Învățare Automată - Laboratorul 6
Rețele Hopfield

Tudor Berariu
Laboratorul AIMAS
Facultatea de Automatică și Calculatoare

24 martie 2014

1 Scopul laboratorului

Scopul laboratorului îl reprezintă înțelegerea și implementarea unei rețele Hopfield.

2 Rețele Hopfield

O rețea Hopfield este o rețea *asincronă* cu n neuroni, total conectată (fiecare neuron are intrările conectate la ieșirile tuturor celorlalți $n-1$ neuroni). O rețea este asincronă facă fiecare unitate (neuron) își actualizează starea la momente de timp aleatoare, independent de timpii de actualizare ale cerlolalte unități.

Într-o rețea Hopfield *funcția de activare* (actualizare) pentru un neuron este cea din Formula 1.

$$x_i \longleftarrow sgn\Big(\sum_{j=1}^{n} w_{ij} x_j\Big) \tag{1}$$

Utilizând principiul lui Hebb, o rețea Hopfield poate fi folosită ca o memorie asociativă pentru a reține un număr de șabloane binare (un șablon va fi format din n valori din mulțimea $\{-1, 1\}$). Ponderile unei rețele Hopfield

se calculează pe baza celor m șablaone ($\mathbf{s}^i, 1 \leq i \leq m$) conform Formulei 2 (învățare hebbiană).

$$\mathbf{W} = \sum_{i=1}^{m} \mathbf{s}^i \cdot (\mathbf{s}^i)^T \ - \ m\mathbf{I} \tag{2}$$

Atenție: $w_{ii} = 0, \forall i \in \{1 \dots n\}$.

Pentru a folosi rețeaua ca un clasificator, după ce aceasta a fost *antrenată* (Formula 2), se utilizează Algoritmul 1.

Algoritmul 1 Recunoașterea șabloanelor

Intrări: ponderile rețelei W, șablonul nou t
Ieșire: șablonul recunoscut s

1: $s \longleftarrow t$
2: **repetă**
3: alege aleator un neuron i
4: $s_i \longleftarrow sgn\Big(\sum_{j=1}^{n} w_{ij} s_j\Big)$
5: **până când** stările de activare ale neuronilor nu se mai schimbă

3 Cerințe

În cadrul acestui laborator se va implementa o rețea Hopfield pentru a recunoate imagini ce reprezintă cifrele de la 0 la 9. O imagine este formată din 10×12 pixeli, precum cea de mai jos

```
__xxxxxxxx__
__xxxxxxxxx_
________xxx_
________xxx_
___xxxxxxx__
___xxxxxxx__
________xxx_
________xxx_
__xxxxxxxxx_
__xxxxxxxx__
```

Limbajul de programare este la alegere, dar pentru Python, C++ și Octave există un schelet de cod cu următoarele funcții implementate:

- `read_digits(m)` - citește primele m șabloane din fișierul `digits`
  - fiecare șablon este memorat ca un vector / listă de lungime 120 conținând doar valori 1 și -1
  - rezultatul funcției este o colecție de m astfel de vectori (o matrice $m \times 120$ cu șabloanele pe linii)
- `print_digit(d)` - afișează un șablon
- `add_noise(pattern, noise)` - întoarce un șablon nou adăugând zgomot unui șablon original
  - $0 \leq noise \leq 1$ reprezintă probabilitatea cu care un *pixel* al șablonului este schimbat din 1 în -1 și invers

Cerințe

Implementați următoarele funcții:

1. `compute_weights(patterns)` - în care calculați ponderile unei rețele Hopfield pentru șabloanele primite.
2. `converge(weigths, new_pattern)` - în care implementați algoritmul de recunoaștere a șabloanelor pentru rețele Hopfield.
3. `compute_accuracy(patterns, noise)` - în care construiți o rețea Hopfield care să memoreze șabloanele `patterns`, generați imagini noi adăugând zgomot celor originale și testați capacitatea rețelei de a reface șablonul inițial.

Bonus

1. Construiți un grafic al variației acurateței clasificării în funcție de nivelul de zgomot. Fixați mai multe valori pentru numărul de șabloane învățate și adăugați câte o linie în grafic pentru fiecare.

2. Construiți un grafic al variației acurateței clasificării în funcție de numărul de șabloane învățate. Fixați mai multe valori pentru nivelul de zgomot adăugat la recunoaștere și adăugați câte o linie în grafic pentru fiecare dintre acestea.

Arhiva

Arhiva conține schelet de cod în Octave (sau Matlab), C++11 și Python.