Nuotakos uždavinys

Teorinė dalis:
Nuotakos uždavinys yra statistinio tipo uždavivys.Statistinems uždavinems spresti yra naudojamas dinaminis programavimas. Nuotakos uždavinio tiklas yra maksimizuoti pelną, išrenkant optimalų jaunikį.

Užduotis:
Egzistuoja n jaunikių. Apibrėšime tikrajį jaunikio gerumą kaip w_i , i = 1..n. Jaunosios įspudį apibrėšime kaip s_i. Toliau apibrėšime p(w) kaip išanktinio gerumo w_i tikimybės tankį. s_i tikimybės tankį apibrėšime kaip p(s|w). Priimame, kad skirtingu jaunikių gerumai yra nepriklausomi ir pasiskirstę pagal Gauso dėsnį. Parašyti programą kuri nubrėžtų jaunosios kritinių pasirinkimų kreivę.

Programos komentavimas:
(Programos tekstas pateiktas 1 priede)

Skaičiavimuose naudojami masyvai:

double p_w[] - saugomos suskaičiuotos p(w) reikšmės;
double ps_s[] - saugomos apskaičiuotos $p_s(s)$ reikšmės;
double p_sw[][] - saugomos suskaičiuotos p(s) * p(w);
double p_w_s[][] - saugomos suskaičiuotos p(s|w) reikšmės;
double rel[] - nuotakos pasirinkimų matrica;
double w[] - suskaičiuoti nuotakos kritiniai pasirinkimų taškai.

1) formuojamas p(w) masyvas:

```
for (I=0;I<100;I++) {
        p_w[I] = tikimybe(Sigma_w,nuoseklus(I), -0.1);
}
čia :
        Sigma_w - pasiskirstymo išsibarstymas;
        nuoseklus(I) - funkcija grazinanti reiksme is intervalo -1..1;
```

2) analogiškai skaičiuojami ir p(s) * p(w):

```
for(i=0;i<100;i++) {
        s = nuoseklus(i);
        for(j=0;j<100;j++) {
           w = nuoseklus(j);
           p_sw[i][j] = p_w[j]*tikimybe(Sigma_w,s,w);
        }
    }
```

3) susormuojamas $p_s(s)$:

```
for(i=0;i<100;i++) {
   ps_s[i] = 0.0;
   for(j=0;j<100;j++) {
         ps_s[i] += p_sw[i][j];
   }
   ps_s[i]/=100;
}
```

4) suformuojame p(s|w) :

```
for(i=0;i<100;i++) {
                  for(j=0;j<100;j++) {
                  p_w_s[i][j] = p_sw[j][i]/ps_s[j];
                  }
}
```

5) apskaičiuojama $U_n(s)$:

```
for(i=0;i<100;i++) {
                  rel[jaun_sk-1][i] = EUn(i);
}
```

6) artėjant prie pirmo jaunikio skaičiuojame U_{n-k}, o taipogi suskaičiuojame nuotakos kritinius pasirinkimo taškų masyvą w[], įvertinant laukimo nuostolius:

```
w[jaun_sk-1] = -1;

for(j=jaun_sk-2;j>=0;j--) {
         temp = EUn_1(j+1);
         for(i=0;i<100;i++) {
            rel[j][i] = Math.max(temp,rel[j+1][i]);
         }
         w[j] = Krit_task(j)-j/100;
      }
  }
```

Turint kritinius pasirinkimo taškus kiekvienam jaunikiui, belieka tik sulyginti juos su įspudžiu ir galima nuspresti ar tekėti už jo ar ne. Jei jaunikio įspudis didesnis nuotaka toliau nelaukia, o teka už jo.

1. Priedas. Programos tekstas

```
import java.awt.*;
import java.util.*;
import java.applet.*;

public class nuotakos_uzd extends Applet {
   Button button1;

   public void init() {

      super.init();

      setLayout(null);
      resize(100,100);
      button1 = new Button("Pradeti");
      add(button1);
      button1.reshape(20,10,150,30);

   }

   public boolean handleEvent(Event event) {
      if (event.id == Event.ACTION_EVENT && event.target == button1) {
           new langas();
            return true;
      }
      return super.handleEvent(event);
   }

}

//---------------------------------------------------------------------------
//Si klase atsakinga uz pagrindini programos langa
//---------------------------------------------------------------------------
final class langas extends Frame {

   //patikrinimas ar visos ivestos reiksmes tenkina intervalus
   boolean check_par()
   {
      jaun_pas_s = toDouble(jaun_pas.getText());
      if(jaun_pas_s<0.1) {
         Msg = new MessageBox(this, "Jaunikiu pasiskyrstymas turi buti didesnis nei 0.1");
         Msg.show();
              return false;
      }
      else if(jaun_pas_s>1.0) {
         Msg = new MessageBox(this, "Jaunikiu pasiskyrstymas turi buti mazesnis nei 1.0");
         Msg.show();
         return false;
      }
      return true;
   }
```

```
//---------------------------------------------------------------------------
// konvertuoja eilute i realu skaiciu
//---------------------------------------------------------------------------
   double toDouble(String s) {
      int i, j=0, j1 = 0, i1 = 1;
      boolean point = false;
      for(i=0;i<s.length();i++) {
         if(s.charAt(i) == '.') point = true;
         else {
            if(point) {
               i1*=10;
               j1 = j1*10+(int)(s.charAt(i))-48;
            }
            else j=j*10+(int)(s.charAt(i))-48;
         }
      }
      return (double)j+(double)j1/(double)i1;
   }
//---------------------------------------------------------------------------
// konstruktorius
//---------------------------------------------------------------------------
   public langas() {

            super("Nuotaku sprendimai (Rolandas Jaskovikas)");

            setLayout(null);
            addNotify();
            resize(insets().left + insets().right + 640, insets().top + insets().bottom + 500);
            label1=new Label("Jaunikiu pasiskyrstymas", Label.RIGHT);
            add(label1);
            label1.reshape(insets().left + 55 ,insets().top + 440,145,22);
            jaun_pas=new TextField("0.24",13);
            add(jaun_pas);
            jaun_pas.reshape(insets().left + 220,insets().top + 440,40,22);
            rec_button=new Button("Perskaiciuoti");
            add(rec_button);
            rec_button.reshape(insets().left + 450,insets().top + 440,161,26);
            quit_button=new Button("Pabaiga");
            add(quit_button);
            quit_button.reshape(insets().left + 450,insets().top + 470,161,26);

            show();
   }

   public void paint(Graphics g) {

         g.clearRect(0,0,640,500);
         g.setFont(new Font("TimesRoman", Font.BOLD,10));
         g.drawString("1",25,56);
         g.drawString("0.0",25,266);
         g.drawString("1",445,266);
         g.drawString("-1",25,451);
         g.setColor(Color.red);
         g.drawLine(48,50,52,50);
```

```
        g.drawLine(50,0,50,500);
        g.drawLine(50,250,500,250);
        g.drawLine(450,252,450,248);
        g.drawLine(48,450,52,450);
            data.Draw(g);
}
   //ivykiu valdymas
   public boolean handleEvent(Event event) {
          if (event.id == Event.ACTION_EVENT && event.target == rec_button) {
                  clickedPerskaicButton();
                  return true;
          }
          else
          if (event.id == Event.ACTION_EVENT && event.target == quit_button){
          handleEvent(new Event(this, Event.WINDOW_DESTROY, null));
          }
          else
          if (event.id == Event.WINDOW_DESTROY) {
             hide();        // hide the Frame
             dispose();     // tell windowing system to free resources
             System.exit(0); // exit
             return true;
          }
          return super.handleEvent(event);
   }

   public static void main(String args[]) {
          new langas();
   }

   // kintamieji
   Label label1;
   TextField jaun_pas;
   Button rec_button;
   Button quit_button;
   MessageBox Msg;
   double jaun_pas_s = 0.0;
   skaic data = new skaic();

   public void clickedPerskaicButton() {
      if(check_par()) {
         data.init(jaun_pas_s);
            data.recalc_bride();
            repaint();
      }
   }
}

//---------------------------------------------------------------------------

class MessageBox extends Dialog {

   public MessageBox(Frame parent, String msg) {
```

```
super(parent, "Alert...", true);
        setResizable(false);

        //{{INIT_CONTROLS
        setLayout(null);
        addNotify();
        resize(insets().left + insets().right + 421, insets(). top + insets().bottom + 129);
        label1=new Label(msg, Label.CENTER);
        label1.setFont(new Font("TimesRoman",Font.BOLD,14));
        add(label1);
        label1.reshape(insets().left + 24,insets().top + 22,358,27);
        OKButton=new Button("OK");
        add(OKButton);
        OKButton.reshape(insets().left + 172,insets().top + 65,72,26);
        //}}
}

public synchronized void show() {
        Rectangle bounds = getParent().bounds();
        Rectangle abounds = bounds();

        move(bounds.x + (bounds.width - abounds.width)/ 2,
            bounds.y + (bounds.height - abounds.height)/2);

        super.show();
}

public synchronized void wakeUp() {
        notify();
}

public boolean handleEvent(Event event) {
        if (event.id == Event.ACTION_EVENT && event.target == OKButton) {
                clickedOKButton();
                return true;
        }
        else
        if (event.id == Event.WINDOW_DESTROY) {
           hide();
           return true;
        }
        return super.handleEvent(event);
}

//{{DECLARE_CONTROLS
Label label1;
Button OKButton;
Label label2;
Label label3;
//}}

public void clickedOKButton() {
        handleEvent(new Event(this, Event.WINDOW_DESTROY, null));
}
```

```
}

//---------------------------------------------------------------------------
//Si klase atsakinga uz visus atliekamus skaiciavimus
//---------------------------------------------------------------------------
class skaic {
   double Sigma_w ;
   double ps_s[];            //apskaiciuotas ps(s)
   double p_sw[][];          //apskaiciuotas p(s)*p(w)
   double p_w_s[][];         //apskaiciuotas p(w|s)
   double rel[][];           //nuotakos pasirinkimu matrica
   double w[];               //kritiniai nuotakos pasirinkimu taskai
   int jaun_sk=100;

   public void Draw(Graphics g) {
           int i;
           g.setColor(Color.white);
           for(i=0;i<100-1;i++) g.drawLine(i*4+50,250-(int)(w[i]*200),(i+1)*4+50,250-(int)(w[i+1]*200));
   }
//---------------------------------------------------------------------------
// nuoseklus pasiskyrstymas intervale -1..1
//---------------------------------------------------------------------------
  double nuoseklus(int i)
  {
           return 2.0/100*i-1;
  }
//---------------------------------------------------------------------------
//uzpildo reiksmes i masyvus:
//  p_sw    - p(s)*p(w)
//  ps_s    - ps(s)
//  p_w_s   - p(w|s) = p(s)*p(w)/ps(s)
//---------------------------------------------------------------------------
   void formuoti_duomenis(double []p_w) {
      int i, j;
      double sum, w, s;

      for(i=0;i<100;i++) {
         s = nuoseklus(i);
         for(j=0;j<100;j++) {
            w = nuoseklus(j);
            p_sw[i][j] = p_w[j]*tikimybe(Sigma_w,s,w);
         }
      }

      for(i=0;i<100;i++) {
              ps_s[i] = 0.0;
         for(j=0;j<100;j++) {
            ps_s[i] += p_sw[i][j];
         }
              ps_s[i]/=100;
      }

      for(i=0;i<100;i++) {
         for(j=0;j<100;j++) {
```

```
                p_w_s[i][j] = p_sw[j][i]/ps_s[j];
            }
        }

    }
//---------------------------------------------------------------------------
// suskaiciuoja Un reiksmes
//---------------------------------------------------------------------------
    double EUn(int i) {
        int j;
        double sum = 0.0;

        for(j=0;j<100;j++) {
            sum += nuoseklus(j)*p_w_s[j][i];
        }
        return sum/100;
    }

//---------------------------------------------------------------------------
// suskaiciuoja Un-1 reiksmes
//---------------------------------------------------------------------------
    double EUn_1(int j) {
        double sum = 0.0;
        int i;

        for(i=0;i<100;i++) {
            sum += rel[j][i]*ps_s[i];
        }
        return sum/100;
    }
//---------------------------------------------------------------------------
//  perskaiciuojami duomenys, susije su nuotakos pasirinkimais
//---------------------------------------------------------------------------
    void recalc_bride() {
        double []p_w = new double[100];
        int i, j;
        double temp;
        double sum;

        //suformuojamas p(w)
        for(i=0;i<100;i++) {
            p_w[i] = tikimybe(Sigma_w,nuoseklus(i),-0.1);
        }
        formuoti_duomenis(p_w);

        for(i=0;i<100;i++) {
            rel[jaun_sk-1][i] = EUn(i);
        }
        w[jaun_sk-1] = -1;

        //pasirinkimu formavimas
        for(j=jaun_sk-2;j>=0;j--) {
            temp = EUn_1(j+1);
            for(i=0;i<100;i++) {
```

```
                rel[j][i] = Math.max(temp,rel[j+1][i]);
            }
            w[j] = Krit_task(j)-j/100;
        }
    }
//---------------------------------------------------------------------------
// surandamas kritinis taskas
//---------------------------------------------------------------------------
    double Krit_task(int j) {
        int i, min;

        min = jaun_sk-1;
        for(i=0;i<100;i++) {
            if(rel[j][i] < rel[j][min]) min = i;
        }
        return rel[j][min];
    }
//---------------------------------------------------------------------------
//inicializavimas
//---------------------------------------------------------------------------
    void init(double d_gr) {
        int i;

        rel = new double[100][100];
        w = new double[100];
        p_w_s = new double[100][100];
        p_sw = new double[100][100];
        ps_s = new double[100];

        Sigma_w = d_gr;
    }

//---------------------------------------------------------------------------
// tikimybe, pagal Gauso pasiskirstyma
//---------------------------------------------------------------------------
    double tikimybe(double sigma, double omega, double miu) {
        double x = (omega-miu)/sigma;
        return 2.0/(Math.sqrt(2.0*Math.PI)*sigma)*Math.exp(-x*x*0.5);
    }
}
```

2. Priedas HTML tekstas.

```
<HTML>
<HEAD>
<TITLE> Nuotakos uzdavinys (Rolandas Jaskovikas) </TITLE>
</HEAD>
<BODY>

<P>
<CENTER></B><FONT SIZE = 14> Nuotakos uzdavinys </FONT></B></CENTER>
</P>
<p>
Norint pradeti darba reikia nuspausti mygtuka "Pradeti". Atsiradus papildomui
langui reikia nurodyti jaunikiu gerumu pasiskirstyma(nuo 0.1 iki 1).
Kreives perskaiciavimiui reikia nuspausti myktuka "Perskaiciuoti". Programos
darbas baigiamas nuspaudus muktuka "Pabaiga".
</p>
<p>
        Rolandas Jaskovikas (jaskov@soften.ktu.lt)
</P>

<APPLET CODE="nuotakos_uzd.class" WIDTH=283 HEIGHT=120></APPLET>
</BODY>
</HTML>
```