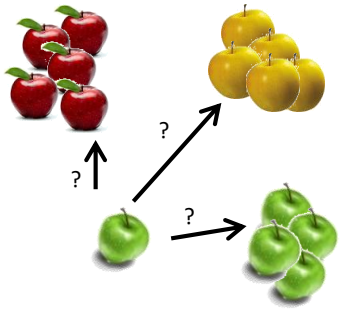
PROCESAREA CUNOȘTINȚELOR ȘI CALCUL INTELIGENT - PCCI

Reţele neuronale artificiale pentru clasificare

Recapitulare: Clasificare şi clustering

- Clasificare = repartizare a unui obiect (model de intrare) în una dintre mai multe categorii cunoscute.



Cărei clase îi aparţine noul măr ?

- Clustering (grupare) = identificarea categoriilor existente într-un grup de date (modele de intrare) despre care iniţial nu se cunoaşte nimic.

Câte feluri de mere sunt în grămadă ?
Cum arată fiecare fel de măr ?
Care măr aparţine cărui fel ?

Recapitulare:
Algoritme de clasificare şi clustering

- Algoritm de clasificare: *algoritmul celor mai apropiaţi k vecini* (k-nearest neighbors, acronim kNN)

- Algoritm de clusterizare: *k-medii* (k-means)

Recapitulare: Algoritmul kNN

- Definirea bazei de date: se dă un număr de elemente pentru care se cunoaşte clasa căreia aparţine fiecare dintre ele, c, $c=1 \ldots C$.
- Definirea numărului de vecini k
- Prezentarea noului element, care se doreşte a fi clasificat
- Calcul distanţă (de obicei euclidiană) dintre noul element şi elementele cunoscute din baza de date
- Sortarea în ordine crescătoare a distanţelor şi reţinerea primelor k elemente cu cele mai mici distanţe calculate
- Identificarea claselor cărora aparţin cele k elemente reţinute
- Clasa c căreia îi va fi asociat noul element va fi clasa dominantă (care apare de cele mai multe ori)

Recapitulare: Algoritmul k-means

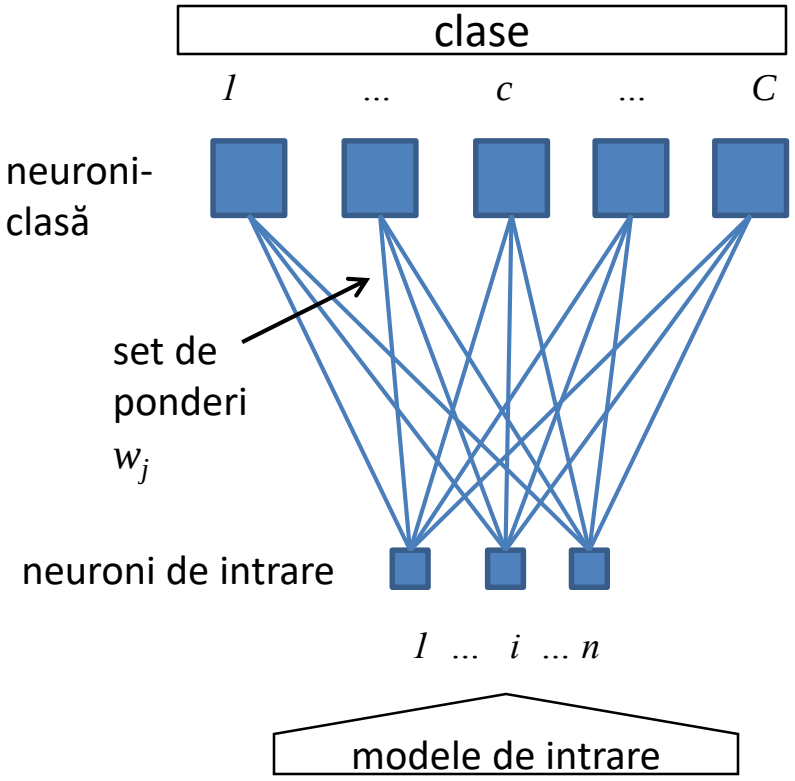
- Definirea bazei de date: un număr de elemente pentru care nu se cunoaşte clasa de apartenenţă
- Definirea unui număr fix de k centre, dintre punctele existente în baza de date (analog numărului C din algoritmul kNN)
- Repetă până la îndeplinirea criteriului de oprire
  - Calculul distanţei între fiecare centru şi toate punctele din baza de date
  - Alocarea pentru centrul k a punctelor celor mai apropiate ca distanţă Recalcularea fiecărui centru ca medie a elementelor asociate centrului respectiv în iteraţia curentă
  - Verifică condiţia de oprire: nici un punct nu a schimbat centrul căruia i-a fost alocat.

Cuprins:

- Reţele neuronale artificiale folosite pentru clasificare
  - Un algoritm general pentru RNA clasificatoare
  - Algoritmul Learning Vector Quantizer (LVQ) pentru învăţare supravegheată
  - Self Organizing Feature Maps (hărţi cu autoorganizare) (SOFM) sau RNA Kohonen, pentru învăţare nesupravegheată.

Modelul general al unei RNA clasificatoare

- RNA cu două straturi: unul de intrare şi unul de ieşire
- Un neuron de ieşire este legat prin ponderi de toţi neuronii de intrare; un asemenea set de ponderi este notat cu w_j
- Numărul de neuroni de intrare este egal cu n, numărul de elemente care formează un model de intrare; un model generic are indicele m, $m=1...M$ (se consideră în general M modele în setul de antrenare).
- Numărul de neuroni de ieşire este egal cu numărul de clase cunoscute C; un model generic este notat c, $c=1...C$.



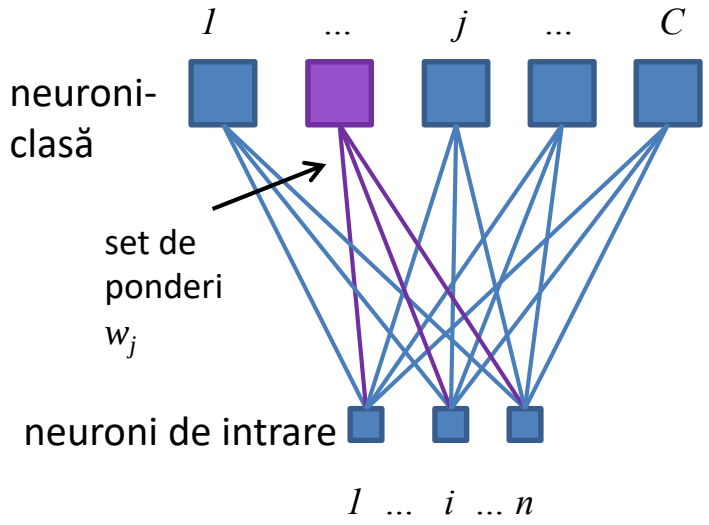
Modelul general al unei RNA clasificatoare

- Se poate recunoaşte modelul unui neuron elementar
- Fiecare neuron de ieşire corespunde unei clase; ponderile sale w_j formează prototipul (codebook vector) asociat clasei respective
- Algoritmul are două etape:
  - Identificarea claselor posibile folosind un set de date de antrenare
  - Recunoaşterea sau clasificarea unor modele noi într-una din clasele existente

- Funcţia de activare:

$$y = \begin{cases} 1, \textit{ dacă modelul de intrare curent este asociat clasei } c \\ 0, \textit{ în caz contrar} \end{cases}$$

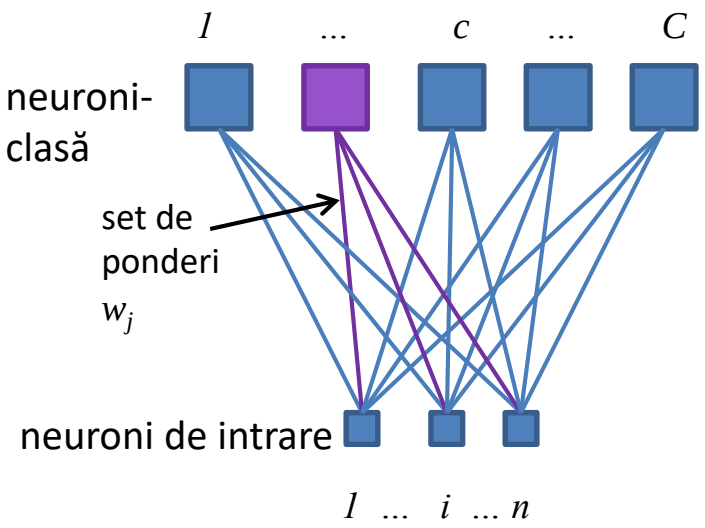
$$c = 1..C$$



Etapa de antrenare

- Ponderile neuronale sunt iniţializate aleatoriu, de obicei în intervalul [0,1]
- Algoritm iterativ; într-o iteraţie se parcurg următorii paşi:
  - Fiecare model de intrare este comparat, prin calculul unei distanţe (de obicei cea euclidiană) cu prototipurile tuturor neuronilor-clasă
  - Este ales drept neuron câştigător cel cu distanţa calculată cea mai mică
  - Ponderile neuronului clasă ales sunt corectate pentru a se apropia de modelul de intrare respectiv
  - Dacă s-a îndeplinit condiţia de oprire, procesul iterativ se opreşte
- În finalul algoritmului, valorile ponderilor fiecărui neuron-clasă codifică caracteristicile clasei respective, determinate pe baza setului de antrenare disponibil

$$d_{m,j} = (x^{(m)} - w_j) = \left\| x^{(m)} - w_j \right\|$$

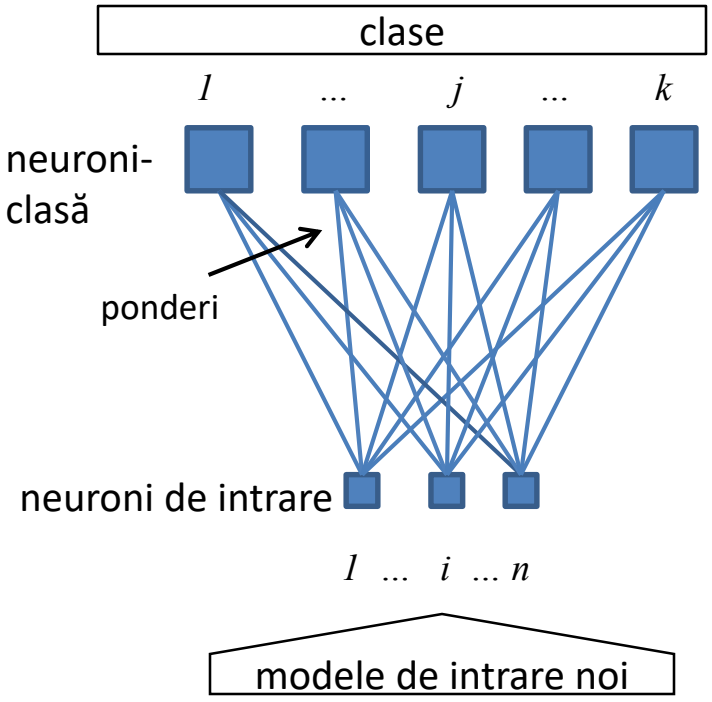


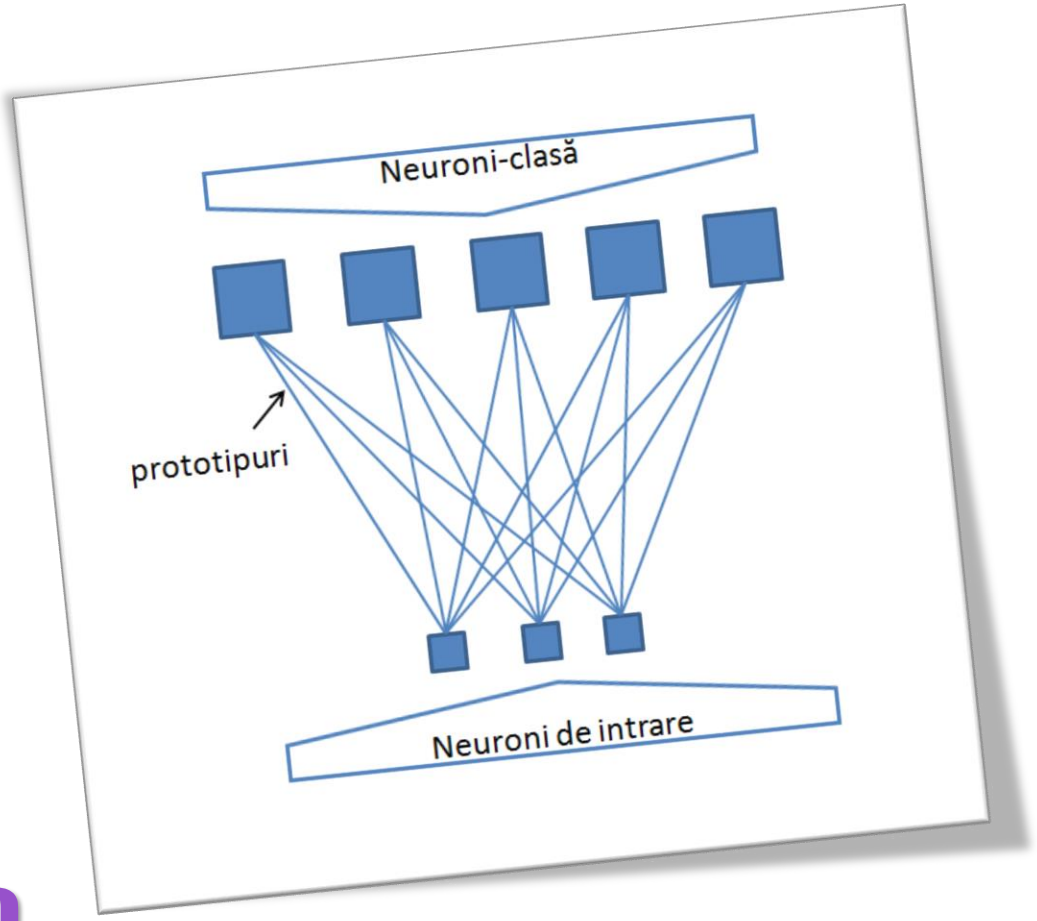
Etapa de recunoaştere

- Pentru fiecare model de intrare nou, care nu a fost folosit în etapa de antrenare, se calculează distanţa până la fiecare neuron-clasă, la fel ca în etapa de antrenare
- Este aleasă drept clasă câştigătoare cea cu distanţa calculată cea mai mică

- Antrenarea poate fi refăcută ulterior, dacă apar clase noi.

- RNA de acest tip se mai numesc şi RNA competitive





Algoritmul LVQ
(LEARNING VECTOR QUANTIZER)

Algoritmul LVQ

- Learning vector quantizer = cuantificator vectorial cu învăţare
- Inventat de profesorul finlandez Teuvo Kohonen (n. 1934)
- Cuantificare vectorială: transformarea unui set de vectori mai mare într-un set de vector mai mic, reprezentativ, de prototipuri (codebook vectors, prototypes)
  - ❖ Utilizată original în aplicaţii de comprimare a datelor

- Algoritmul LVQ foloseşte învăţarea supravegheată - pentru fiecare vector din baza de date folosită pentru antrenare se cunoaşte clasa din care face parte, reprezentată printr-o etichetă $c=1 \ldots C$.

- Realizează comprimarea vectorilor modelelor din baza de date de antrenare $x^{(m)}$, $m=1 \ldots M$ în vectorii doriţi $w^{(c)}$, $c=1 \ldots C$ cu $C<M$.

Algoritmul LVQ - funcționare

- Se identifică din baza de date de antrenare numărul de modele M, numărul de clase C şi dimensiunea unui model de antrenare n.
- Se iniţializează aleatoriu o matrice W cu n linii şi C coloane; fiecare coloană din această matrice reprezintă prototipul iniţial al clasei c
- Repetă până la îndeplinirea unui criteriu de oprire:
  - Pentru fiecare model de intrare m, $m=1..M$
    - Calculează distanţa euclidiană între model şi ponderile fiecărui neuron-clasă
    - Alege drept neuron câştigător pe cel cu distanţa calculată cea mai mică
    - Dacă numărul clasei alese coincide cu cel asociat modelului în baza de date, adaptează ponderile neuronului-clasă respectiv în sensul apropierii lui de modelul de intrare
    - Dacă numărul clasei alese coincide cu cel asociat modelului în baza de date, adaptează ponderile neuronului-clasă respectiv în sensul îndepărtării lui de modelul de intrare

Algoritmul LVQ - implementare (1/2)

- Se dau:
- $[X]$ - matricea modelelor de intrare, cu M linii şi n coloane unde M este numărul de modele din setul de antrenare, iar n este numărul de elemente care formează un model de intrare. Fiecare element i din setul de antrenare este codificat într-o linie din matricea $[X]$, $i=1...M$.
- $[CX]$ - vector coloană cu M elemente, în care fiecare element $CX(i)$ este un număr întreg c, $c=1..C$, reprezentând clasa căruia îi este asociat elementul i din matricea X, $i=1...M$.
- O rată de învăţare η (eta).

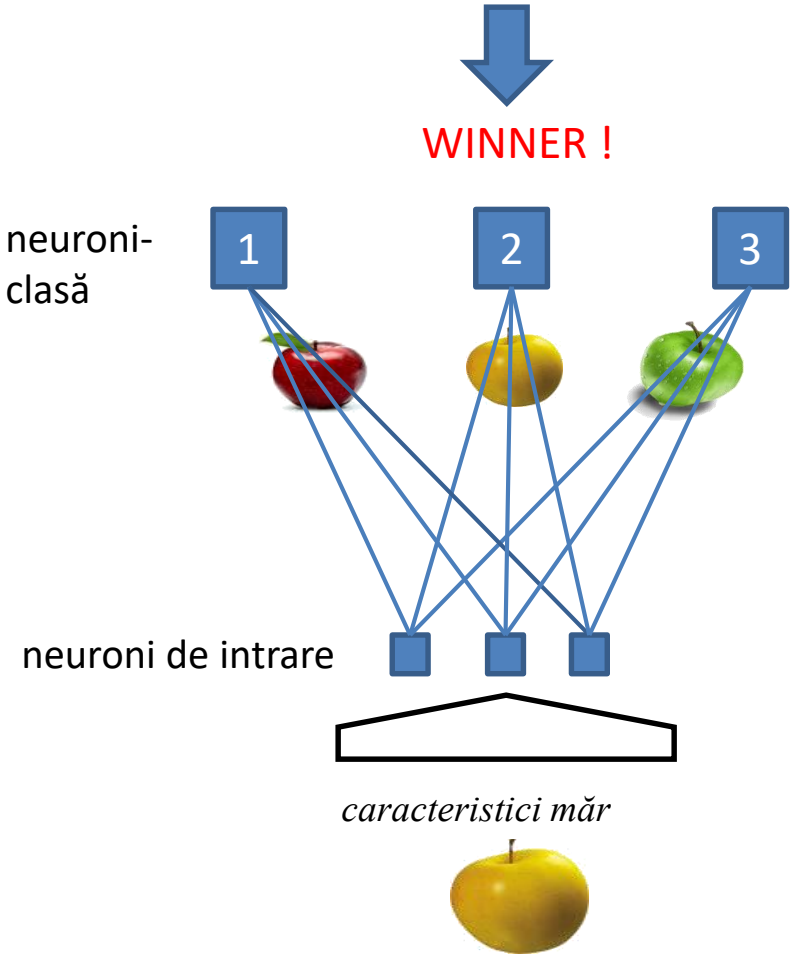
- Se defineşte criteriul de oprire: de exemplu, un număr maxim de iteraţii $MaxIt$.
- Se iniţializează aleatoriu o matrice W cu C linii şi n coloane; fiecare linie din această matrice reprezintă prototipul iniţial al fiecărei clase c.

Algoritmul LVQ - implementare (2/2)

```
for i=1 : MaxIt    % Începe o nouă iteraţie
   for m=1 : M % pentru fiecare model din setul de date de antrenare
        % citeşte model din matricea X
        model=X(m, :);
        % calculează distanţa dintre model şi ponderile neuronilor-clasă
        D=zeros(1,C);
        for c=1:C
                prototip=W(c,:); % extrage prototipul curent
                D(1,c)= sqrt(sum((model - prototip) .^ 2));
        end; % for c prototipuri
        % identifică neuronul-clasă câştigător
        winn=find(D==min(D));
        % adaptează ponderi neuron câştigător
        if winn==CX(m)
            % dacă clasa identificată este cea dorită, adaptare pozitivă
             W(winn,:)= W(winn,:)+eta*(model- W(winn,:))
        else
            % dacă clasa identificată este greşită, adaptare negativă
             W(winn,:)= W(winn,:)-eta*(model- W(winn,:))
        end; % if winn
   end; % for m modele
end; % for i iteraţii
```

Algoritmul LVQ - interpretare intuitivă

Câte feluri de mere sunt în grămadă ?
Cum arată fiecare fel de măr ?
Care măr aparţine cărui fel ?



Algoritmul LVQ – avantaje şi dezavantaje

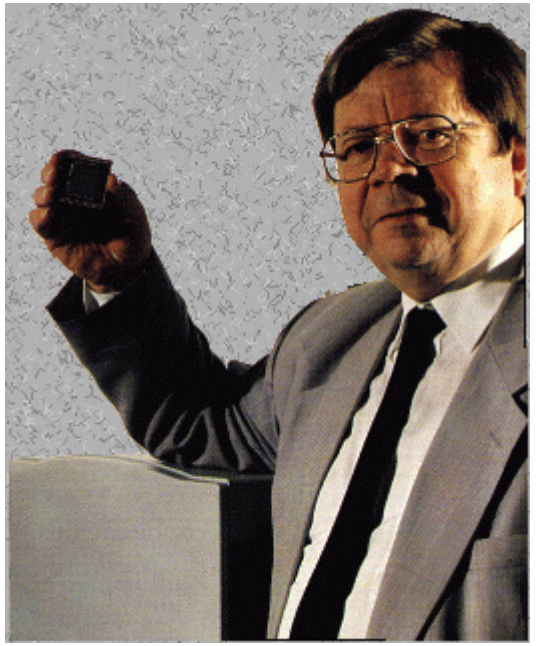
- îmbunătăţeşte performanţele algoritmului k-medii

- foloseşte un număr fix de clase
- învăţare supravegheată - în realitate, nu se cunoaşte întotdeauna împărţirea pe clase a elementelor din baza de date folosită la antrenare
  - Aceste dezavantaje pot fi remediate folosind RNA de tip SOFM

Rețele neuronale Kohonen
(Self-organizing Feature Maps - SOFM)

Istoric

- Inventate de profesorul finlandez Teuvo Kohonen (n. 1934)



RNA Kohonen – inspirația biologică

- Biologic, învăţarea sau deprinderea se realizează prin întărirea anumitor sinapse (legături) dintre anumiţi neuroni, formând căi sinaptice.

- Creierul copilului este o pânză albă; reţelele de neuroni din anumite zone ale cortexului formează centrii nervoşi prin antrenare (exersare).

- Cortexul arată ca o pânză întinsă pe suprafaţa creierului

- Relaţiile dintre stimulii exteriori se regăsesc în relaţii similare între neuroni
- Dispunerea centrilor nervoşi în creier respectă principiul vecinătăţii: mână lângă braţ.

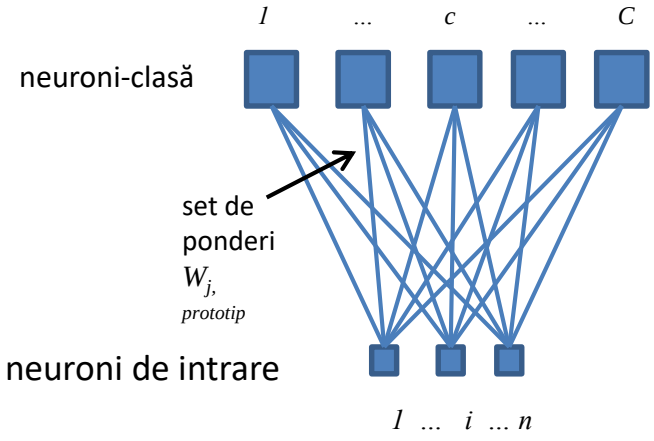
RNA Kohonen – principii de funcționare

- SOFM = Self-Organizing Feature Maps (hărţi de trăsături cu autoorganizare)
  - Autoorganizare: capacitatea unui sistem de a descoperi şi învăţa structura datelor de intrare în absenţa oricăror informaţii privitoare la această structură. RNA mapează intrările către ieşiri.
  - RNA cu autoorganizare învaţă trăsăturile (caracteristicile) modelelor de intrare în absenţa unor corespondenţe cunoscute cu datele de ieşire (învăţare nesupravegheată)
- Harta presupune organizarea topologică a informaţiei şi vecinătăţi.

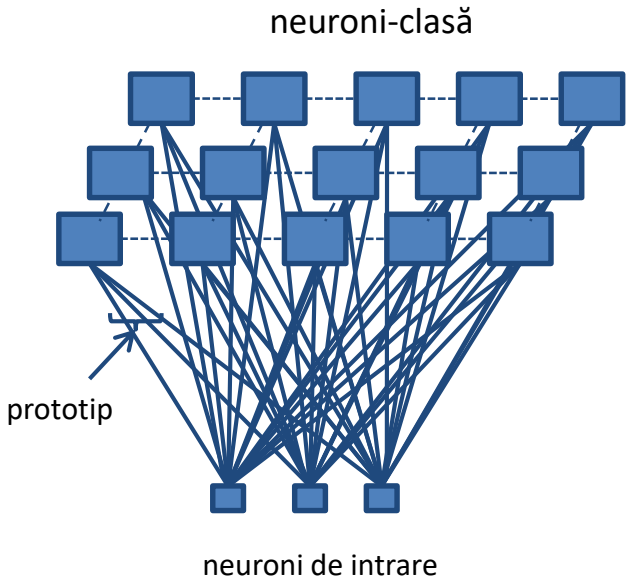
RNA Kohonen – arhitectură

- RNA Kohonen extinde RNA generală clasificatoare adăugând conceptele de organizare topologică şi vecinătăţi.

- Neuronii-clasă sunt organizaţi în grile uni- bi- sau multidimensionale



grilă unidimensională



grilă bidimensională

RNA Kohonen - algoritmul fără creşterea rețelei

- Etapa de antrenare:
  - Pregătire bază de date de modele de intrare pentru antrenare
  - Iniţializare număr de clase (de prototipuri)
  - Iniţializare ponderi prototipuri, de obicei în intervalul [0,1]
  - Proces iterativ:
    - Fiecare model de intrare este comparat, prin calculul unei distanţe (de obicei cea euclidiană) cu prototipurile tuturor neuronilor-clasă
    - Este ales drept neuron câştigător cel cu distanţa calculată cea mai mică
    - Se calculează vecinătatea neuronului-clasă câştigător şi se reduce valoarea ratei de învăţare
    - Ponderile neuronului clasă ales şi ale celor aflate în vecinătatea sa sunt corectate pentru a se apropia de modelul de intrare respectiv
    - Dacă s-a îndeplinit condiţia de oprire, procesul iterativ se opreşte
  - În finalul algoritmului, valorile ponderilor fiecărui neuron-clasă codifică caracteristicile clasei respective.

- Etapa de recunoaştere:
  - Pentru fiecare model de intrare nou, care nu a fost folosit în etapa de antrenare, se calculează distanţa până la fiecare neuron-clasă, la fel ca în etapa de antrenare
  - Este aleasă drept clasă câştigătoare cea cu distanţa calculată cea mai mică

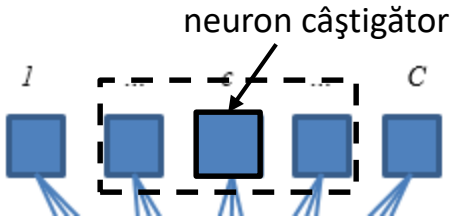
RNA Kohonen - vecinătăți

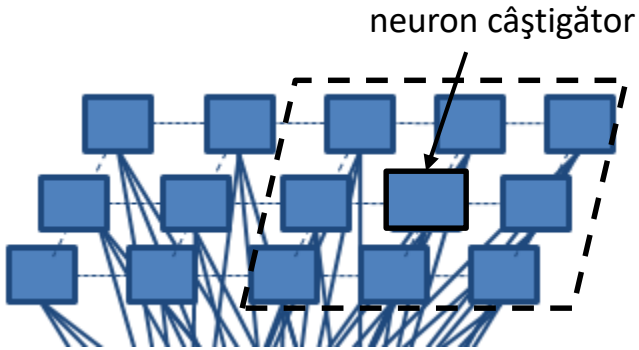
- Vecinătăţi continue: se foloseşte o funcţie continuă, de obicei gaussiană:

$$V_c = e^{-\rho_c^2/2\sigma^2}, \; unde \; \rho_c = \left| P_c - P_{c^*} \right|$$

  - σ^2 - *dispersie*, ρ_c -distanţa dintre neuronul-clasă *c* şi neuronul câştigător *c**

- Vecinătăţi discrete, care folosesc poziţiile neuronilor în grilă





RNA Kohonen - vecinătăți

- În prima etapă a antrenării, vecinătatea are valoare mare, pentru a încuraja colaborarea sau cooperarea între neuroni.
- Cu cât dispersia σ este aleasă mai mare, cu atât vecinătatea inițială vă fi mai mare
- Treptat, vecinătatea este îngustată, pentru ca spre finalul antrenării să devină 0 (sunt adaptate doar ponderile neuronului-clasă câştigător)

- Acelaşi principiu al descreşterii se aplică, de obicei, şi pentru rata de învăţare η.

RNA Kohonen - adaptarea ponderilor

- Adaptarea ponderilor:

  - unidirecţională

$$w_c^{(t+1)} = w_c^{(t)} + \eta \cdot (x^{(m)} - w_c^{(t)}), \quad daca \quad c \in V_c \quad (stimulare)$$
$$w_c^{(t+1)} = w_c^{(t)}, \quad daca \quad c \notin V_c \quad (ponderile\ raman\ neschimbate)$$

  - bidirecţională

$$w_c^{(t+1)} = w_c^{(t)} + \eta \cdot (x^{(m)} - w_c^{(t)}), \quad daca \quad c \in V_c \quad (stimulare)$$
$$w_c^{(t+1)} = w_c^{(t)} - \eta \cdot (x^{(m)} - w_c^{(t)}), \quad daca \quad c \notin V_c \quad (penalizare)$$

RNA Kohonen - algoritmul cu creşterea reţelei

- RNA Kohonen permite creşterea reţelei, plecând de la o singură clasă iniţială şi creşterea numărului de clase pe măsură ce se identifică modele de intrare cu caracteristici noi.

- Criteriu de departajare pentru crearea unei clase noi: o valoare maximă a distanţei calculate între modelul de intrare respectiv şi ponderile fiecărui neuron-clasă existent în reţea

RNA Kohonen - algoritmul cu creşterea rețelei

- Algoritm de creştere pentru o grilă de neuroni pătrată [Engelbrecht]:

- Se identifică neuronul-clasă *ij* faţă de care distanţa calculată este cea mai mare
- Se identifică, pe linii şi pe coloane, cei mai îndepărtaţi neuroni-clasă vecini, *mn* (linie) şi *rp* (coloană)
- Se introduce o linie nouă între linia *ij* şi linia *mn*
- Se introduce o colană nouă între coloana *ij* şi coloana *rp*
- Se iniţializează ponderile pe noua linie şi coloană:

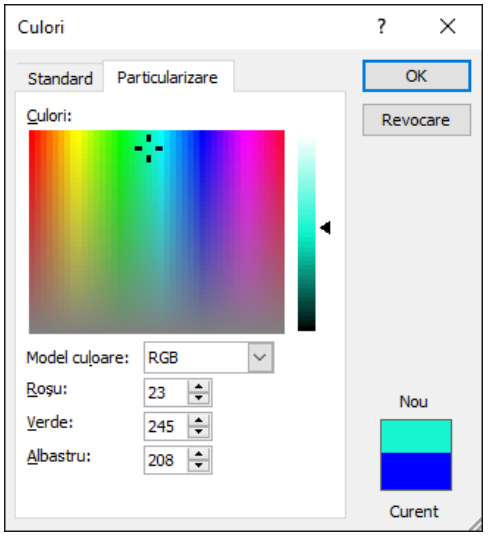
$$w_{x,colnou} = \alpha \cdot (w_{x,colnou-1} + w_{x,colnou+1})$$

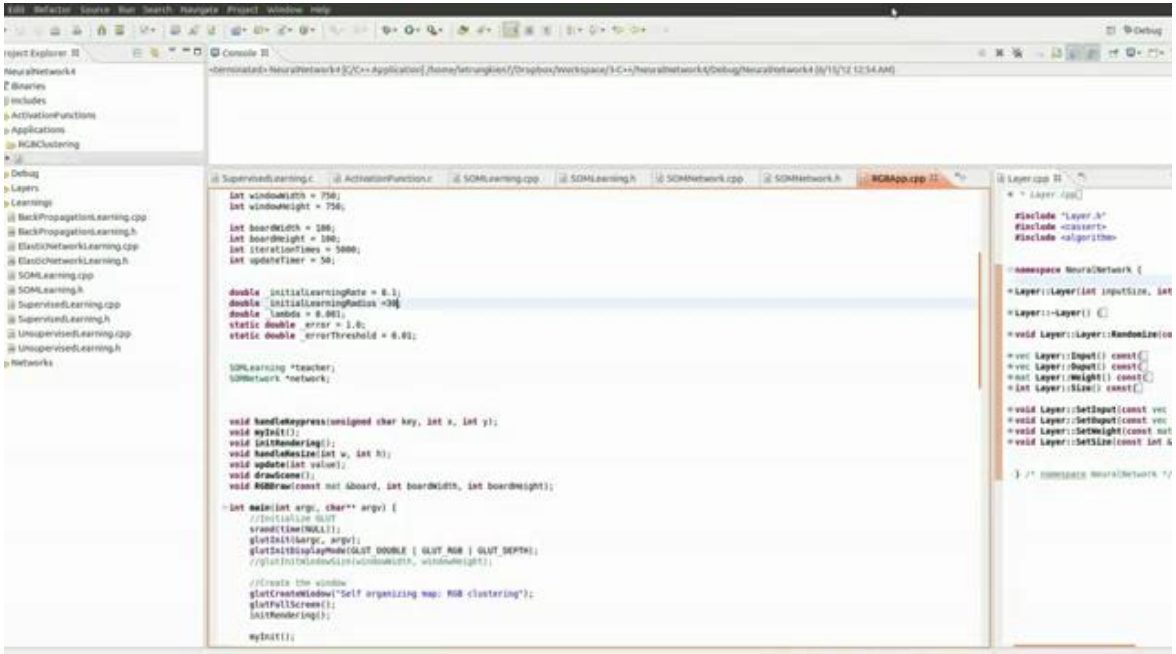
$$w_{linnou,y} = \alpha \cdot (w_{linnou-1,y} + w_{linnou+1,y})$$

unde x/y iau valori între 1 şi indexul maxim de linie/coloană şi $\alpha \in (0,1)$

Intuiție grafică a antrenării unei RNA Kohonen

- Identificarea culorilor după codurile RGB
- Red : 0-256
- Green: 0-256
- Blue : 0-256





https://www.youtube.com/watch?v=ENLLPg9cb2M

va urma...

- examen

Vă mulțumesc pentru atenție !