

Kohonen Maps

Application au MNIST

```
In[3]:= mnistData = ExampleData[{"MachineLearning", "MNIST"}, "Data"];
          [données d'exemples

In[4]:= maxDigits = 10;

In[5]:= GetDataFromMNIST[] := Module[{i, dataToPCA = {}, images, dataGroup},
          [module
          For[i = 0, i < maxDigits, i++,
          [pour chaque
            images = #[[1]] & /@ Select[mnistData, #[[2]] == i &];
            [sélectionne
            dataGroup = Flatten /@ ImageData /@ images;
            [aplatis      [données d'image
            dataToPCA = Join[dataToPCA, dataGroup];
            [joins
          ];
          dataToPCA
        ]
      ]

In[6]:= dataToPCA = GetDataFromMNIST[];
```

Kohonen

■ Initialisation des représentants de la map

```
In[22]:= weights = Table[{i, j, RandomReal[1, 784]}, {i, 1, 10}, {j, 1, 10}] // Flatten[#, 1] &;
          [table           [nombre réel aléatoire                  [aplatis
```

■ Apprentissage (attention au nombre d'itération, ça peut être long ...)

```
In[61]:= weights =
Module[{i, t, bmu, shuffleData, updatedWeights},
Module
Print[Dynamic[i]];

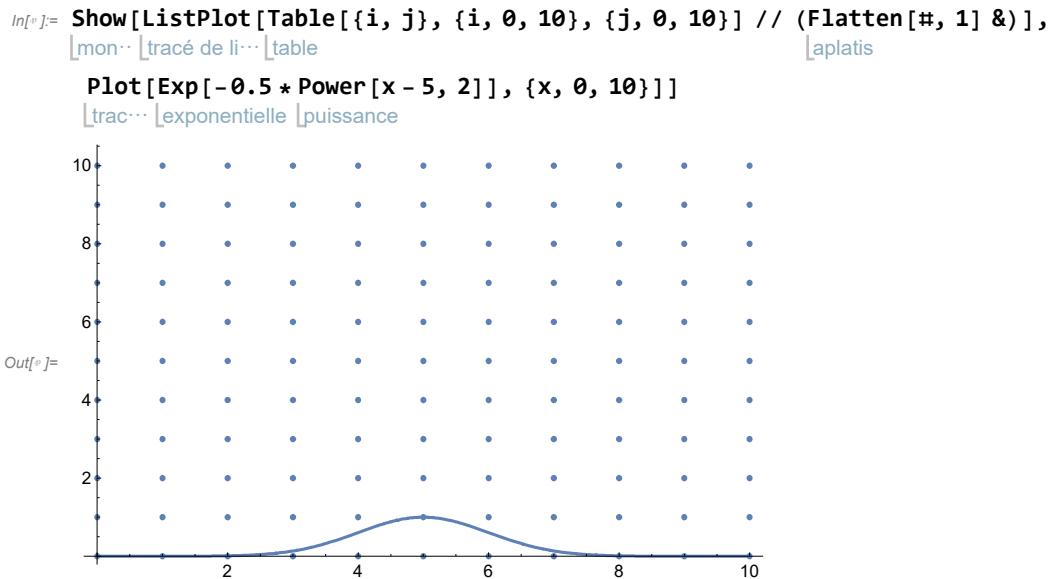
updatedWeights = weights;
For[i = 0, i < 50000, i++,
For chaque
shuffleData = RandomSample[dataToPCA, 200];
échantillon aléatoire
For[t = 1, t ≤ Length[shuffleData[[1 ;; 100]]], t++,
longueur
bmu = MinimalBy[updatedWeights, Norm[#[[3]] - shuffleData[[t]], 2] &, 1][[1]];
minimum pour norme
updatedWeights =
((# + {0, 0, (Exp[-0.5 * Power[Norm[#[[1 ;; 2]] - bmu[[1 ;; 2]], 2], 2], 2]) *
exponentielle puiss... norme
0.1 * (shuffleData[[t]] - #[[3]]))) & /@ updatedWeights);
];
];
updatedWeights
]

50000

Out[61]=
{{1, 1, {1., 1., 1., 1., 1., 1., 1., 1., 1., 1., ..., 764 ...},
1., 1., 1., 1., 1., 1., 1., 1., 1., 1.}}, ..., 98 ..., ..., 1 ...}}
```

large output | show less | **show more** | show all | set size limit...

■ Représentation de la grille et de l'impact de la fonction de voisinage



■ Préparation de la visualisation de la projection des points dans l'espace des représentants

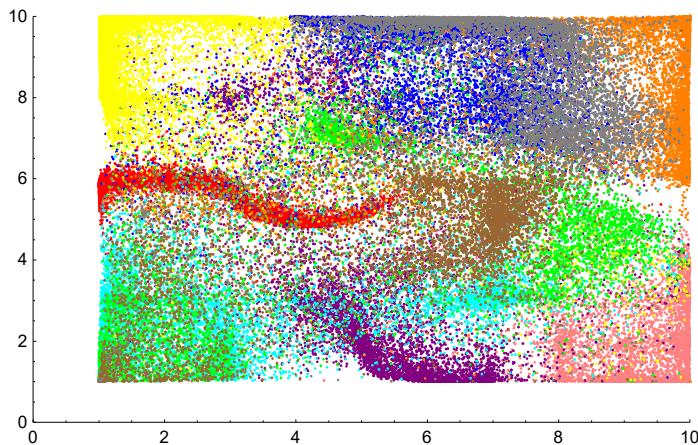
```
In[26]:= classesCount = Table[Length[Select[mnistData, #[[2]] == i &]], {i, 0, 9}];  
In[27]:= classesStart = Table[Total[classesCount[[1 ;; i]]], {i, 0, 9}];  
In[28]:= classesEnd = Table[classesStart[[i + 1]] + classesCount[[i + 1]], {i, 0, 9}];  
In[29]:= colors = {Pink, Red, Purple, Cyan, Blue, Green, Yellow, Orange, Brown, Gray};  
Out[29]=
```

■ Projection des points dans l'espace des représentants

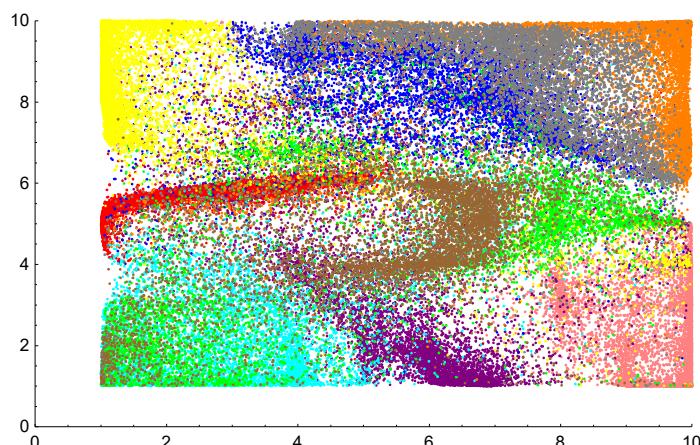
```
In[62]:= Show[Table[ListPlot[Table[Total[bests = Table[{w[[1]], w[[2]], Exp[-Min[0.5 * Power[Norm[elt - w[[3]]], 2], 2], 2], 200]}], {w, weights}]; bests * bests[[;; , 3]] / Total[bests[[;; , 3]]]], {elt, dataToPCA[[classesStart[[i + 1]] + 1 ;; classesEnd[[i + 1]]]]]}[[;; , 1 ;; 2]], PlotStyle -> {colors[[i + 1]]}, PlotRange -> {{0, 10}, {0, 10}}]], {i, 0, 9}]]
```

Out[62]=

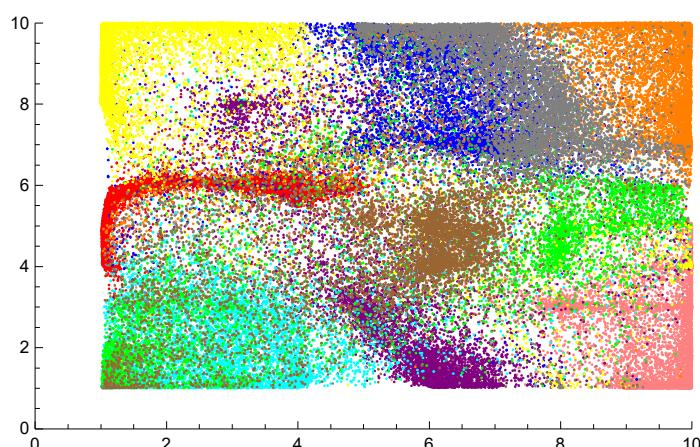
Iteration 100 x batch 200



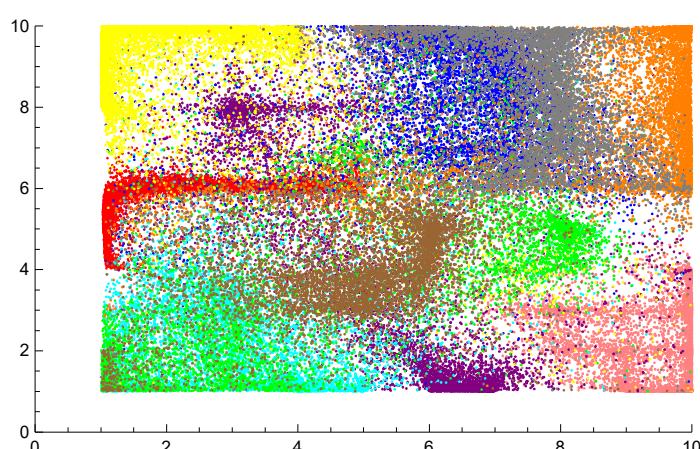
Iteration 200 x batch 200



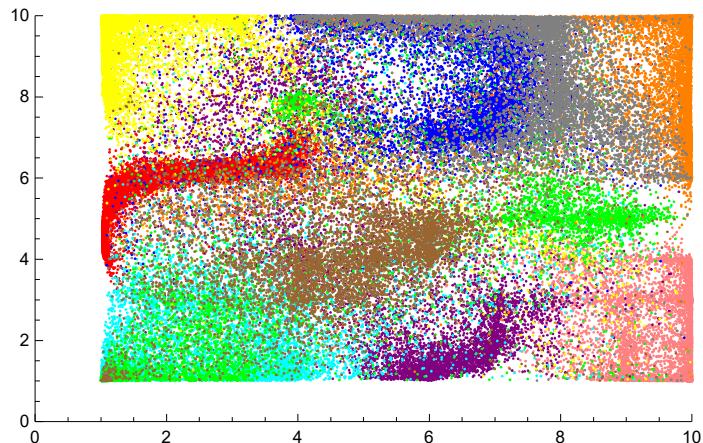
Iteration 500 x batch 200



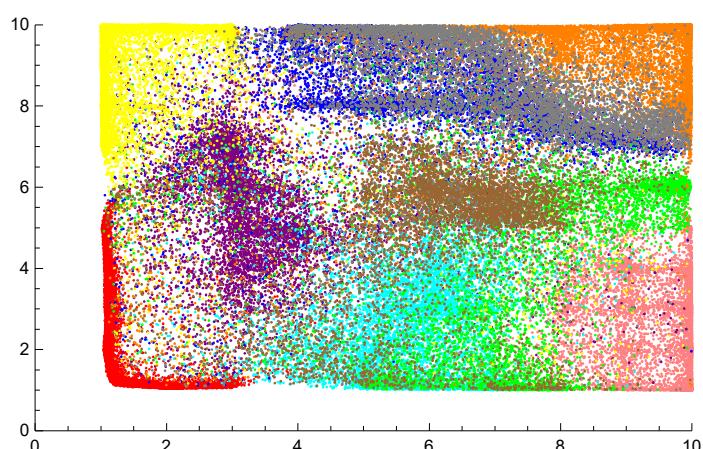
Iteration 1000 x batch 200



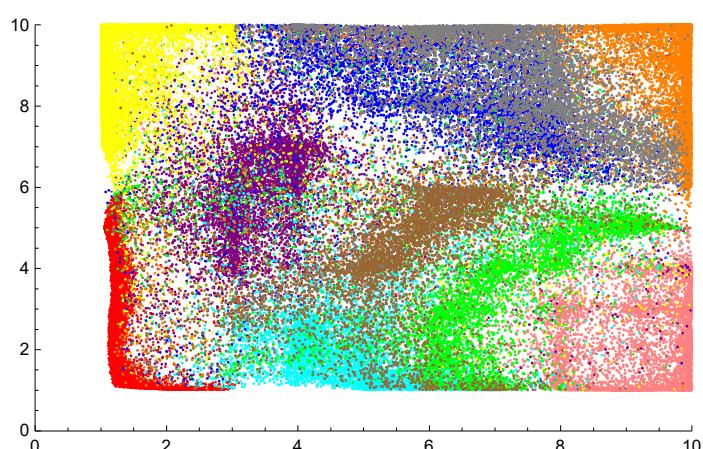
Iteration 2000 x batch 200



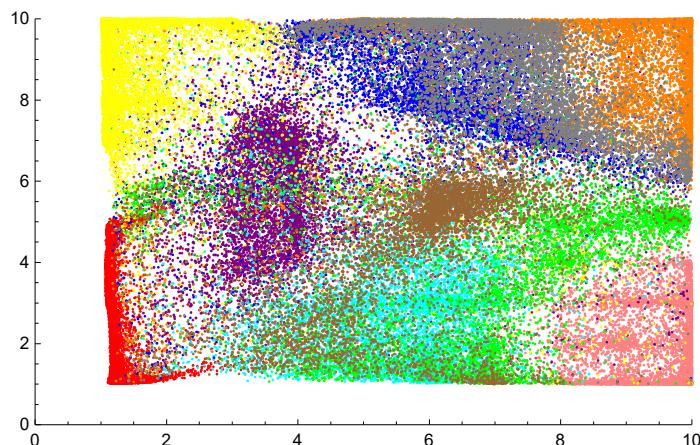
Iteration 10000 x batch 200



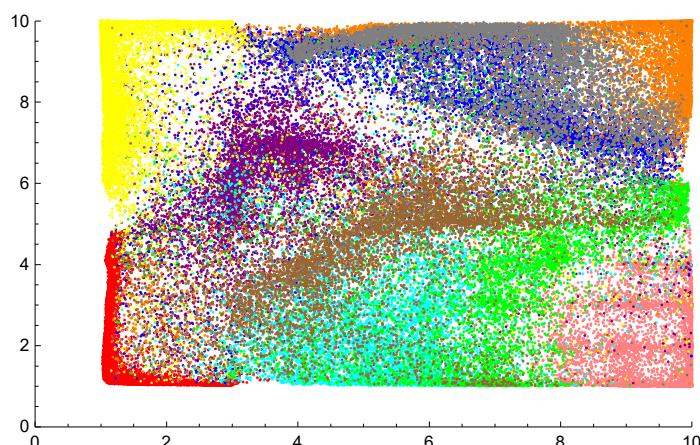
Iteration 20000 x batch 200



Iteration 50000 x batch 200



Iteration 100000 x batch 200



■ Affichage sous forme d'image des représentants de la SOM :

```
In[63]:= ArrayReshape[Image /@ (ArrayReshape[#, {28, 28}] & /@ weights[[;; , 3]]), {10, 10}] //  
  redimensionne ... Image    redimensionne tableau  
Transpose // Reverse // MatrixForm  
  transposée   renversé   apparence matricielle
```

Out[63]//MatrixForm=

6	6	6	9	9	9	9	9	7	7
6	6	6	9	9	9	9	9	7	7
6	6	6	9	9	9	9	9	7	7
6	6	2	2	2	8	9	9	7	7
6	8	2	2	2	8	8	8	8	8
1	2	2	8	8	8	8	8	8	8
1	2	8	8	8	8	8	8	0	0
1	2	8	8	3	5	5	0	0	0
1	1	3	3	3	5	5	0	0	0
1	1	1	3	3	5	5	0	0	0

Iteration 100 x batch 200

6	6	6	9	9	9	9	9	7	7
6	6	6	9	9	9	9	9	7	7
6	4	4	8	8	9	9	9	7	7
6	4	4	8	8	9	9	9	7	7
1	1	1	1	8	8	8	8	7	7
1	1	1	1	2	2	8	8	5	5
3	3	2	2	2	8	8	5	5	0
3	3	3	2	2	3	3	5	0	0
5	5	3	2	2	3	0	0	0	0
5	5	3	3	2	2	0	0	0	0

Iteration 200 x batch 200

6	6	6	4	4	9	9	9	9	7	7
6	6	6	4	4	9	9	9	9	7	7
6	4	4	4	4	9	9	9	9	7	7
4	4	4	5	5	9	9	9	9	7	7
1	1	1	1	8	8	8	8	9	7	7
1	1	1	1	2	2	8	8	8	5	5
3	3	3	2	2	8	8	5	5	0	0
3	3	3	2	2	3	3	5	0	0	0
3	3	3	2	2	3	3	0	0	0	0
5	5	5	3	2	2	2	0	0	0	0

Iteration 500 x batch 200

6	6	6	4	4	9	9	9	9	7	7
6	6	6	4	4	9	9	9	9	7	7
6	6	2	2	4	4	9	9	9	7	7
6	1	2	1	4	4	9	9	9	7	7
1	1	1	1	8	8	8	8	5	7	7
1	1	1	1	8	8	8	5	5	0	0
1	3	3	3	8	8	8	8	0	0	0
3	3	3	3	2	2	8	0	0	0	0
3	3	3	3	2	2	2	0	0	0	0
3	3	3	3	2	2	2	0	0	0	0

Iteration 1000 x batch 200

6	6	6	6	9	9	9	9	7	7
6	6	6	6	9	9	9	9	7	7
6	6	2	2	4	4	9	9	9	7
6	1	1	1	8	9	9	9	9	7
1	1	1	1	8	8	8	9	9	7
1	1	1	1	8	8	8	5	5	0
1	3	3	3	8	8	8	5	0	0
3	3	3	3	2	8	8	0	0	0
3	3	3	3	2	2	2	0	0	0
3	3	3	3	2	2	2	0	0	0

Iteration 2000 x batch 200

6	6	6	9	9	9	9	9	7	7
6	6	6	9	9	9	9	9	9	7
6	6	2	2	8	9	9	9	9	7
6	1	1	1	8	9	9	9	9	7
1	1	1	1	8	8	9	9	9	7
1	1	1	2	8	8	8	5	5	9
1	3	3	8	8	8	8	5	0	0
3	3	3	8	8	8	2	0	0	0
3	3	3	3	2	2	2	0	0	0
3	3	3	3	2	2	2	0	0	0

Iteration 10000 x batch 200

6	6	6	9	9	9	9	7	7	7	7
6	6	6	9	9	9	9	7	7	7	7
6	6	6	9	9	9	9	9	9	9	7
6	2	2	2	8	8	8	9	9	9	7
1	2	2	2	8	8	8	8	5	5	5
1	2	2	2	8	8	8	9	0	0	0
1	2	2	2	3	3	3	0	0	0	0
1	2	2	3	3	3	3	0	0	0	0
1	1	1	3	3	3	5	5	0	0	0
1	1	1	3	3	3	5	5	0	0	0

Iteration 20000 x batch 200

6	6	6	9	9	9	9	9	7	7	7
6	6	6	9	9	9	9	9	7	7	7
6	6	6	9	9	9	9	9	9	9	7
6	2	2	2	8	9	9	9	9	9	7
1	2	2	2	8	8	8	9	9	9	9
1	2	2	2	8	8	8	8	5	5	0
1	2	2	2	8	8	8	8	5	5	0
1	1	2	8	8	8	8	5	5	0	0
1	1	3	8	8	5	5	0	0	0	0
1	1	3	3	3	5	5	0	0	0	0
1	1	3	3	3	5	5	0	0	0	0

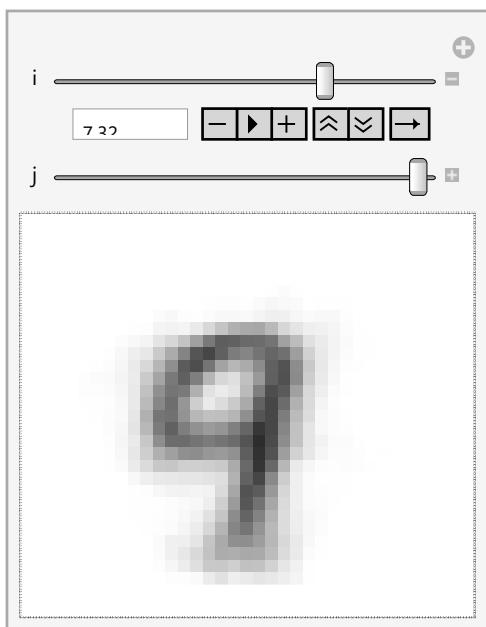
Iteration 50000 x batch 200

6	6	6	4	9	9	9	9	7	7
6	6	6	9	9	9	9	9	7	7
6	6	2	2	4	4	4	9	7	7
6	6	2	2	2	8	9	9	7	7
8	2	2	2	2	8	8	8	8	7
1	2	2	2	2	8	8	8	5	5
1	2	2	2	8	8	5	5	0	0
1	2	3	3	3	5	5	0	0	0
1	1	3	3	3	5	0	0	0	0
1	1	3	3	3	5	0	0	0	0

Iteration 100000 x batch 200

6	6	6	9	9	9	9	9	7	7
6	6	6	9	9	9	9	9	7	7
6	6	2	9	9	9	9	9	7	7
6	6	2	2	2	8	9	9	7	7
8	2	2	2	2	8	8	8	8	7
1	2	2	2	2	8	8	8	0	0
1	2	8	8	8	8	5	5	0	0
1	2	8	8	3	5	5	0	0	0
1	1	3	3	3	5	5	0	0	0
1	1	1	3	3	5	5	0	0	0

```
In[67]:= Manipulate[
  manipulate[
    (Total[bests = Table[{w[[3]], Exp[-Min[
      total          |table          |exp...|minimum
      0.5 * Power[Norm[{i, j} - {w[[1]], w[[2]]}, 2], 2], 2], 200]}], {w, weights}];
    bests * bests[[;; , 2]] / Total[bests[[;; , 2]]][[1]]) // |
      total
    ArrayReshape[#, {28, 28}] &) // Image
    |redimensionne tableau |image
  ,
  {i, 0, 10}, {j, 0, 10}]
```



Out[67]=