

Not quite what you are looking for? You may want to try:

- [A Nearly Self-organizing Tag Cloud](#)
- [Kohonen's Self Organizing Maps in C++ with Application in Computer Vision Area](#)



highlights off

13,504,202 members

1.2K jash.liao



[articles](#) [Q&A](#) [forums](#) [lounge](#)

Self-Organizing Maps



Follow



Self-Organizing Feature Maps (Kohonen maps)



Bashir Magomedov, 7 Nov 2006



4.73 (44 votes)

Rate:

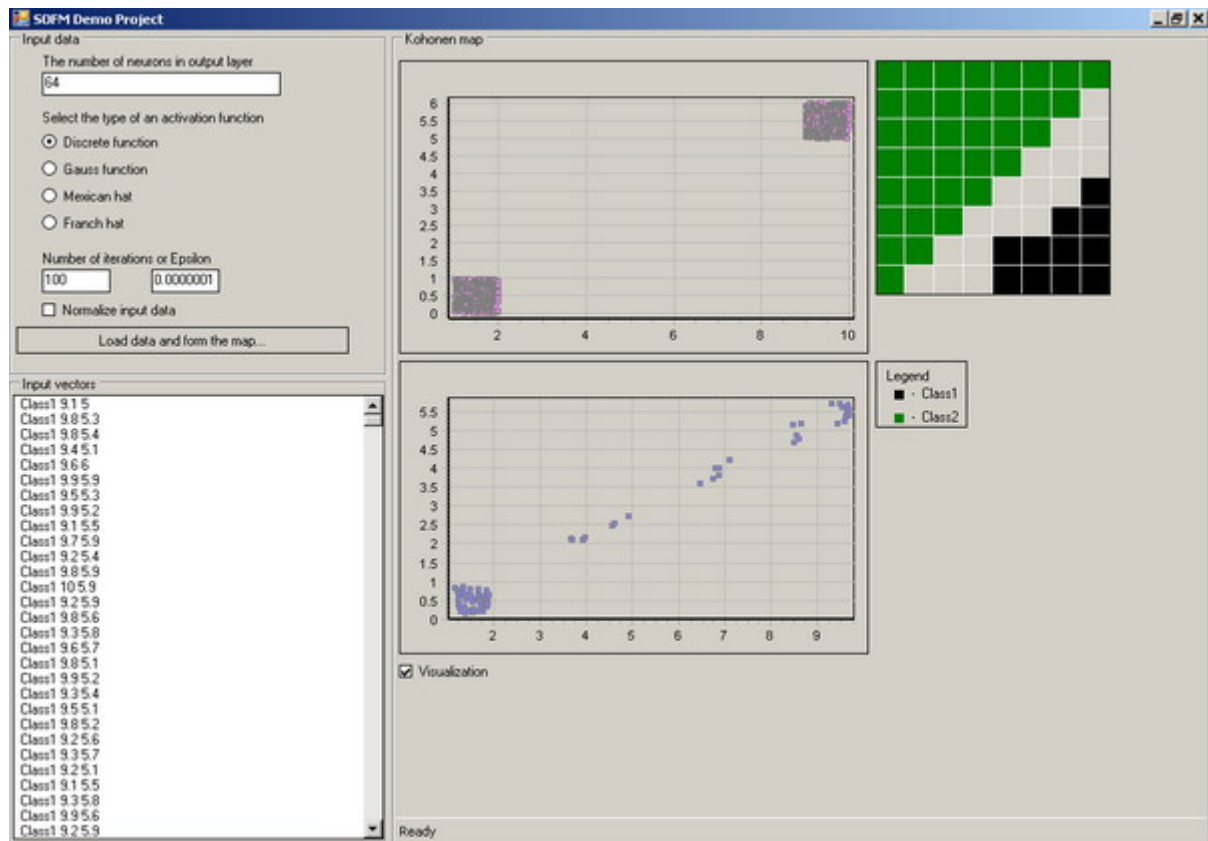
The article describes **Self-Organizing Feature Maps**. Theory and code realization are provided.



Is your email address OK? You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please [click here to have a confirmation email sent](#) so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](#).

[Download source files - 12.8 KB](#)

[Download demo project - 787 KB](#)



Glossary

- **SO(F)M - Self-Organizing (Feature) Maps**
- **(A)NN** – (Artificial) Neural Network

Introduction

Self-Organizing Feature maps are competitive neural networks in which neurons are organized in a two-dimensional grid (in the most simple case) representing the feature space. According to the learning rule, vectors that are similar to each other in the multidimensional space will be similar in the two-dimensional space. SOFMs are often used just to visualize an n -dimensional space, but its main application is data classification.

Data classification

Suppose we have a set of n -dimensional vectors describing some objects (for example, cars). Each vector element is a parameter of an object (in the case with cars, for instance – width, height, weight, the type of the engine, power, gasoline tank volume, and so on). Each of such parameters is different for different objects. If you need to determine the type of a car by looking only on such vectors, then using SOFMs, you can do it easily.

The architecture of SOFM

The SOFM NN consists of two layers of neurons (Fig. 1). The first layer is not actually a neurons layer, it only receives the input data and transfers it to the second layer. Let us consider the simplest case, when neurons of the second layer are combined into a two-dimensional grid. Other structures, like three-dimensional spheres, cylinders, etc., are out of the scope of this article. Each neuron of the third layer connects with each neuron of the second layer. The number of neurons in the second layer can be chosen arbitrarily, and differs from task to task. Each neuron of the second layer has its own *weights vector* whose dimension is equal to the dimension of the input layer. The neurons are connected to adjacent neurons by a neighborhood relation, which dictates the topology or structure of the map. Such a neighborhood relation is assigned by a special function called a *topological neighborhood* (see below).

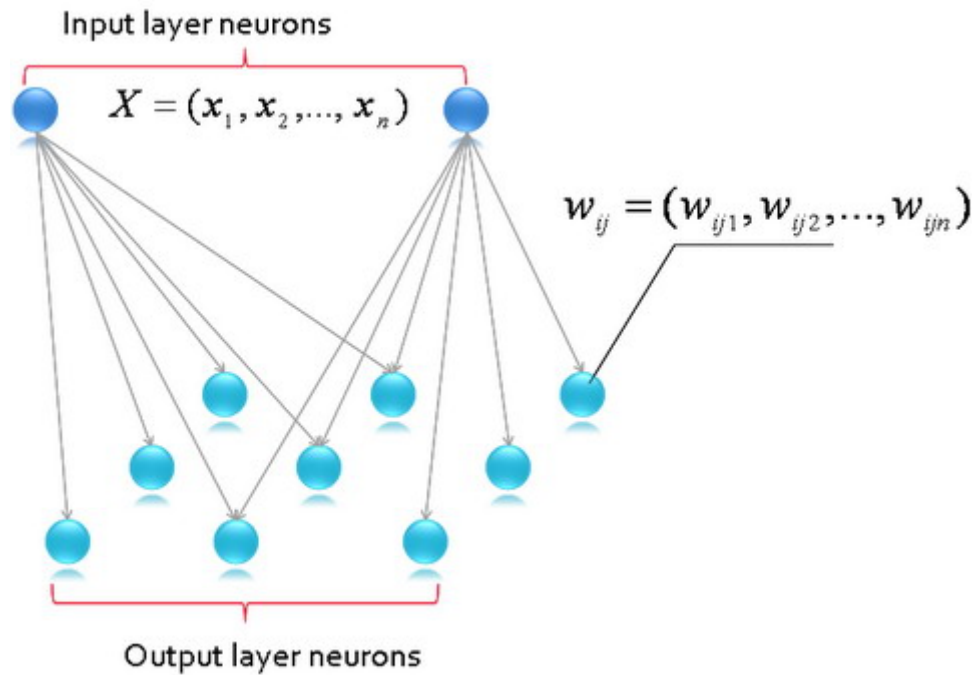


Fig. 1. The architecture of an SOFM NN

Learning rule

In the beginning of the functioning, all weights vectors of the second layer's neurons are set to random values. After that, some input-vector from the set of learning vectors is selected and set to the input of the NN. At this step, the differences between the input vector and all neurons vectors are calculated as follows:

$$D_{ij} = |X^l - W_{ij}| = \sqrt{(x_1 - w_{ij1})^2 + \dots + (x_n - w_{ijn})^2}$$

Where, i and j are the indices of neurons in the output layer. After that, the NN chooses the *winner-neuron*, i.e., the neuron whose weights vector is the most similar to the input vector:

$$D(k_1, k_2) = \min_{i,j} D_{i,j}$$

Here, k_1 and k_2 are indices of the winner-neuron. Now, we need to make a correction of the weights vectors of the winner and all the adjacent neurons. The neighborhood of a neuron is determined by a topological neighborhood function, as follows:

$$h(\rho, t) = \exp \left(-\frac{\rho^2}{2\sigma^2(t)} \right)$$

Here, ρ is the distance to the winner-neuron:

$$\rho = \sqrt{(k_1 - i)^2 + (k_2 - j)^2}$$

s is a function dictating the space of the neighborhood. In the beginning of the functioning, it involves almost the whole space of the grid, but with time, the value of s decreases. As the attraction function equal to 1, ρ equals to zero. Shown here is the simplest form of a topological neighborhood function, but in real tasks, its modifications are often used:

$$h(\rho, t) = \exp \left(-\frac{\rho^2}{\sigma^2(t)} \right) \left(1 - \frac{2}{\sigma^2(t)} \rho^2 \right)$$

- "Mexican hat";

- "French hat" - a is a priory assumed constant

$$h(\rho) = \begin{cases} 1, & |\rho| \leq a, \\ -\frac{1}{3}, & a < |\rho| \leq 3a, \\ 0, & |\rho| > 3a, \end{cases}$$

At the next step, after calculating the topological neighborhood function for each neuron, the weights of all the neurons are updated as follows:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(t)h(\rho, t)(X^l(t) - W_{ij}(t))$$

Here $\alpha(t)$ is a learning rate function that also decreases with time. If a neuron is a winner or adjacent to the winner, then its weight vector is updated, or remains unchanged otherwise. On each step, the NN determines the neuron whose weights vector is the most similar to the input vector, and corrects it and its neighbors' weights vectors to make them closer to the input vector (Fig. 2).

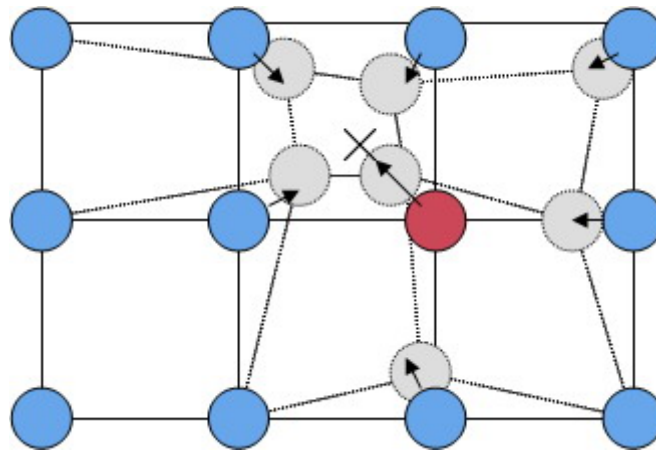


Fig. 2. Updating the winner-neuron and its neighbors towards the input vector marked with x. The solid and dashed lines correspond to the situations before and after updating, respectively

Often, each input vector from the training set is presented to the NN, and learning continues either some fixed number of cycles or while the difference between an input and the weights vectors reach some epsilon value. The difference between adjacent neurons decreases with time, and hence they organize into groups (**maps**) which correspond to one of the classes from the learning set.

Playing with the demo

Before we start, I need to say a few words about data that can be handled by the demo project. It must be tabled data. Each row of the table represents an object. The items on the row are variables of the data set. The important thing is that every data sample has the same set of variables. Thus, each column of the table holds all the values for a variable. The last variable in the row corresponds to the class name; if you do not know it, just add an additional space character. So, let us begin.

1. Run the demo project – **SOFMTest.exe**.
2. Set number of neurons in the second layer equal to **100**. Set the number of iterations to **10**. The value of the epsilon must be less than **0.01**. Select the discrete topology neighborhood function and leave the "Normalize input data" checkbox unchecked.
3. First of all, let's consider the two-classes example. Press the "Load data and form the map" button and select the **2classtest.data** file.
4. The map building will be in progress. You can see the data distribution on the top graph (Fig. 3a), and the process of neuron weights modification during the learning process (Fig. 3b).
5. When learning is finished, you will see the color map (Fig. 4) which shows how the classification process was realized.

Note: You can uncheck the visualization checkbox to accelerate the calculations.

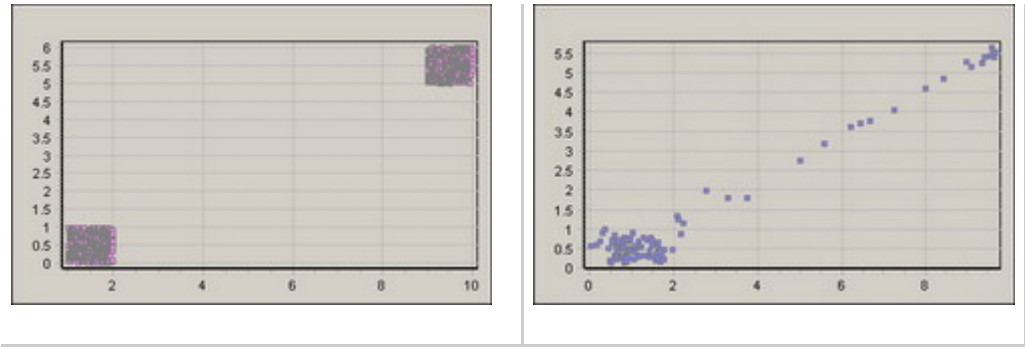


Fig. 3. The two-classes example. Each object from the data set consists of two parameters; the x-axis corresponds to the first parameter and the y-axis corresponds to the second one. a) Input data distribution. b) Weights changing process.

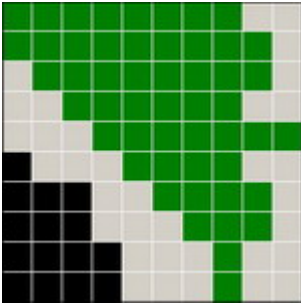


Fig. 4. Map obtained for the two-classes example. Each rectangle corresponds to a neuron from the second layer. Black neurons – first class, green neurons – second class.

Additionally, you can try how SOFM works on a standard testing data set – the well-know Fisher’s Iris data set. The data set consists of measurements from 150 Iris flowers: 50 Iris-Setosa, 50 Iris-Versicolor, and 50 Iris-Virginica (Fig. 5).

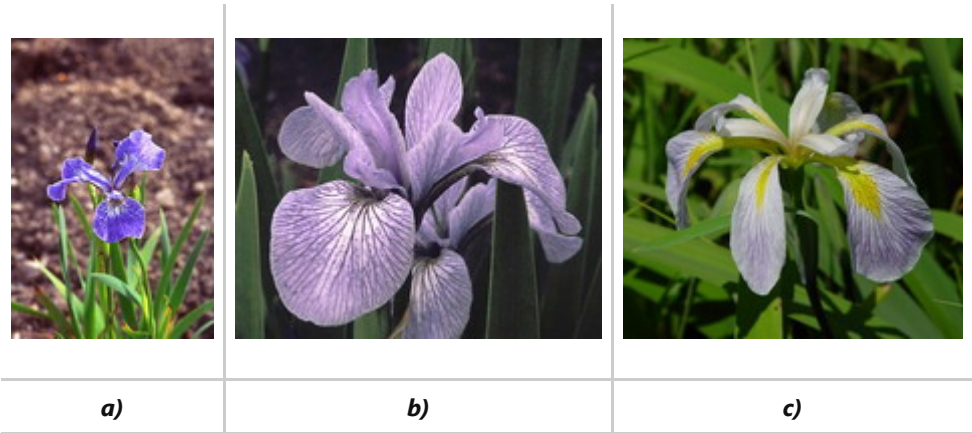


Fig. 5. The Irises. a) Setosa b) Versicolor c) Virginica.

Set the number of second layer neurons to 64, set the "Normalize input data" checkbox to checked position (all other parameters remain same), and press the "Load data and form map" button. Then, choose the *iris.data* file. After some calculations (you can uncheck the "Visualization" checkbox to perform them faster), the map will be constructed (Fig. 6). Each color on it corresponds to one of the Iris classes.

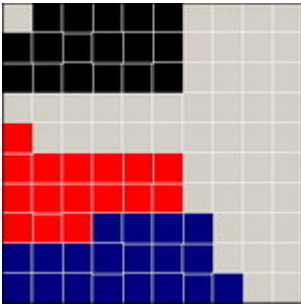


Fig. 6. Map obtained for the Iris data example. Each rectangle corresponds to a neuron from the second layer. Black neurons – first class (Setosa), red neurons – second class (Versicolor), blue neurons – third class (Virginica).

Diving into the code

Let's examine how it works. There are two main classes in the object model of SOFM (Fig. 7): **Neuron** and **NeuralNetwork**.

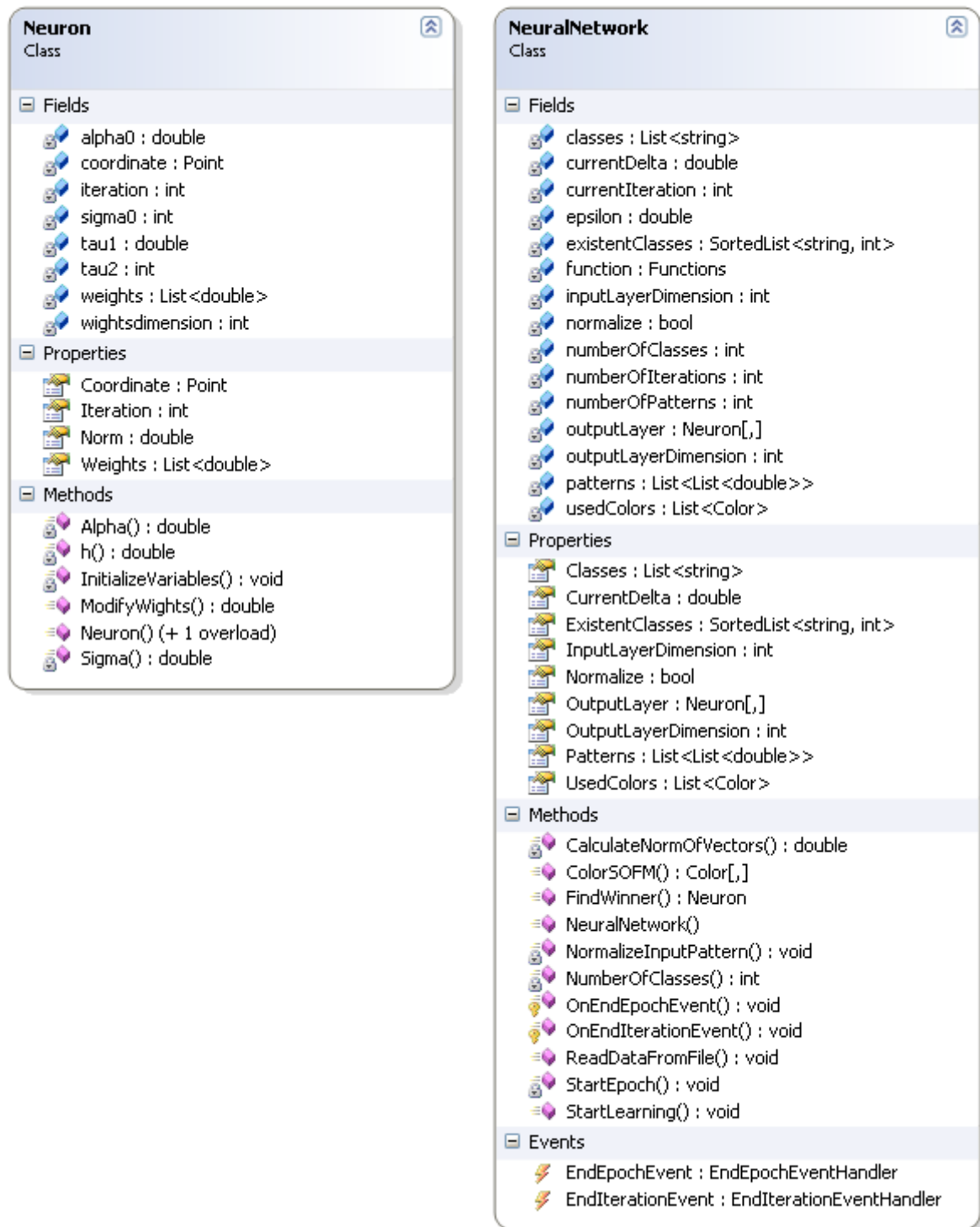


Fig. 7. The class diagram. SOFM.Neuron and SOFM.NeuralNetwork.

Each **Neuron** class has a **Coordinate** field, which shows the neuron's position in two-dimensional grid. The **Weights** field is a weights vector of generic type, and contains some **double** values. The **Neuron** class can perform the **ModifyWeights** operation, which modifies the weights vector of a neuron according to the selected topological neighborhood function (**h**) type, distance to the winner, and the number of the current iteration.

The **NeuralNetwork** class contains a two-dimensional grid of neurons, and can perform a number of operations:

Hide Copy Code

```

public NeuralNetwork(int m, int numberOfIterations, double epsilon, Functions f)
// Creates NN, with m x m neurons in second layer, and defined number of iterations,
  
```

```
// epsilon and function of topological neighborhood.
public void ReadDataFromFile(string inputDataFileName)
// Reads data from inputDataFileName
public void StartLearning()
// Starts constructing of the map.
```

All patterns from a learning set presents to NN input **numberOfIterations** times or while the weights vector modification is minor (**currentEpsilon** <= **Epsilon**).

[Hide](#) [Copy Code](#)

```
public void StartLearning()
{
    int iterations = 0;
    while (iterations <= numberOfIterations && currentEpsilon > epsilon)
    {
        List<List<double>> patternsToLearn = new List<List<double>>(numberOfPatterns);
        foreach (List<double> pArray in patterns)
            patternsToLearn.Add(pArray);
        Random randomPattern = new Random();
        List<double> pattern = new List<double>(inputLayerDimension);
        for (int i = 0; i < numberOfPatterns; i++)
        {
            pattern = patternsToLearn[randomPattern.Next(numberOfPatterns - i)];

            StartEpoch(pattern);

            patternsToLearn.Remove(pattern);
        }
        iterations++;
        OnEndIterationEvent(new EventArgs());
    }
}
```

The private **StartEpoch()** method responds for finding the winner neuron and for starting the process of weights modification through all the **outputLayer** neurons.

[Hide](#) [Copy Code](#)

```
private void StartEpoch(List<double> pattern)
{
    Neuron Winner = this.FindWinner(pattern);
    currentEpsilon = 0;
    for (int i = 0; i < outputLayerDimension; i++)
        for (int j = 0; j < outputLayerDimension; j++)
        {
            currentEpsilon +=
                outputLayer[i, j].ModifyWights(pattern, Winner.Coordinate,
                                                currentIteration, function);
        }
    currentIteration++;
    currentEpsilon = Math.Abs(currentEpsilon /
        (outputLayerDimension * outputLayerDimension));
    EndEpochEventArgs e = new EndEpochEventArgs();
    OnEndEpochEvent(e);
}
```

So, an epoch is a process of presenting one of the patterns to an NN input, and an iteration is a set of epochs when all patterns from a training set are presented to an NN input. Each time an epoch or iteration ends, the corresponding event is raised.

[Hide](#) [Copy Code](#)

```
protected virtual void OnEndEpochEvent(EndEpochEventArgs e)
{
    if (EndEpochEvent != null)
        EndEpochEvent(this, e);
}

protected virtual void OnEndIterationEvent(EventArgs e)
{
    if (EndIterationEvent != null)
        EndIterationEvent(this, e);
}
```


The **ColorSOFM()** method returns the colored matrix, where each element corresponds to a neuron from an output layer. Colors for map painting are chosen randomly.

How to use an SOFM

To use SOFM classes in your program, first of all, you have to add a reference to the *sofm.dll* assembly. Then, you need to add the following line to the **using** section:

[Hide](#) [Copy Code](#)

```
using SOFM;
```

Next, you have to create an instance of the **SOFM.NeuralNetwork** class like:

[Hide](#) [Copy Code](#)

```
SOFM.NeuralNetwork nn = new NeuralNetwork(NumberOfCards,
                                           NumberOfIterations, Epsilon, function);
```

Then, add subscribers to the **NeuralNetwork** events (if needed):

[Hide](#) [Copy Code](#)

```
nn.EndEpochEvent += new EndEpochEventHandler(nn_EndEpochEvent);
nn.EndIterationEvent += new EndIterationEventHandler(nn_EndIterationEvent);
```

And, define the **Normalize** property of instance (ether to normalize or not input data). Then, read data from the file:

[Hide](#) [Copy Code](#)

```
nn.ReadDataFromFile(ofd.FileName);
```

And start the learning process:

[Hide](#) [Copy Code](#)

```
nn.StartLearning();
```

After learning is finished, the colored map can be obtained like follows:

[Hide](#) [Copy Code](#)

```
System.Drawing.Color[,] colorMatrix = nn.ColorSOFM();
```

That's all.

Last word

An SOFM can divide the input data set to classes, only in cases when input vectors from different classes are not highly correlated (are not too similar). Only the basic concepts of SOFM are provided here. For more detailed description please refer to one of the books from references section.

References

The Steema TeeChart control was used in the demo project for graphs plotting. It is a great control migrated to .NET from Delphi, the lite version is for free. It can be downloaded from the Steema Software [site](#).

There are some links for additional study on SOFM:

- [T. Kohonen. Self organizing maps](#) - The book wrote by the creator of SOFM.
- <http://www.cs.bham.ac.uk/~jlw/sem2a2/Web/Kohonen.htm> - Nice example.
- <http://www.google.ru/search?hl=ru&q=Kohonen+maps&lr=> - Google search is also useful.

License

This article, along with any associated source code and files, is licensed under [The GNU General Public License \(GPLv3\)](#)

Share

[TWITTER](#)[FACEBOOK](#)

About the Author



Bashir Magomedov

Software Developer (Senior)

United Kingdom

[Follow this Member](#)

Work: HSBC (<http://www.hsbc.co.uk/>).

Regalia: PhD in CS, MCAD, MCPD: Web Developer, MCTS: .Net Framework 2.0., 3.5.

Interests: Programming, artificial intelligence, C#, .NET, HTML5, ASP.NET, SQL, LINQ.

Marital Status: Married, daughter

Blog: <http://www.magomedov.co.uk>

You may also be interested in...

[A Solution Blueprint for DevOps](#)

[AAMVA Barcode Driver's License Recognition and Creation with LEADTOOLS](#)

[Kohonen's Self Organizing Maps in C++ with Application in Computer Vision Area](#)

[MQTT Publication to Amazon Web Services \(AWS\)](#)

[Mapping with a GPS and C#](#)

[Using Intel® Math Kernel Library with Arduino Create](#)

Comments and Discussions

Add a Comment or Question



Search Comments



[First](#) [Prev](#) [Next](#)

SOMTEST2**KuidoKylm 10-Jul-13 3:27**

Modified program based on Bashir work
<http://mihkel.org/failid/Data%20Mining.docx>
<http://mihkel.org/failid/SOM.7z>

[Reply](#) · [Email](#) · [View Thread](#)

5.00/5 (2 votes)

**skin segmentation****lajpaal 29-Mar-12 8:18**

any body have skin segmentation idea in SOM matlab?

[Reply](#) · [Email](#) · [View Thread](#)**Nodes movement****Gobsek 20-Oct-10 6:42**

Hello!

What is the algorithm (formula) for node movements (toward to BMU) ?

Can you please refer how do you recalculate topology of neighbor nodes.

Regards.

[Reply](#) · [Email](#) · [View Thread](#)

5.00/5 (1 vote)

**My vote of 5****henrywang0423 11-Oct-10 0:35**

One of the best articles I have once met. Thank you so much for your contribution. I of course gave you 5.

[Reply](#) · [View Thread](#)**Self-organizing map****je01621 23-Nov-09 16:16**

Project title: Intelligent Humanoid Behavior Robot using self-organizing map
Our software project is to make the robot move through voice commands(SAPI voice tools) eg. walk, turn left and turn right. It uses kinematics, gait analysis and self-organizing map specifically extended Kohonen's model. Kohonen's model is used to produce set of angles for the different servomotors. Our Robot is biped each leg has 5 servomotors. There are 10 degrees of freedom all in all. My problem was how to implement the Kohonen map using the values of x, y, and z coordinates as input.. Anyone has an idea? I really need help..Thanks... 😊

[Reply](#) · [Email](#) · [View Thread](#)**college project****prashantoct02 21-Aug-09 0:37**

Hi,

Was useful reading the article. Planning to implement SOMS in Semi Digital Watermarking in our college project. Request your thoughts on that.

Prashant
(India)

[Reply](#) · [Email](#) · [View Thread](#)



SOM --> RSOM

Member 6038196 5-Apr-09 8:01



Thanks for the article. A great read. I've successfully implemented a SOM in my program. In order to make it an recurrent SOM do I simply add a leaky integrator, or are there other things I need to do?
Thanks

[Reply](#) · [Email](#) · [View Thread](#)



Image Classification work

anksw 19-Feb-09 13:54



Hello Bashir,

We are a BPO based at New Delhi, India. We are looking for some solutions which can help us in our image classification work. We have millions of electronic images which are primarily legal documents of US real estate, like Deeds, Mortgages etc. These documents have variety of formats, which is difficult to classify. We want to identify different formats available, and then create a learning sample based on it, to further segregate them from millions of images.

We have basically two requirements:

1. An unsupervised learning mechanism which can help us to identify the unique pattern images from amongst mixed images.
2. Image/Layout classification mechanism which can take input as unique formats (identified from first), and then map each available image to any of the known formats.

Please let me know if you can help us or provide guidelines in achieving this task.

PS : Please let me know your email id where I can discuss with you further.

Regards,

Ankit Jain

ankitj@competentsoftware.com

[Reply](#) · [Email](#) · [View Thread](#)



private int iteration;

Henk Meijerink 26-Jan-09 14:41



Hi Bashir,

I love your article and was pleased to rate it a 5.

In class Neuron your private int field 'iteration' seems to always have the value 1. It is never incremented or set to any other value.

Is that what you intended?

The field is only used in the private double method: `h()` and in the read-only property `Iteration`.

The only other mention of 'iteration' in class Neuron is a formal parameter in the public double method: `ModifyWeights()`, where it actually hides the field.

Thanks, Henk.

[Reply](#) · [Email](#) · [View Thread](#)

Need help for SOM using C#...

kulingo 2-Jul-08 3:37



Hi Mr. Bashir, I sent a mail for SOM using C#. Please check your "gmail" account.

[Reply](#) · [Email](#) · [View Thread](#)

Thanks for this great article! [modified]

Florian.Witteler 13-Mar-08 5:23



I have two little suggestions. It would be great, if you have time to edit your post.

1. On a Mac, the font "Symbol" is not available.

In the Chapter 'Learning rule' it would be great, if you don't use the Symbol font, but the greek letters in unicode. I posted all letters below, because Macs (and I think most of the UNIX machines) display it like this:
e.g.: "...", here r is a distance to the winner-neuron"

[here](#) is a list of all greek letters in unicode (UTF8)

This should work on all platforms.

2. In the formula-graphic for the gaussian-curve, you forgot the initial minus.

Kind regards,

Florian

modified on Wednesday, March 12, 2008 5:52 PM

[Reply](#) · [Email](#) · [View Thread](#)

formate of the input data

mralfarra 21-Jul-07 21:06



hi,

please could you give me explanation about the formate of the input data ...

I have a samples in the formate of vectors , each vector represent one sample ... how can I use them in this s.w ?

thanx

[Reply](#) · [Email](#) · [View Thread](#)

Re: formate of the input data

Bashir Magomedov 23-Jul-07 0:08



Good day. Thank you for your message. I'm actually on vacations now and can't answer to you difenetly (I just have no access to my source codes machine). But you should consider sample files from demo project, and use the same data structure.





Thank you for your attention

[Reply](#) · [Email](#) · [View Thread](#)

Re: formate of the input data



mralfarra 12-Aug-07 21:51

 thank you

[Reply](#) · [Email](#) · [View Thread](#)





**Thanks a lot,
Le Tien 22-Jun-07 4:31**

 Thank u, your project help me a lot. Bravo...

[Reply](#) · [Email](#) · [View Thread](#)



**Re: Thanks a lot,
Bashir Magomedov 22-Jun-07 14:35**


 You are welcome!


Thank you for your attention

[Reply](#) · [Email](#) · [View Thread](#)



**TeeChart.Lite.dll
stano 13-Apr-07 9:08**

 The code looks good, but I'm curious as to what TeeChart.lite.dll is?

Could you just provide the code without these third party tools?



Thanks,

Paul.

[Reply](#) · [Email](#) · [View Thread](#)



**Re: TeeChart.Lite.dll
Bashir Magomedov 13-Apr-07 12:49**

 Goof day, Paul!
 Thank you for your note. The implementation of Neural network is fully detached from any vizualization features and provided in separated assembly named SOFM.DLL. So if you want to get only NN code, just download the sources and examine SOFM project (There are no other projects in source archive file).

About Teechart. It is used only in demonstration program for some visualizations, i.e. to show Hamming distance between patterns and others.

Good luck!

Thank you for your attention

[Reply](#) · [Email](#) · [View Thread](#)

Excellent article!**Martin Welker** 26-Feb-07 20:51

You got my 5!

Martin

: Vote for me!

[Reply](#) · [Email](#) · [View Thread](#)

Missing data files**Progress** 8-Nov-06 22:31

The data files for the demo are missing.

[Reply](#) · [Email](#) · [View Thread](#)

5.00/5 (1 vote)



Re: Missing data files**Bashir Magomedov** 9-Nov-06 14:15Oops. Shame on me . Sorry.
FIXED! 

Thank you for your attention

[Reply](#) · [Email](#) · [View Thread](#)

Source code is missing**TBermudez** 7-Nov-06 22:26

Source code link is broken.

[Reply](#) · [Email](#) · [View Thread](#)

Re: Source code is missing**Bashir Magomedov** 7-Nov-06 22:28

Posting in process. Please wait for 5-10 min. Thanks



Thank you for your attention

[Reply](#) · [Email](#) · [View Thread](#)

Re: Source code is missing**Daywalker213** 12-Jan-11 13:14



thank you for such a nice article..I am trying to count blood cells in a jpg image.Can u guide me inn this direction?



Reply · Email · View Thread



Refresh

1

- General
- News
- Suggestion
- Question
- Bug
- Answer
- Joke
- Praise
- Rant
- Admin

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)
Web03 | 2.8.180417.1 | Last Updated 7 Nov 2006

請選取語言 ▼

Layout: [fixed](#) | [fluid](#)

Article Copyright 2006 by Bashir Magomedov
Everything else Copyright © [CodeProject](#), 1999-2018