Învățare Automată - Laboratorul 6
Rețele Hopfield

Tudor Berariu
Laboratorul AIMAS
Facultatea de Automatică și Calculatoare

28 martie 2016

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, complet conectată (fiecare neuron are intrările conectate la ieșirile tuturor 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 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 învățării Hebbiene, 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

3.1 Cerința 1

```
python test_hopfield.py --patterns_no 2 --train_lr 0.1 --task 1
```

3.1.1 Antrenarea rețelei Hopfield

În clasa `HopfieldNetwork` implementați metoda `learn_patterns` în care se determină valorile ponderilor rețelei. Parametrul `patterns` este o listă de șiruri de caractere formate doar din "_" și "X" reprezentând șabloanele ce trebuie *memorate* în rețea. Ponderile trebuie calculate în `self.weights`.

3.1.2 Actualizarea unui neuron

În clasa `HopfieldNetwork` implementați metoda `single_update` care actualizează un singur neuron astfel încât energia rețelei să scadă. Stările neuro-

nilor se găsesc în vectorul `self.state` și trebuie să fie -1 sau 1.

3.1.3 Calcularea energiei unei configurații

În clasa `HopfieldNetwork` implementați în metoda `energy` calculul energiei pentru starea curentă a rețelei. Ponderile rețelei sunt în matricea `self.weights` (calculat anterior), iar starea rețelei se găsește în vectorul `self.state`.

3.1.4 Deteriorarea unui șablon

În scriptul `test_hopfield.py` implementați funcția `apply_noise` care primește un șablon (un șir de caractere) și o probabilitate de inversare a fiecărei componente a șablonului.

3.2 Testarea rețelei Hopfield

```
python test_hopfield --patterns_no 3 --noise 0.15 --task 2
```

În scriptul `test_hopfield.py` implementați funcția `test_hopfield` care calculează capacitatea rețelei de a reface un șablon afectat de zgomot.

3.3 Calculul distribuției de șabloane învățate de rețea

În clasa `HopfieldNetwork` implementați metoda `get_final_states_distribution` care aproximează distribuția de stări corespunzătoare unor minime energetice locale. Fiecare stare corespunde, desigur, unui șablon. Argumentul `samples_no` controlează numărul de probe folosite pentru aproximarea distribuției. Inițializați rețeaua de fiecare dată cu o stare aleatoare și conduceți-o către un minim energetic. Construiți histograma acestor stări finale.

3.3.1 [Bonus] Dezvățarea șabloanelor nedorite

Observați că pentru un număr mai mare de șabloane, funcția de energie dezvoltă puncte de minim local ce nu corespund unui șablon din setul de antrenare.

Forțați rețeaua să *uite* acele șabloane printr-un proces de dezvățare.

Implementați metoda `unlearn_patterns` din clasa `HopfieldNetwork`.

Evaluați rețeaua înainte și după procesul de *dezvățare*. Încercați mai multe valori pentru rata de învățare.