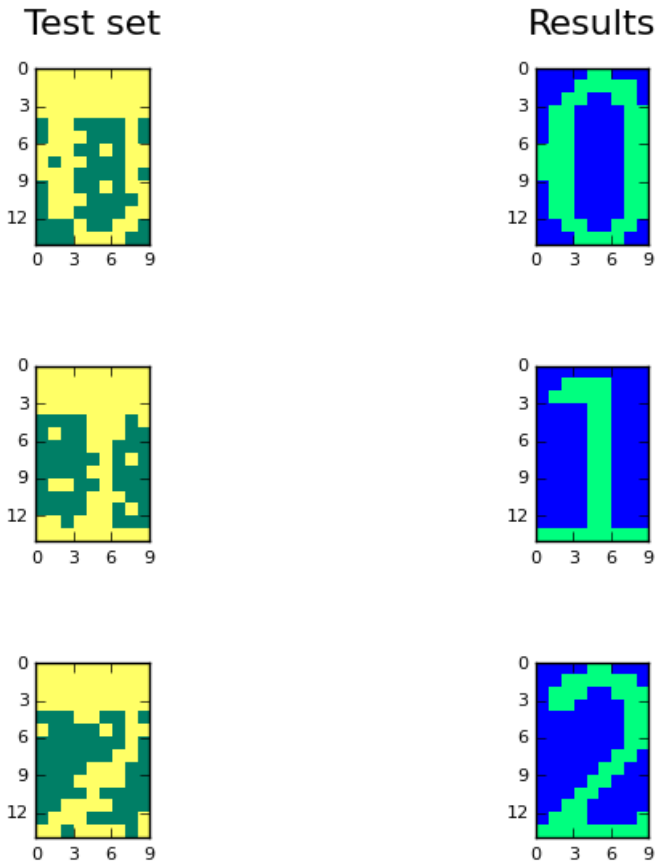
- Pseudo-Inverse resulted the best learning algorithm, with an accuracy of 100%
- Storkey resulted second with an accuracy of $98,7\%$
- Hebbian rule is the worst one, performing only an accuracy of $74,4\%$

Time performances:

- Hebbian resulted the best one, by training a network with 10 patterns in just 0.0002 seconds.
- Pseudo-inverse rule obtained 1.5 seconds for the same number of patterns
- Storkey's resulted the worst algorithm (21 seconds)

Testing with Courier Font Data Set III

Checking of the ability of the network to recall stored memories with the use of corrupted or partial images.



Testing with other Data sets

Digital 7 Font Digits

- chosen dimension is of $25x16$ (network of 400 units)
- aim: check the network behaviour with larger and clearer images.
- increased the number of errors: this font uses digits that are very similar.

Semeion Handwritten Data Set

- Courier Font Digits dataset as training, Semeion Handwritten Data Set as test set (digits picked randomly)
- pattern dimension: $16x16$ (total number of units to 256)
- the network appears to be able to recognize some of the handwritten digits

Filtering data set images

- Hebbian rule results are improved by adding a filter in the image sampling before the network training
- median filter provided by the Python Image Library, inserted after the conversion of the image to black and white
- it fills white pixels or removes useless pixels near the images edges
- Results: with the Hebbian tule, there were an improvement of about **9%** (with respect to results without the use of the filter)

Conclusions

- Convergence to wrong pattern occurs, specially with Hebbian training
- The Pseudo-Inverse rule algorithm resulted to be the best learning rule
- tests also shows the importance of the training images:
  - images that are very different between them perform better
  - digits that results in similar images, even if they represents different numbers, are mismatched by the network

References

Hebb, D. O. (1949)

The Organization of Behavior

Wiley, The Organization of Behavior

Wei, Gang and Yu, Zheyuan

Storage Capacity of Letter Recognition in Hopfield Networks

Faculty of Computer Science, Dalhousie University

Amos Storkey (1997)

Increasing the capacity of a Hopfield network without sacrificing functionality

ICANN97: Lecture Notes in Computer Science

Pelillo, M. (2014)

Artificial Intelligence Course notes

The End