Probabilistische Algorithmen

Hans Marktwart und Mateusz Zwierz

31. Mai 2010

Gliederung des Vortrages

Einleitung

Einleitung

Gliederung des Vortrages

Einleitung

Einleitung

2.Zufallszahlen

2.1 Philosophie des Zufalls
2.2 Methoden zur Erzeugung von Zufallgrößen
2.3 Pseudozufallszahlen in Scheme und Java

Gliederung des Vortrages

Einleitung

Einleitung

2.Zufallszahlen

2.1 Philosophie des Zufalls
2.2 Methoden zur Erzeugung von Zufallgrößen
2.3 Pseudozufallszahlen in Scheme und Java

3. Probilistische Algorithmen

3.1 Was sind probabilistische Algorithmen?
3.2 Vorteile gegenüber den deterministischen Ansätzen
3.3 Anwendungsbereiche

Gliederung des Vortrages

Einleitung

Einleitung

2.Zufallszahlen

2.1 Philosophie des Zufalls
2.2 Methoden zur Erzeugung von Zufallgrößen
2.3 Pseudozufallszahlen in Scheme und Java

3. Probilistische Algorithmen

3.1 Was sind probabilistische Algorithmen?
3.2 Vorteile gegenüber den deterministischen Ansätzen
3.3 Anwendungsbereiche

4. Monte Carlo Methode

4.1 Grundprinzip
4.2 Beispiele in Scheme und Java

Gliederung des Vortrages

Einleitung

Einleitung

2.Zufallszahlen

2.1 Philosophie des Zufalls
2.2 Methoden zur Erzeugung von Zufallgrößen
2.3 Pseudozufallszahlen in Scheme und Java

3. Probilistische Algorithmen

3.1 Was sind probabilistische Algorithmen?
3.2 Vorteile gegenüber den deterministischen Ansätzen
3.3 Anwendungsbereiche

4. Monte Carlo Methode

4.1 Grundprinzip
4.2 Beispiele in Scheme und Java

Gliederung

5. Las Vegas

5.1 Grundprinzip
5.2 Beispiele

6. Ant Colony Optimization

6.1 Allgemeines
6.2 Verhalten von echten Ameisen
6.3 Verhalten von künstlichen Ameisen
6.4 Anwendungsbeispiele

Zufall

1. Eine Erscheinung ist zufällig, wenn sie ohne Plan eintritt
2. Epikur meinte, dass der Zufall die eigentliche Natur der Erscheinung ist
   - ist der Zufall wirklich objektiv?
3. Demokrit sagte, dass der Zufällige das Nichterkannte ist
4. Streitfrage seit der Antike

Zufall

1. Eine Erscheinung ist zufällig, wenn sie ohne Plan eintritt
2. Epikur meinte, dass der Zufall die eigentliche Natur der Erscheinung ist
    - ist der Zufall wirklich objektiv?
3. Demokrit sagte, dass der Zufällige das Nichterkannte ist
4. Streitfrage seit der Antike

Zufall

1. Eine Erscheinung ist zufällig, wenn sie ohne Plan eintritt
2. Epikur meinte, dass der Zufall die eigentliche Natur der Erscheinung ist
   - ist der Zufall wirklich objektiv?
3. Demokrit sagte, dass der Zufällige das Nichterkannte ist
4. Streitfrage seit der Antike

Zufall

1. Eine Erscheinung ist zufällig, wenn sie ohne Plan eintritt
2. Epikur meinte, dass der Zufall die eigentliche Natur der Erscheinung ist
   - ist der Zufall wirklich objektiv?
3. Demokrit sagte, dass der Zufällige das Nichterkannte ist
4. Streitfrage seit der Antike

Zufall

1. Eine Erscheinung ist zufällig, wenn sie ohne Plan eintritt
2. Epikur meinte, dass der Zufall die eigentliche Natur der Erscheinung ist
   - ist der Zufall wirklich objektiv?
3. Demokrit sagte, dass der Zufällige das Nichterkannte ist
4. Streitfrage seit der Antike

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)
3. rand() % 6 + 1

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)
3. rand() % 6 + 1
4. (random 5)

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)
3. rand() % 6 + 1
4. (random 5)
5. Random randomGenerator = new Random();
   int randomInt = randomGenerator.nextInt(100);

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)
3. rand() % 6 + 1
4. (random 5)
5. Random randomGenerator = new Random();
   int randomInt = randomGenerator.nextInt(100);
6. int i = (int) (Math.random()*3+1);

Generator

1. Random zufall = new Random();
   int x = zufall.Next(1, 13);
2. Randomize;
   Random(45)
3. rand() % 6 + 1
4. (random 5)
5. Random randomGenerator = new Random();
   int randomInt = randomGenerator.nextInt(100);
6. int i = (int) (Math.random()*3+1);

Zufallszahlen

1. Math.random();

Zufallszahlen

1. Math.random();
2. Math.floor();

Zufallszahlen

1. Math.random();
2. Math.floor();
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab

Zufallszahlen

1. Math.random();
2. Math.floor();
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab
3. Math.round();

Zufallszahlen

1. Math.random();
2. Math.floor();
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab
3. Math.round();
   - rundet kaufmänisch

Zufallszahlen

1. **Math.random();**
2. Math.floor();
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab
3. Math.round();
   - rundet kaufmänisch

Zufallszahlen

1. **Math.random();**
2. *Math.floor();*
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab
3. Math.round();
   - rundet kaufmänisch

Zufallszahlen

1. **Math.random();**
2. *Math.floor();*
   - rundet jede Zahl auf die nächstniedrige ganze Zahl ab
3. *Math.round();*
   - rundet kaufmänisch

Zufallszahlen

Ziel

zufällige Zahl aus dem Intervall [5;10]
NZ = { 5,6,7,8,9,10 }
Jede Zahl = gleiche Warscheinlichkeit der Erzeugung

Zufallszahlen

Quellcode

```
int min = 5;
int max = 10;
int erg = (int) (Math.random() * (max - min)) + min ;
```

Zufallszahlen

Quellcode

```
int min = 5;
int max = 10;
int erg = (int) (Math.random() * (max - min)) + min ;
```

Error

Zahlen aus dem Intervall [5;10[

Zufallszahlen

Math.round();

```
int erg =(int) Math.round(Math.random() * (max - min)) + min;
```

Zufallszahlen

Math.round();

```
int erg =(int) Math.round(Math.random() * (max - min)) + min;
```

Ergebnis

Zahlen aus dem Interval [5;10]

Zufallszahlen

Math.round();

```
int erg =(int) Math.round(Math.random() * (max - min)) + min;
```

Ergebnis

Zahlen aus dem Interval [5;10]

Error

die Warscheinlichkeit ist nicht gleichmäßig

Zufallszahlen

a = Math.random() * (10 - 5)
b = Math.round(a)
c = b + 5

a	[0;0.5[	[0.5 ;1[	[1;1.5[	[1.5;2[	[2;2.5[	...	[4.5;5[
b	0	1	1	2	2	...	5
c	5	6	6	7	7	...	10
W	$\frac{1}{10}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{10}$

Zufallszahlen

Math.floor

```
int x =(int) Math.floor(Math.random() * (max - min + 1)) + min;
```

Zufallszahlen

Math.floor

```
int x =(int) Math.floor(Math.random() * (max - min + 1)) + min;
```

Ergebnis

gleichverteilte Zufallszahlen aus dem Intervall [5;10]

Zufallszahlen

a = Math.random() * (10 - 5 +1)
b = Math.floor(a)
c = b + 5

a	[0;1[	[1;2[	[2;3[	[3;4[	[4;5[	[5;6[
b	0	1	2	3	4	5
c	5	6	7	8	9	10
W	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

Zufallszahlen

Methode 1

```
int rand (min, max) {
return (int)Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Methode 2

```
int rand (min, max) {
return (int)Math.round(Math.random() * (max - min )) + min; }
```

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. Lineare Kongruenzmethode
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. Lineare Kongruenzmethode
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. Spezialfall
   - $x_n = (a \cdot x_{n-1}) \bmod m$

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. Lineare Kongruenzmethode
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. Spezialfall
   - $x_n = (a \cdot x_{n-1}) \bmod m$
4. RANROT-W Pseudo-Zufallszahlen-Generator

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. Lineare Kongruenzmethode
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. Spezialfall
   - $x_n = (a \cdot x_{n-1}) \bmod m$
4. RANROT-W Pseudo-Zufallszahlen-Generator
5. Mother-of-All Pseudo-Zufallszahlen-Generator

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. Lineare Kongruenzmethode
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. Spezialfall
   - $x_n = (a \cdot x_{n-1}) \bmod m$
4. RANROT-W Pseudo-Zufallszahlen-Generator
5. Mother-of-All Pseudo-Zufallszahlen-Generator
6. twister Pseudo-Zufallszahlen-Generator

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. **Lineare Kongruenzmethode**
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. Spezialfall
   - $x_n = (a \cdot x_{n-1}) \bmod m$
4. RANROT-W Pseudo-Zufallszahlen-Generator
5. Mother-of-All Pseudo-Zufallszahlen-Generator
6. twister Pseudo-Zufallszahlen-Generator

Zufallsgeneratoren

1. Berechnung der Zufallszahlen ist deterministisch
2. **Lineare Kongruenzmethode**
   - $x_n = (a \cdot x_{n-1} + b) \bmod m$
3. **Spezialfall**
   - $x_n = (a \cdot x_{n-1}) \bmod m$
4. RANROT-W Pseudo-Zufallszahlen-Generator
5. Mother-of-All Pseudo-Zufallszahlen-Generator
6. twister Pseudo-Zufallszahlen-Generator

Zufallsgeneratoren

Kongruenzmethode

$x_n = (a \cdot x_{n-1} + b) \bmod m$

x_n ... Startwert (Seed, Keim)

m ... 2^n - 1 oder 2^n

Seed ... fest — gleiche Periode von Zufallszahlen

Seed ... zufällig — unterschiedliche Periode von Zufallszahlen

Zufallsgeneratoren

Kongruenzmethode

$x_n = (a \cdot x_{n-1} + b) \bmod m$

x_n ... Startwert (Seed, Keim)

m ... 2^n - 1 oder 2^n

Seed ... fest — gleiche Periode von Zufallszahlen

Seed ... zufällig — unterschiedliche Periode von Zufallszahlen

System.currentTimeMillis();