Self-organizing Map – Arrows along a line

This is a partial listing of the Java code for the Self-organizing Map program that organizes data that correspond to randomly-oriented arrows. It includes all the code from the **Initialize** and **Evolution** sections, but does not include the variables' descriptions or the display section. The specification of the variables, for example the type of each variable and reasonable dimensions for the arrays, should be clear from the context.

Initialize section

```
// Set up initial parameters

cycle=0;
nhood=4;
updatemult=2.0;
scaleit=7.0;

for (int i=0; i<m; i++)
{
   arrows2d[i][0][0]=2+15*(double)(i)/m;
   arrows2d[i][0][1]=3;
   arrows2d[i][0][2]=scaleit*(0.5-Math.random());
   arrows2d[i][0][3]=scaleit*(0.5-Math.random());
   arrows2d[i][0][4]=25;
}
```

Evolution section

```
cycle++;

/* Generate random values for the elements [2] and [3] in the arrows array */

double two=scaleit*(0.5-Math.random());
double three=scaleit*(0.5-Math.random());

/* Find the winning node */

int ibest=0;
double Euclid=1.0e10;
for (int i=0; i<m; i++)
```

```
{
   double distance=(arrows2d[i][0][2]-two)*(arrows2d[i][0][2]-
two)+(arrows2d[i][0][3]-three)*(arrows2d[i][0][3]-three);
   if (distance<Euclid)
   {
      Euclid=distance;
      ibest=i;
   }
}

/* Update the weights to the winning node and to the nodes in its neighbourhood.
Assume a linear fall-off with distance from the nodes and that the updating diminishes
with cycle number. */

int left=ibest-nhood;
if (left<0) left=0;
int right=ibest+nhood;
if(right>m-1) right=m-1;

for (int i=left; i<=right; i++)
{
   double distance_from_target=1.0+(i-ibest)*(i-ibest);
   double update=updatemult/(distance_from_target+0.00001*cycle*cycle);
   arrows2d[i][0][2]=arrows2d[i][0][2]-update*(arrows2d[i][0][2]-two);
   arrows2d[i][0][3]=arrows2d[i][0][3]-update*(arrows2d[i][0][3]-three);
}
```