Research Journal of Applied Sciences 2 (10): 1057-1059, 2007
ISSN: 1815-932X

On a User Friendly Code on Visualizing Higher Dimensional Data Using Parallel Coordinates

[1]Abhishek Kumar Maurya and [2]Soubhik Chakraborty
[1]Department of Computer Science, [2]Department of Applied Mathematics,
B.I.T. Mesra, Ranchi-835215, India

Abstract: The present note is gives a user friendly code on parallel coordinates which can be used to visualize higher dimensional data.

Key words: Parallel coordinates, visualizing higher dimensional data

INTRODUCTION

There are several tools for representing and visualizing an n dimensional point in two dimensions (Wegman and Solka, 2002). The use of parallel co-ordinates is one of them. They were introduced first by Inselberg (1981, 1985) and suggested as a tool for higher dimensional data analysis by Wegman (1990). Since the original proposal, much subsequent work has been accomplished; see for example Eickemeyer *et al.* (1992). Chatterjee *et al.* (1993) explored the use of Parallel coordinates to represent the simplex algorithm in linear programming. If a pair of points in n dimensions are closer than another pair, their corresponding plots in two dimensions will also be closer. Thus we can identify clusters easily.

In traditional Cartesian coordinates we have the drawback of not being able to go beyond three dimensions due to the orthogonality constraints. In parallel coordinates this problem is settled by drawing the axes in parallel. To draw an n dimensional point say $(x_1, x_2, x_3, \ldots\ldots x_n)$ in 2 dimensions using parallel coordinates, we draw n parallel lines and locate the point x_i on the i-th coordinate axis, i = 1, 2, 3…..n and then simply join x_i with x_{i+1} for i = 1, 2, 3…..n-1. Since any number of parallel lines can be drawn in a plane, there is no upper bound on the dimension of the data that can be represented this way at least in principle but in practice there are limits to the resolution available on a computer screen and to a human eye.

An interesting link between Cartesian and parallel coordinates is the duality principle, namely, that points in one map into lines in the other and vice versa. The proof is not difficult and can be found in Wegman and Solka (2002) itself.

The present note is all about a user friendly code on parallel coordinates which can be used to visualize higher dimensional data.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
class abc{
      private:
            float value1[50][20];
            int var_len,j,value[100][100];
            char *table_name, field_name[100][100];
      public:
            void input();
            void display();
            void plot();
};

void abc::input()
{
      int i;
      double x;
      clrscr();
      cout<<"Enter the table name\n";
      cin>>table_name;
      cout<<"\nEnter the number of attributes";
      cin>>var_len;
      cout<<"\nEnter name of the attributes\n\n";
      j=0;
      for(i=0;var_len>i;i++)
       cin>>field_name[i];
      do{
          for(i=0;var_len>i;i++)
          {
```

Corresponding Author: Soubhik Chakraborty, Department of Applied Mathematics, B.I.T. Mesra, Ranchi-835215, India

```
        cout<<"\nS.          No."<<j+1<<":"<<field_
        name[i]<<"\n";
        cin>>value1[j][i];
        value[j][i]=value1[j][i]*1000;
        }
    j++;
    cout<<"\nPess 'n' to enter the next record\n\n Press
    any key to see the table....";
    }while(getch()=='n');
    display();
}

void abc::display()
{
    clrscr();
    char c;
    cout<<"******************      "<<table_name<<"
**********************\n\nS.No.";
    for(int i=0;var_len>i;i++)
     cout<<"\t"<<field_name[i];
    cout<<"\n";
    for(int k=0;k<j;k++)
    {
        cout<<"\n"<<k+1<<"\t";
        for(i=0;var_len>i;i++)
        cout<<value1[k][i]<<"\t";
    }
    cout<<"\n\n\npress....\n 'n' key to plot the table.....\n
    'u' key to update table....\n'p' key to previous....";
    do{
      c=getch();
      if(c=='n')
      {
        plot();
        break;
      }
      else if(c=='u')
      {
        cout<<"\nenter row no.\n";
        cin>>i;
        cout<<"\nenter column no.\n ";
        cin>>k;
        cout<<"\nenter new value\n";
        cin>>value1[i-1][k-1];
        value[i-1][k-1]=value1[i-1][k-1]*1000;
        display();
        break;
    }
    else if(c=='p')
    {
        input();
        break;
    }
    }while(1);
}

void abc::plot()
{

  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;
  int max,h=0,y,z;
  char temp[100],sh;
  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "");
  setcolor(2);
  line(60,30,60,450);
  line(60,450,540,450);
  line(540,30,540,450);
  line(60,30,540,30);
  outtextxy(0,0,"ATTRIBUTE");
  outtextxy(550,5,"RECORD NO.");
  outtextxy(57,22,"-3.0");
  outtextxy(97,22,"-2.5");
  outtextxy(133,22,"-2.0");
  outtextxy(173,22,"-1.5");
  outtextxy(213,22,"-1.0");
  outtextxy(253,22,"-0.5");
  outtextxy(293,22,"0.0");
  outtextxy(333,22,"0.5");
  outtextxy(373,22,"1.0");
  outtextxy(413,22,"1.5");
  outtextxy(453,22,"2.0");
  outtextxy(493,22,"2.5");
  outtextxy(533,22,"3.0");
  outtextxy(50,460,"Previous screen Press 'p'");
  outtextxy(350,460,"Press 'q' key to QUIT...");
  for(int i=0;var_len>i;i++)
  {
     outtextxy(0,h*70+58,field_name[i]);
     h++;
  }
  for(int k=0;k<j;k++)
  {
   y=300;z=30; h=0;
   for(int i=0;var_len>i;i++)
   {
     setcolor(k+1);
     max=value[k][i]*8/100;
     outtextxy(300+max,h*70+58,"*");
     line(y,z,300+max,h*70+58);
     y=300+max;
     z=h*70+58;
     h++;
   }
```

```
    outtextxy(550,30+k*10,"S. No.");
    sprintf(temp,"%d",k+1);
    outtextxy(600,30+10*k,temp);
  }
  do{
    sh=getch();
    if(sh=='p')
    {
      closegraph();
      display();
      break;
    }
  }while(sh!='q');
}

void main()
{
      abc a;
      a.input();
}
```

Remarks:

- This program demands the range of each variable x_i to be [-3, 3]. If the range is different say [a, b] then define $y_i = 3(2x_i - b - a)/(b - a)$ and we shall have the y_i's in the desired range [-3, 3]. We shall then plot for the y_i's instead of the x_i's.
- This code can be successfully executed in MS WINDOWS and LINUX operating system with minimum criteria of 64 MB RAM in Celeron, Pentium 3 or later versions of Pentium.
- The code has the charming facility of correcting any incorrect data at the time of inputting at every step. This makes it user friendly.

CONCLUSION

We have obtained a user friendly code on parallel coordinates which can be used to visualize higher dimensional data. The reader is encouraged to modify it so as to read the data from some file and also develop a JAVA version of the code so that the plot can be taken in a file.

REFERENCES

Chatterjee, A., P.P. Das and S. Bhattacharya, 1993. Visualization in linear programming using parallel coordinates. Pattern Recognition, 26: 1725-1736.

Eickemeyer, J.S., A. Inselber and B. Dimsdale, 1992. Visualizing p-flats in n-space using parallel coordinates. Technical Report G320-3581, IBM Palo Alto Scientific Center.

Inselberg, A., 1981. n-dimensional graphics, part I--lines and hyperplanes. Technical Report G320-2711, IBM Los Angeles Scientific Center, IBM Scientific Center, 9045 Lincoln Boulevard, Los Angeles (CA), 900435.

Inselberg, A., 1985. The plane with parallel coordinates. The Visual Computer, 1: 69-91.

Wegman, E.J. and J.L. Solka, 2002. On some mathematics for visualizing high dimensional data. Sankhya, 64: 429-452.

Wegman, E.J., 1990. Hyperdimensional data analysis using parallel coordinates. J. Am. Stat. Assoc., 85: 664-675.