Învățare Automată
Laboratorul 6: **Rețele Hopfield**

Tudor Berariu
Laboratorul AIMAS
Facultatea de Automatică și Calculatoare

27 martie 2012

1 Rețele Hopfield

O rețea Hopfield este o rețea *asincronă* cu n neuroni total conectată (fiecare neuron are intrările conectate la ieșirile celorlalți $n-1$ neuroni). O rețea este asincronă dacă fiecare unitate (neuron) își actualizează starea la momente de timp aleatoare, independent de timpii de actualizare ale celorlalte 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. Ponderile unei rețele Hopfield se calculează pe baza celor m șabloane ($\mathbf{s}^i, 1 \leq i \leq m$) conform formulei 2 (învățare Hebbiană). Un șablon este format din n valori $\in \{-1, 1\}$.

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

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

Pentru a folosi rețeaua ca un clasificator (care recunoaște șabloanele noi similare celor învățate) se folosește Algoritmul 1.

Algoritmul 1 Recunoașterea șabloanelor

Intrări: ponderile W, șablonul nou t
Ieșire: șablonul învățat s

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

2 Cerințe

În acest laborator veți implementa o rețea Hopfield care să recunoască imagini ce reprezintă cifrele de la 0 la 9. O imagine are o rezoluție de 10×12 pixeli.

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

Cerințele din acest laborator vor fi rezolvate în Matlab. Patru funcții Matlab sunt deja scrise:

- `read_digits(m)` - citește primele m șabloane din fișierul `digits`
  - fiecare șablon este memorat ca un vector de dimensiune 1×120 de valori 1 și -1
  - rezultatul funcției este o matrice $m \times 120$ unde fiecare linie reprezintă un șablon
- `print_digit(d)` - afișează un șablon (primește un vector 1×120)

- `add_noise(pattern, noise)` - adaugă zgomot unui șablon
  - $0 \leq noise \leq 1$ reprezintă probabilitatea cu care un *pixel* al șablonului este schimbat din 1 în -1 sau invers
- `compute_accuracy(weights, learned_patterns, noise)`
  - estimează performanța rețelei de a clasifica (repara) variante ale șabloanelor învățate afecate de zgomot

Cerințe:

1. Scrieți o funcție matlab care învață ponderile pentru m șabloane inițiale.
   - funcția va avea antetul `compute_weights(patterns)` unde `patterns` este o matrice de dimensiune $m \times 120$
2. Scrieți o funcție matlab care repară o reprezentare cu zgomot a unui șablon.
   - funcția va avea antetul `converge(weights,new_pattern)` unde `weights` sunt ponderile calculate la punctul 1, iar `pattern` este un șablon din cele învățate asupra căruia s-a aplicat zgomot.

Bonus:

1. Construiți un grafic al variației acurateței clasificării în funcție de nivelul de zgomot (probabilitatea de a *schimba* un pixel al imaginii).
   - Utilizați funcția `plot`.
2. Construiți un grafic al variației acurateței de clasificare în funcție de numărul de șabloane învățate ($1 \leq m \leq 10$) - câte o linie de grafic pentru fiecare valoare a nivelului de zgomot $noise \in \{0.1, 0.15, 0.20, 0.25, 0.3\}$.