

# Метод                   штучних нейронних мереж



# Класифікація нейромереж:

## ■ *за напрямом зв'язку*

### ➤ *без оберненого зв'язку*

➤ *мережі з оберненим розповсюдженням помилки;*

➤ *інші мережі (когнітрон, неокогнітрон, інші складні моделі)*

### ➤ *з оберненим зв'язком*

➤ *мережі Хопфілда (задачі асоціативної пам'яті);*

➤ *мережі Кохонена (задачі кластерного аналізу)*

# Класифікація нейромереж:

- мережі прямого розповсюдження
  - *персептрони;*
  - *мережа Back Propagation;*
  - *мережа зустрічного розповсюдження;*
  - *карта Кохонена*
- рекурентні мережі
  - *мережа Хопфілда;*
  - *мережа Елмана*



# Підготовка даних до навчання:

- *кількість спостережень у наборі;*
  - *робота із викидами;*
  - *репрезентативність навчальної вибірки;*
  - *навчальна вибірка без протиріч;*
  - *робота тільки із числовими даними.*
- При використанні на вхід нейронної мережі слід подавати значення із того ж діапазону, на якому нейронна мережа навчалась.
- Нормалізація даних.

# Принципи, якими необхідно керуватись при розробці нової конфігурації нейронної мережі:

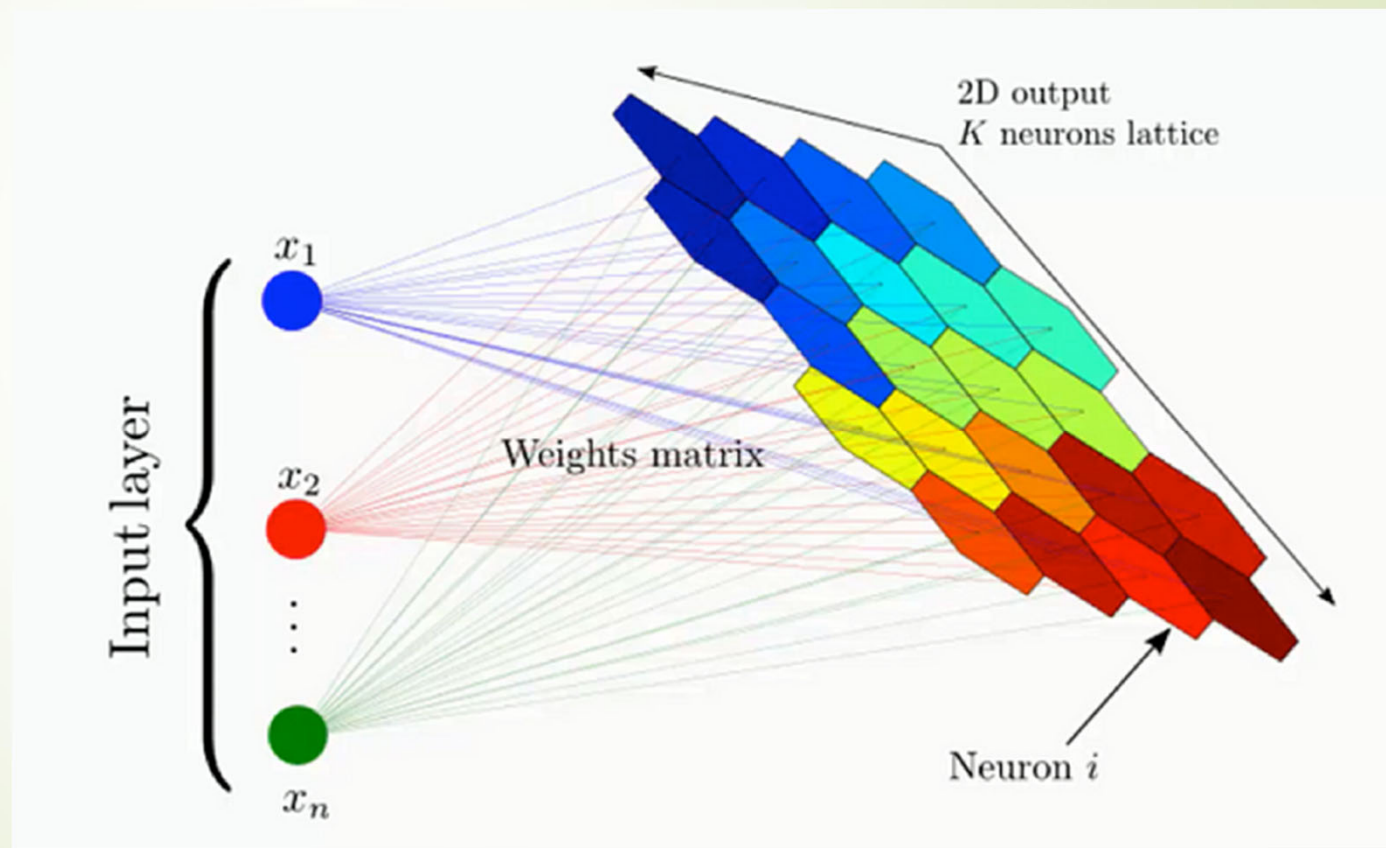
- *можливості мережі зростають зі зростанням числа комірок мережі, щільності зв'язків між ними та числом виділених шарів;*
- *введення зворотних зв'язків разом із збільшенням можливостей мережі підіймає питання про динамічну стійкість мережі;*
- *складність алгоритмів функціонування мережі також сприяє підсиленню потужності мережі.*

# Карти, що самоорганізуються (Self-Organizing Maps, SOM)

*Карти Кохонена*



Teuvo Kohonen





# Фази нейронної мережі Кохонена

In the training phase, each input vector  $X_s$  is input into the network, and only those winning neurons closest to the current weight vector of the input receive a corresponding stimulus. The pattern vector  $X_s$  is calculated as the minimum Euclidean distance from the selected winning neurons:

$$neuron c \leftarrow \min_j \left\{ \sum_i (x_{si} - w_{ji})^2 \right\}, \quad j = 1, 2, \dots, N \times N \quad (1)$$

where  $c$  represents the winning neuron and  $x_{si}$  represents the  $i$ th coordinate of the input vector. In addition, the level of the  $i$ th weight of neuron  $j$  is denoted by  $w_{ji}$ . The number of neurons in a Kohonen level is denoted by  $N \times N$ . Once the winning neuron is selected, the corresponding weight  $w_{ji}$  of each neuron  $j$  in the layer is updated according to the difference between the original weight and the input neuron, as follows:

$$\Delta w_{ji} = \eta \left( 1 - \frac{d_r}{d_{max} + 1} \right) (x_{si} - w_{ji}^{old}), \quad d_r = 0, 1, \dots, d_{max} \quad (2)$$

where the learning rate is  $\eta$ , the weight of the previous generation of  $w_{ji}$  is  $w_{ji}^{old}$ , and the number of neurons between neuron  $j$  and the superior neuron is represented by the topological distance  $d_r$ . The size of the adjacent area  $d_{max}$  decreases from the coverage of the entire network to the winning neurons as training progresses. In addition, the learning rate  $\eta$  changes during training:

$$\eta = (\eta^{start} - \eta^{final}) \left( 1 - \frac{n_{epoch}}{n_{tot}} \right) + \eta^{final} \quad (3)$$

where  $n_{tot}$  represents the total number of iterations;  $n_{epoch}$  represents the current iteration times.

# Модифікація SOM алгоритму

Мелссен та ін. (2006) запропонували більш гнучкий підхід, в якому відстані в просторі X та Y обчислюються окремо. Обидві відстані масштабуються так, що максимальна відстань дорівнює 1, а загальна відстань є зваженою сумою обох:

$$D(o, u) = \alpha D_x(o, u) + (1 - \alpha) D_y(o, u) ,$$

де  $D(o, u)$  вказує на загальну відстань об'єкта  $o$  до одиниці  $u$ , а  $D_x$  і  $D_y$  вказують на відстані в окремих просторах.

Навчання карти відбувається у звичайному режимі, при цьому елемент-переможець та його околиця оновлюються, а швидкість навчання та розмір околиці зменшуються. Кінцевий результат складається з двох карт: одна карта для змінних X, інша для змінних Y. Для керованих SOM один екстра параметр, вага для простору X (або Y), має бути визначений користувачем. Цей принцип може бути поширений і на більшу кількість шарів; у такому випадку ми називаємо результат суперорганізованою картою. Для кожного шару обчислюється значення схожості, і всі окремі значення схожості потім об'єднуються в одне значення, яке використовується для визначення одиниці-переможця:

$$D(o, u) = \sum_i \alpha_i D_i(o, u) ,$$

де ваги  $\alpha_i$  масштабуються до одиничної суми. Ці ваги є єдиними додатковими параметрами (порівняно з класичними методами SOM), які повинні бути визначені користувачем.

Melssen WJ, Wehrens R, Buydens LMC (2006). "Supervised Kohonen Networks for Classification Problems." *Chemometrics and Intelligent Laboratory Systems*, 83, 99–113.



# The kohonen package for R

Function name	Short description
<code>som</code>	standard SOM
<code>xyf</code>	supervised SOM: two parallel maps
<code>bdk</code>	supervised SOM: two parallel maps (alternative formulation)
<code>supersom</code>	SOM with multiple parallel maps
<code>plot.kohonen</code>	generic plotting function
<code>summary.kohonen</code>	generic summary function
<code>map.kohonen</code>	map data to the most similar unit
<code>predict.kohonen</code>	generic function to predict properties
<code>wines</code>	wine data: a 177-by-13 matrix
<code>nir</code>	NIR spectra of 95 ternary mixtures
<code>yeast</code>	microarray data of the yeast cell cycle

## *Wine Dataset*

(навчальний репозиторій UCI Machine)

177 зразків італійського вина з трьох різних сортів винограду з 13 атрибутами:

- **алкоголь,**
- **яблучна кислота,**
- **зола,**
- **лужність золи,**
- **магній,**
- **загальні феноли,**
- **флавоноїди,**
- **нефлаваноїдні феноли,**
- **проантоціани,**
- **інтенсивність кольору,**
- **відтінок,**
- **розбавлені вина OD280/OD325,**
- **пролін.**

**а. Перш за все, нам потрібно встановити пакет «kohonen» в RStudio. Це пакет, який надає всі необхідні нам функції та набір даних.**

```
install.packages("kohonen")
```

```
library(kohonen)
```

```
search()
```

```
[1] ".GlobalEnv"      "package:kohonen"  "tools:rstudio"   [4] "package:stats"  
"package:graphics" "package:grDevices" [7] "package:utils"  
"package:datasets" "package:methods" [10] "Autoloads"      "package:base"
```

```
ls(2)
```

```
[1] "add.cluster.boundaries" "check.whatmap"      [3] "classmat2classvec"  
"classvec2classmat"      [5] "dist2WU"           "expandMap"         [7]  
"getCodes"               "layer.distances"   [9] "map"               "nunits"  
[11] "object.distances"      "som"                [13] "somgrid"  
"supersom"               [15] "tricolor"          "unit.distances"    [17] "xyf"
```

в. Ми можемо побачити опис і структуру набору даних Wine за допомогою функцій `head()`, `scale()`, `dim()` і `str()`.

`data(wines)`

`head(wines)`

```
      alcohol malic acid  ash ash alkalinity magnesium tot. phenols
[1,]   13.20      1.78 2.14      11.2      100      2.65
[2,]   13.16      2.36 2.67      18.6      101      2.80
[3,]   14.37      1.95 2.50      16.8      113      3.85
[4,]   13.24      2.59 2.87      21.0      118      2.80
[5,]   14.20      1.76 2.45      15.2      112      3.27
[6,]   14.39      1.87 2.45      14.6       96      2.50
      flavonoids non-flav. phenols proanth col. int. col. hue
[1,]      2.76              0.26   1.28   4.38   1.05
[2,]      3.24              0.30   2.81   5.68   1.03
[3,]      3.49              0.24   2.18   7.80   0.86
[4,]      2.69              0.39   1.82   4.32   1.04
[5,]      3.39              0.34   1.97   6.75   1.05
[6,]      2.52              0.30   1.98   5.25   1.02
      OD ratio proline
[1,]      3.40   1050
[2,]      3.17   1185
[3,]      3.45   1480
[4,]      2.93    735
[5,]      2.85   1450
[6,]      3.58   1290
```

в. Ми можемо побачити опис і структуру набору даних Wine за допомогою функцій `head()`, `scale()`, `dim()` і `str()`.

`scale(wines)`

#нормування значень

`head(scale(wines))`

```
      alcohol  malic acid      ash  ash alkalinity  magnesium
[1,] 0.2551008 -0.50020530 -0.8221529    -2.4930372  0.02909756
[2,] 0.2056453  0.01796903  1.1045562    -0.2748590  0.09964918
[3,] 1.7016732 -0.34832662  0.4865552    -0.8144158  0.94626865
[4,] 0.3045563  0.22345196  1.8316163     0.4445501  1.29902677
[5,] 1.4914875 -0.51807338  0.3047901    -1.2940219  0.87571703
[6,] 1.7264010 -0.41979894  0.3047901    -1.4738742 -0.25310893
      tot. phenols flavonoids non-flav. phenols  proanth
[1,]  0.5710456  0.7375437    -0.8208101 -0.5370519
[2,]  0.8104843  1.2181890    -0.4999191  2.1399040
[3,]  2.4865554  1.4685250    -0.9812556  1.0376281
[4,]  0.8104843  0.6674496     0.2220856  0.4077561
[5,]  1.5607256  1.3683906    -0.1790282  0.6702028
[6,]  0.3316069  0.4972211    -0.4999191  0.6876992
      col. int.  col. hue  od ratio  proline
[1,] -0.29030665  0.4059482  1.1284966  0.96830550
[2,]  0.26896630  0.3186634  0.8023031  1.39703475
[3,]  1.18101141 -0.4232572  1.1994082  2.33388755
[4,] -0.31611925  0.3623058  0.4619272 -0.03206274
[5,]  0.72929095  0.4059482  0.3484686  2.23861439
[6,]  0.08397601  0.2750210  1.3837785  1.73049083
```



**в. Ми можемо побачити опис і структуру набору даних Wine за допомогою функцій `head()`, `scale()`, `dim()` і `str()`.**

```
dim(wines)  
[1] 177 13
```

```
str(wines)
```

```
num [1:177, 1:13] 13.2 13.2 14.4 13.2 14.2 ...
```

```
- attr(*, "dimnames")=List of 2 ..
```

```
- $ : NULL ..
```

```
- $ : chr [1:13] "alcohol" "malic acid" "ash" "ash alkalinity " ...
```

с. Потім створіть змінну з назвою «grid», яка містить дані з розмірами 5x5 для створення самоорганізаційних карт з гексагональною топологією.

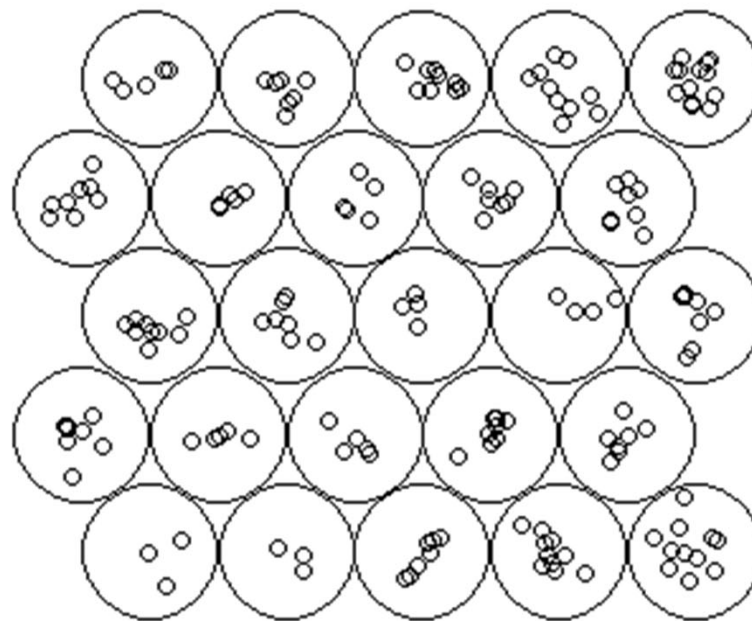
```
grid <- somgrid(xdim = 5, ydim = 5, topo = "hexagonal")
```

Потім виконайте команду самоорганізації карт зі змінною з ім'ям *som.wines*

```
som.wines <- som(scale(wines), grid = somgrid(xdim = 5, ydim = 5, "hexagonal"))  
str(som.wines)  
plot(som.wines, type = "mapping") #display the SOM result plot
```

с. Потім створіть змінну з назвою «grid», яка містить дані з розмірами 5x5 для створення самоорганізаційних карт з гексагональною топологією.

Mapping plot



d. Наступним кроком є з'ясування того, які члени є у вершині 1 і т.д.

```
> som.wines$grid$pts
```

```
      x      y  
[1,] 1.5 0.8660254  
[2,] 2.5 0.8660254  
[3,] 3.5 0.8660254  
[4,] 4.5 0.8660254  
[5,] 5.5 0.8660254  
[6,] 1.0 1.7320508  
[7,] 2.0 1.7320508  
[8,] 3.0 1.7320508  
[9,] 4.0 1.7320508  
[10,] 5.0 1.7320508  
[11,] 1.5 2.5980762  
[12,] 2.5 2.5980762  
[13,] 3.5 2.5980762  
[14,] 4.5 2.5980762  
[15,] 5.5 2.5980762  
[16,] 1.0 3.4641016  
[17,] 2.0 3.4641016  
[18,] 3.0 3.4641016  
[19,] 4.0 3.4641016  
[20,] 5.0 3.4641016  
[21,] 1.5 4.3301270  
[22,] 2.5 4.3301270  
[23,] 3.5 4.3301270  
[24,] 4.5 4.3301270  
[25,] 5.5 4.3301270
```

Об'єкти, які потрапляють у 25 кіл, можна побачити за допомогою команди

```
> som.wines$unit.classif
 [1]  9  7  2  6  2  3  1  3  3  2  9  3  3  2  1  1  1  2  8  8  4  9  9  9  6  9  9  1  3
[30]  7  2  9  1  1  9  1  9  9  8  8  4  8  4  8  4  8  3  7  2  3  3  2  1  8  7  8  1  2
[59] 19 23 23 19 10 25 10 14 14 23 13 18 10 20  6 10 19 14 18 13  5 14 14 25 22 10 14 25 25
[88] 24 20 24 24 24 15 10 13 12 14 10  5 14 19 20 15 15 24 15 24 15 10  5 20 25 25 20 25 15
[117] 20 19 15 10  6 20  5  5 20 20 25 20 20 18 17 17 17 23 23 22 22 21 22 22 17 22 22 17 17
[146] 21 21 21 17 11 11 11 16 23 21 21 22 16 16 21 22 22 17 21 21 16 21 11 11 17 21 16 21 21
[175] 16 16 16
```

Потім хочемо побачити загальний графік `som.wines`

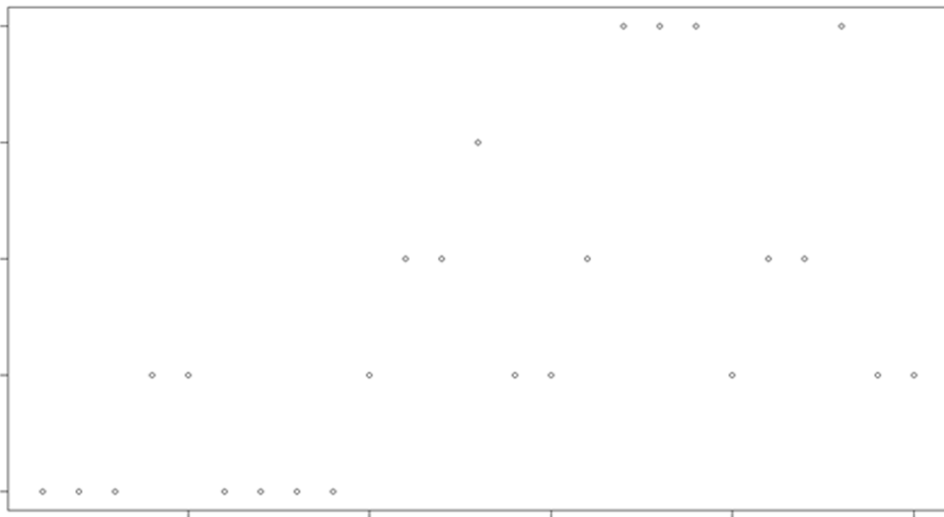
```
text(som.wines$grid$pts, labels = som.wines$unit.classif,
     cex = 1.5)
str(som.wines)
som.wines$codes[[1]]
hclust(dist(som.wines$codes[[1]]))
peta = cutree(hclust(dist(som.wines$codes[[1]])), 5)
plot(peta)
plot(som.wines, type = "codes", bgcol = rainbow(5)
     [peta])
```



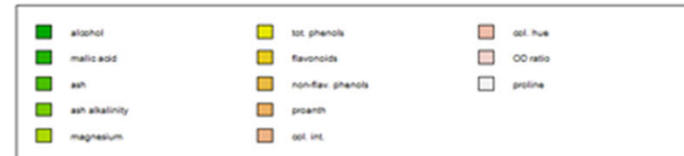
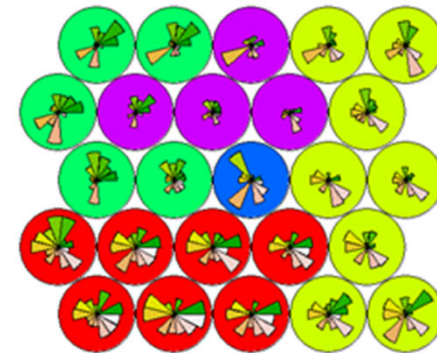
```
> hclust(dist(som.wines$codes[[1]]))
```

```
Call:  
hclust(d = dist(som.wines$codes[[1]]))
```

```
Cluster method : complete  
Distance       : euclidean  
Number of objects: 25
```



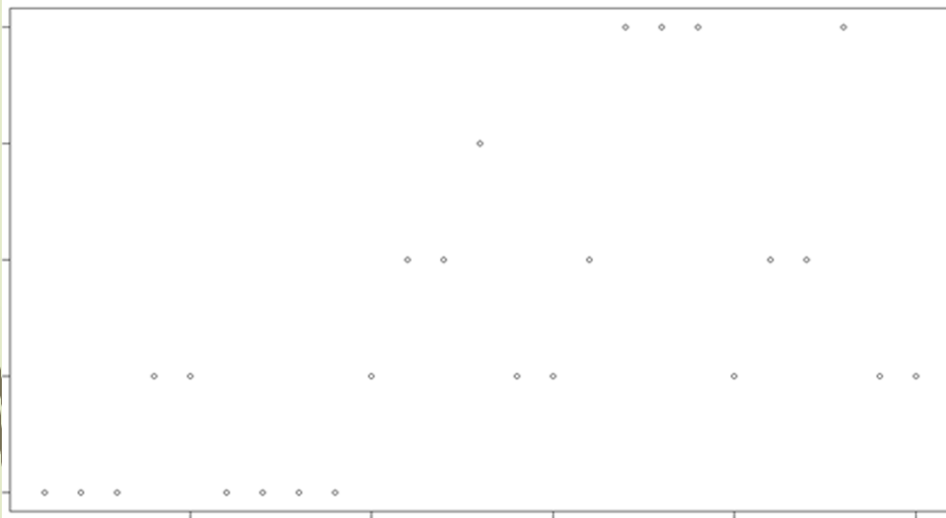
Codes plot



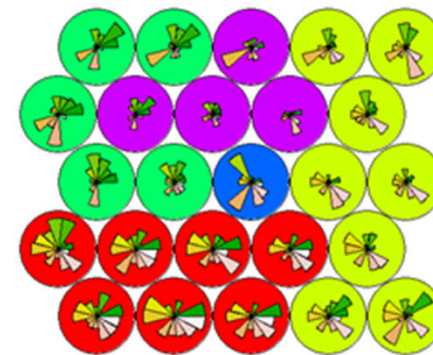
```
> hclust(dist(som.wines$codes[[1]]))
```

```
Call:  
hclust(d = dist(som.wines$codes[[1]]))
```

```
Cluster method : complete  
Distance       : euclidean  
Number of objects: 25
```



Codes plot



*add.cluster.boundaries(som.wines,peta)*

Codes plot

