ДОДАТОК А

Графічний матеріал атестаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Нейромережеві методи побудови математичних моделей нелінійних об'єктів

Атестаційна робота

Другий (магістерський) рівень

1

Автор:
Тищенко О.І.,
студ. гр. КСМм-19-1

Керівник:
Руденко О.Г.
проф. каф. ЕОМ

Мета і задачі роботи

Мета: дослідити та визначити існуючі методи та підходи до побудови поведінкових моделей користувачів комп'ютерних систем.

Задача:

- побудова комплексної моделі аналізу;
- розробка генеративної моделі поведінки;
- реалізація системи моніторингу діяльності користувачів КС у корпоративних мережах.

2

Проблематика

- На сьогодні масштабне використання комп'ютерних технологій у всіх сферах привернуло більшу увагу до самого користувача. Наприклад, в системах спостереження за персоналом, системах безпеки та при створенні екосистеми для користувача, що персоналізуються та у Web-додатках.

3

Проблематика

- В зв'язку з цим виникає питання щодо створення нових ефективних систем аналізу та моніторингу поведінки користувачів комп'ютерних систем

4

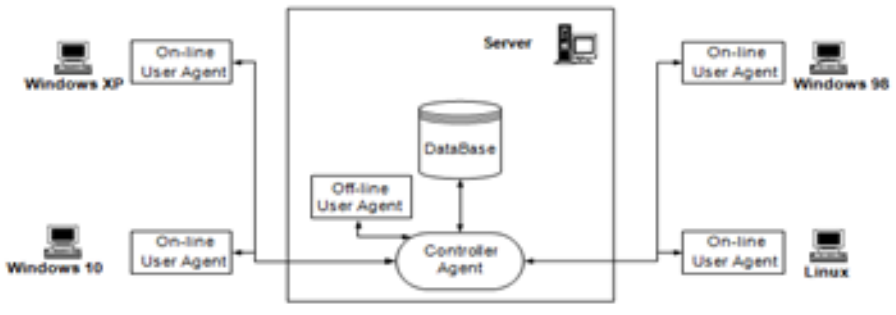
Аналіз та дослідження джерел

- В процесі підготовки до виконання даного проекту було проведено пошук інформації щодо існуючих методів та підходів до побудови поведінкової моделі та комплексного аналізу.
- Аналіз існуючих систем моніторингу показав, що ніяк не аналізується статистична інформація про діяльність користувачів.

5

Рішення проблеми

- Розроблено спеціальне застосування для аналізу поведінки користувачів КС з урахуванням статистичних та динамічних властивостей.
- Система надає можливість автоматизації трудомісткого процесу аналізу запитів у режимі реального часу.



6

Структура програми

- Система реалізована на основі клієнт - серверної архітектури.
- Серверна частина є додатком (створення та розміщення агентів на стороні користувача для аудиту).
- Клієнтська частина (графічний інтерфейс з параметрами клієнтської платформи, стан агента, дані про роботу).

7

Структура програми

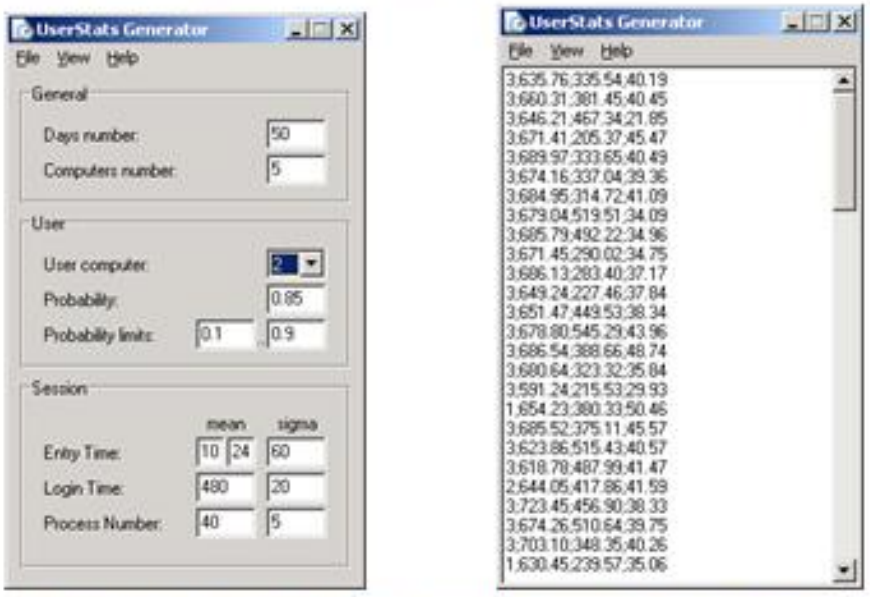
8

Алгоритм роботи застосування

- 1) Відправка запиту в клієнтській частині до сервера.
- 2) Дані надійшли до серверу і обробляються згідно заданих параметрів.
- 3) Перетворення запиту та даних з сервера у відповідь (залежить від параметрів, переданих із запитом, і даних, отриманих зі сховища даних).

9

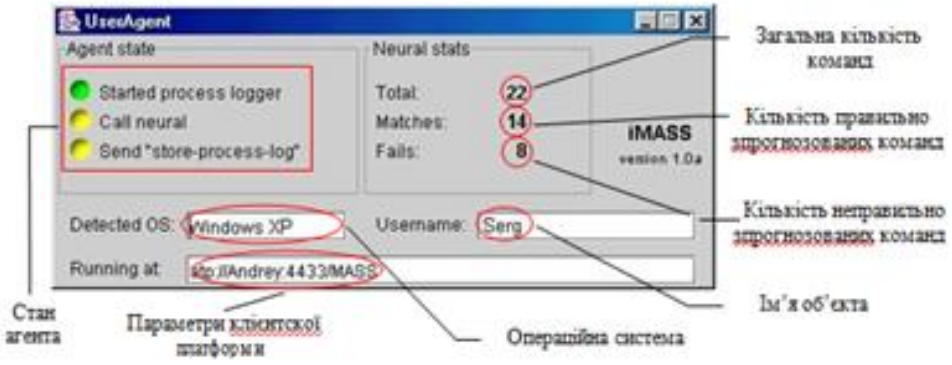
Результати роботи



Інтерфейс системи генерування даних за сеанс (а) і результати її роботи (б)

10

Результати роботи



11

Висновок

- В процесі розробки системи аналізу та моніторингу дій користувачів у комп'ютерних системах, було проведено багато роботи з пошуку матеріалів, по цьому питанню, проведено аналіз існуючих робочих аналогів системи.
- Система пройшла тестові опробування, та дала гарні показники своєї працездатності. Програма повністю готова до роботи та може бути використана звичайними користувачами для вирішення проблем.

12

ДОДАТОК Б

Програмний код реалізації нейромережевих моделей

Б.1 Клас Layer для реалізації оболонки нейронної мережі

```
package neural;
import java.io.Serializable;
public class Layer implements Serializable
{
Math m;
boolean bLast = false; double dMaxNum;
int LSize;
int prevLSize; int MtxSize; double[] Bias; double[] Weights; double[]
Outputs;

public Layer (int CurSize, int PrevSize)
{
LSize = CurSize; prevLSize = PrevSize; MtxSize  = LSize*prevLSize;
Bias  = new double[LSize]; Weights  =  new  double[MtxSize];  Outputs
= new double[LSize];
}

void Run (double[] prevOut)
{
int Index = 0; double dTmp;
for (int i=0; i<LSize; i++)
{
dTmp = Bias[i];
for (int j=0; j<prevLSize; j++)
{
dTmp += prevOut[j]*Weights[Index];
++Index;
}


dLastFunction(dTmp);

}
}

if (bLast) Outputs[i] =

else Outputs[i] = dTrFuntion(dTmp);


double dTrFuntion (double X)
{
if (40.0<X) return 1.0;
if (X<-40.0) return 0.0; return 1.0/(1.0 + m.exp(-X));
}
double dLastFunction (double X)
{
if (dMaxNum/2<X)  return dMaxNum; if (X<-dMaxNum/2) return 0.0; return
X+dMaxNum/2;
}
}
```

Б.2 Клас TestNeural для реалізації тестування мережі

```
package neural;

import java.io.*;
import java.util.Properties; import java.io.Serializable;

public class TestNeural implements Serializable
{
Layer[] Layers; int LNum;
double pdWeights[];
public boolean CanRun = false; String NetFileName;
String ParFileName;

public TestNeural (String Path) throws IOException
{
ParFileName = Path + ".par"; NetFileName = Path + ".net";
System.out.println("PATH1: " + ParFileName); System.out.println("PATH2:
" + NetFileName);
ReadArch();
CanRun = ReadNet(); SetWeights();
}

void ReadArch () throws IOException
{
String myString;
Properties props = System.getProperties(); InputStream is;
int Arch[] = new int[4];
is = new FileInputStream(ParFileName); props.load(is);
LNum  =
(Integer.valueOf((String)props.get("train.parameters.0"))).intValu e();
Layers= new Layer[LNum]; Arch[0] =
(Integer.valueOf((String)props.get("train.parameters.1"))).intValu e();
Arch[1]=
(Integer.valueOf((String)props.get("train.parameters.2"))).intValue();
Arch[2] =
(Integer.valueOf((String)props.get("train.parameters.3"))).intValu e();
Arch[3]                                                                    =
(Integer.valueOf((String)props.get("train.parameters.4"))).intValu e();
int iNNum = 0; switch (LNum)
{


(Arch[iNNum+1],Arch[iNNum]);

++iNNum; (Arch[iNNum+1],Arch[iNNum]);
++iNNum; (Arch[iNNum+1],Arch[iNNum]); Layers[0].bLast = true;

case 3: Layers[2] = new Layer


case 2: Layers[1] = new Layer


case 1: Layers[0] = new Layer

Layers[0].dMaxNum =
(Double.valueOf((String)props.get("train.parameters.12"))).doubleV alue();

break;
}
iNNum = 0;
```

```
for (int i=0; i<LNum; i++) iNNum += Layers[i].MtxSize + Layers[i].LSize;
pdWeights = new double[iNNum];
}

boolean ReadNet () throws IOException
{
double[] WTmp;
File FNet = new File (NetFileName); if (FNet.exists())
{
try
{
ObjectInputStream OS = new
ObjectInputStream(new FileInputStream(FNet));
WTmp = (double[])OS.readObject(); if (WTmp.length !=

pdWeights.length) return false; pdWeights[i] = WTmp[i];
}


for (int i=0; i<WTmp.length; i++) OS.close();


false; }

catch (ClassNotFoundException e) { return

catch (EOFException e) { return false; }
return true;
}
else return false;
}

void SetWeights ()
{
int Seek = 0;
for (int j=LNum-1; 0<=j; j--)
{
for (int i=0; i<Layers[j].LSize; i++) Layers[j].Bias[i] =
pdWeights[Seek+i];
Seek += Layers[j].LSize;
for (int i=0; i<Layers[j].MtxSize; i++) Layers[j].Weights[i] =
pdWeights[Seek+i];
Seek += Layers[j].MtxSize;
}
}

public int Test (double[] dpInp)
{
Layers[LNum-1].Run (dpInp);
for (int i=LNum-1; 0<i; i--) Layers[i- 1].Run(Layers[i].Outputs);
return (int)(Layers[0].Outputs[0] + 0.5);
}
}
```

Б.3 Клас TrainNeural для реалізації навчання нейронної мережі

```
package neural;

import java.io.*;
import java.util.Properties;
```

```
      public class TrainNeural extends TestNeural
      {
      Math m;
      int iRecordLength = 0; int iRecordsNumber = 0; double pdInputBuffer[];
double pdInput[];
      int piClass[];
      int StopFlag = 5500000;
      long lCheckPoint
      int   lNumOfPoints; double dTrFactor; double dOffset;

      double LC[] = new double[3]; double MC[] = new double[3];
      double  MBias[][]  =  new  double[3][];  double  Moments[][]  =  new
double[3][]; double Errors[][] = new double[3][];


      public TrainNeural (String Path) throws IOException
      {
      super(Path); ReadParameters(); ReadData (Path + ".trn");
      }

      void ReadParameters ()
      {
      String myString;
      Properties props = System.getProperties(); InputStream is;
      try
      {
      is = new FileInputStream(ParFileName); props.load(is);
      switch (LNum)
      {
      case                                3:                                LC[1]=
(Double.valueOf((String)props.get("train.parameters.6"))).doubleVa lue();
      MC[1]=
      (Double.valueOf((String)props.get("train.parameters.9"))).doubleVa
lue();

      LC[2]=
      (Double.valueOf((String)props.get("train.parameters.5"))).doubleVa
lue();
      MC[2]=
      (Double.valueOf((String)props.get("train.parameters.8"))).doubleVa
lue();

      break;
      case                                2:                                LC[1]=
(Double.valueOf((String)props.get("train.parameters.5"))).doubleVa lue();
      MC[1]=
      (Double.valueOf((String)props.get("train.parameters.8"))).doubleVa
lue();
      } LC[0]=
      (Double.valueOf((String)props.get("train.parameters.7"))).doubleVa
lue();
      MC[0]=
      (Double.valueOf((String)props.get("train.parameters.10"))).doubleV
alue();
      lCheckPoint                                                          =
(Long.valueOf((String)props.get("train.parameters.14"))).longValue ();
      dTrFactor
      =(Double.valueOf((String)props.get("train.parameters.13"))).doubleV
alue();
      dOffset     =
      (Double.valueOf((String)props.get("train.parameters.16"))).doubleV
alue();
      lNumOfPoints=
```

```
(Integer.valueOf((String)props.get("train.parameters.15"))).intVal ue();
      StopFlag                                                    =
(Integer.valueOf((String)props.get("train.parameters.17"))).intVal ue();

      Rand((Double.valueOf((String)props.get("train.parameters.11"))).do
ubleValue());
      SetWeights();
      for (int i=0; i<LNum; i++)
      {
      MBias[i]    = new
      double[Layers[i].LSize];
      Moments[i] = new double[Layers[i].MtxSize]; Errors[i] =              new
double[Layers[i].LSize];
      }
      }
      catch(Exception ex)
      {
      ex.printStackTrace();
      }
      }

      void Rand (double Range)
      {
      for    (int    i=0;    i<pdWeights.length;    i++)    pdWeights[i]    =
Range*(2.0*m.random() - 1.0);
      }

      void GetWeights ()
      {
      int Seek = 0;
      for (int j=LNum-1; 0<=j; j--)
      {
      for    (int    i=0;    i<Layers[j].LSize;    i++)    pdWeights[Seek+i]    =
Layers[j].Bias[i];
      Seek += Layers[j].LSize;
      for   (int   i=0;   i<Layers[j].MtxSize;   i++)   pdWeights[Seek+i]   =
Layers[j].Weights[i];
      Seek += Layers[j].MtxSize;
      }
      }

      boolean ReadData (String datFile) throws IOException
      {
      String myString; int i, iIndex = 0;
      File FDat = new File (datFile); double[] buf;
      iRecordsNumber = 0;
      if (!FDat.exists()) return false;

      BufferedReader in = new BufferedReader(new InputStreamReader

      (new DataInputStream(new FileInputStream(FDat))));
      try
      {
      myString = in.readLine(); iRecordLength = CheckString(myString);
      --iRecordLength; iRecordsNumber = 1; while (true)
      {
      myString = in.readLine();
      if (myString == null)   break;
      ++iRecordsNumber;
      }
      }
      catch (EOFException e){ }
      catch (IOException e) { iRecordsNumber = 0; } in.close();
      buf = new double[iRecordLength+1]; pdInput = new double[iRecordLength];
```

```
if ((0<iRecordLength)&&(0<iRecordsNumber))
{
piClass = new int[iRecordsNumber]; pdInputBuffer = new
double[iRecordLength*iRecordsNumber];
BufferedReader in1 = new BufferedReader(new InputStreamReader

(new DataInputStream(new FileInputStream(FDat)))); try
{


j++)


break;

for (int j=0; j<iRecordsNumber;

{
myString = in1.readLine(); if (myString == null)

if ((iRecordLength+1) !=

ReadString (myString,buf)) break;
for (i=0; i<iRecordLength;
i++) pdInputBuffer[iRecordLength*j+i] = buf[i];
piClass[j] =
(int)buf[iRecordLength];
}
}



0; }

catch (EOFException e){ }
catch (IOException e) { iRecordsNumber =

in1.close();
}
return 0<iRecordsNumber;
}

int CheckString (String s)
{
int spaceAt, startingFrom = 0, count = 0; while (true)
{
spaceAt = s.indexOf(" ",0); if (spaceAt == -1) break; if (spaceAt == 0)
s =
s.substring(1,s.length());
else
{
++count; s =
s.substring(spaceAt+1,s.length());
}
}
if (0<s.length()) ++count; return count;
}

int ReadString (String s, double[] buf)
{
int spaceAt, startingFrom, count = 0; Double dA;
startingFrom = 0; while (true)
{
spaceAt = s.indexOf(" ",0); if (spaceAt == -1) break; if (spaceAt == 0)
```

```
s =
      s.substring(1,s.length());
      else
      {
      dA = Double.valueOf(s.substring(0,spaceAt));
      buf[count] = dA.doubleValue();
      ++count; s =
      s.substring(spaceAt+1,s.length());
      }
      }
      if (0<s.length())
      {
      dA = Double.valueOf(s); buf[count] = dA.doubleValue();
      ++count;
      }
      return count;
      }

      void ErrorCalc ()
      {
      int Index;
      for (int k=0; k<LNum-1; k++)
      {
      for (int j=0; j<Layers[k+1].LSize; j++)
      {



      i++)

      Index = j; Errors[k+1][j] = 0;
      for (int i=0; i<Layers[k].LSize;

      {
      Errors[k+1][j] += Errors[k][i]*Layers[k+1].Weights[Index];

      Layers[k+1].LSize;
      }


      Index  +=  Errors[k+1][j]  *=  (dOffset  +  Layers[k+1].Outputs[j]*(1-
Layers[k+1].Outputs[j]));
      }
      }
      }

      void Update (int k, double[] Inp)
      {
      int Index = 0;
      for (int i=0; i<Layers[k].LSize; i++)
      {
       LC[k]*Errors[k][i];

      MBias[k][i] = MC[k]*MBias[k][i] +

      Layers[k].Bias[i] += MBias[k][i];
      for (int j=0; j<Layers[k].prevLSize; j++)
      {

      Moments[k][Index] = MC[k]*Moments[k][Index] + LC[k]*Errors[k][i]*Inp[j];
      Layers[k].Weights[Index] +=

      Moments[k][Index];}   }  }
```

```
      ++Index;

      boolean ErrorPropagation (int iRecPointer)
      {
      int i, j; double dTmp;

      for (j=0; j<iRecordLength; j++)
      {
      pdInput[j] = pdInputBuffer[iRecPointer*iRecordLength+j];
      }
      int iTmpClass;
      iTmpClass = Test(pdInput);
      if (piClass[iRecPointer] == iTmpClass) return false; Errors[0][0] =
(piClass[iRecPointer] -
      Layers[0].Outputs[0])/Layers[0].dMaxNum;
      ErrorCalc(); Update(LNum-1,pdInput);
      for (i=LNum-1; 0<i; i--) Update(i- 1,Layers[i].Outputs);
      return true;
      }

      boolean WriteNet() throws IOException
      {
      GetWeights();
      File FNet = new File (NetFileName); ObjectOutputStream OS = new
ObjectOutputStream(new
      FileOutputStream(FNet));
      OS.writeObject(pdWeights); OS.close();
      return true;
      }

      public int Train (int FrameRate) throws IOException
      {
      int j, i = 0, iCount = 0, iErrCount = 0; while
((i<StopFlag)||(iErrCount==0.0))
      {



      : 0;

      if (ErrorPropagation(iCount)) ++iErrCount;
      ++iCount;
      iCount = (iCount<iRecordsNumber) ? iCount


      if (i==lCheckPoint)
      {for (j=0; j<LNum; j++)
      {
      LC[j] *= dTrFactor; MC[j] *= dTrFactor;
      }
      lCheckPoint =
      (long)(lCheckPoint/dTrFactor);}

      if (i%FrameRate == (FrameRate-1))
      {

      System.out.print((100.0*(double)iErrCount)/FrameRate + " ");
      iErrCount = 0;
      }
      ++i;}
      WriteNet(); return 0;
      }}
```