


小帆的帆的专栏

目录视图

摘要视图

RSS 订阅


个人资料



小帆的帆

访问: 50135次

积分: 900

等级: 

排名: 千里之外

原创: 37篇

转载: 30篇

译文: 1篇

评论: 1条

文章搜索

文章分类

Spark (8)

Android (54)

软件推荐 (1)

Mac (3)

深度学习 (3)

文章存档

2016年09月 (1)

2016年08月 (1)

2016年07月 (1)

2016年06月 (8)

2014年03月 (2)

展开

阅读排行

Accuracy(准确率), Precision (8572)

离线安装Android SDK (2430)

快如闪电的Android模拟器 (2373)

Spark RDD、DataFrame (2358)

Spinner显示全国省市 (1251)

Android流量监控 (1230)

Canvas画虚线 (842)

移动信息安全的漏洞和逆向原理

【观点】世界上最好的语言是什么

【知识库】50个精品领域内容，一键直达

晒知识图谱，享技术荣誉

Spark 线性代数库 Breeze API 详解

标签: Spark MLlib Breeze

2016-06-21 16:55 799人阅读 评论(0) 收藏 举报

分类: Spark (7)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?) [+]

转载请标明出处：[小帆的帆的专栏](#)

运算

加，减，乘，除

向量与向量

• 加：+

• 减：-

• 乘：.*

• 除：./

规则1：乘除前面，加冒号；单独的乘号和除号分别表示点积和线性求解

规则2：累加效果，加等号

1 import breeze.linalg.DenseVector

2

3 object Test {

4 def main(args: Array[String]) {

5 val v1 = DenseVector(1.0, 2.0, 3.0, 4.0)

6 val v2 = DenseVector(0.5, 0.5, 0.5, 0.5)

7 // DenseVector(1.5, 2.5, 3.5, 4.5)

8 println("\nv1 + v2 : ")

9 println(v1 + v2)

10

11 // DenseVector(0.5, 1.5, 2.5, 3.5)

12 println("\nv1 - v2 : ")

13 println(v1 - v2)

14

15 // DenseVector(0.5, 1.0, 1.5, 2.0)

16 println("\nv1 .* v2 : ")

17 // 规则1：乘号前面多了冒号

18 println(v1 .* v2)

19

20 // DenseVector(2.0, 4.0, 6.0, 8.0)

21 println("\nv1 ./ v2 : ")

22 // 规则1：除号前面多了冒号

23 println(v1 ./ v2)

24

Spark 线性代数库 Breeze (796)

结合Spark源码分析, com (696)

Error executing aapt: Re (695)

评论排行

结合Spark源码分析, com (1)

返回键点击触发, 回到桌 (0)

自定义Dialog (0)

Dialog中的点击Positive (0)

斜杠与反斜杠 (0)

换行符 (0)

文件续写 (0)

IOException的简单处理 (0)

CursorAdapter (0)

神经网络-前向传播 (0)

推荐文章

* 程序员10月书讯, 评论得书

* Android中Xposed框架篇---修改系统位置信息实现自身隐藏功能

* Chromium插件（Plugin）模块（Module）加载过程分析

* Android TV开发总结--构建一个TV app的直播节目实例

* 架构设计：系统存储--MySQL简单主从方案及暴露的问题

最新评论

结合Spark源码分析, combineBy William-若寒: 感谢分享


400号码选号 高空作业车价格


高空作业车厂家 大兴新

```
25 // 但是v1 和 v2并没有改变
26 // DenseVector(1.0, 2.0, 3.0, 4.0)
27 println("\nv1 : ")
28 println(v1)
29
30 // DenseVector(0.5, 0.5, 0.5, 0.5)
31 println("\nv2 : ")
32 println(v2)
33
34 // 规则2
35 // 如果想把最后的结果保存到v1上, 需要加等号
36 // DenseVector(1.5, 2.5, 3.5, 4.5)
37 println("\nv1 += v2 : ")
38 println(v1 += v2)
39
40 // DenseVector(1.0, 2.0, 3.0, 4.0)
41 println("\nv1 -= v2 : ")
42 println(v1 -= v2)
43
44 // DenseVector(0.5, 1.0, 1.5, 2.0)
45 println("\nv1 *= v2 : ")
46 // 注意: 乘号前面多了冒号
47 println(v1 *= v2)
48
49 // DenseVector(1.0, 2.0, 3.0, 4.0)
50 println("\nv1 /= v2 : ")
51 // 注意: 除号前面多了冒号
52 println(v1 /= v2)
53 }
54 }
```

矩阵与矩阵

跟向量与向量完全一样

- 加: +
- 减: -
- 乘: *
- 除: /

规则1: 乘除前面, 加冒号; 单独的乘号和除号分别表示点积和线性求解

规则2: 累加效果, 加等号

```
1 import breeze.linalg.DenseMatrix
2
3 object Test1 {
4   def main(args: Array[String]) {
5     val m1 = DenseMatrix((1.0, 2.0), (3.0, 4.0))
6     val m2 = DenseMatrix((0.5, 0.5), (0.5, 0.5))
7
8     /**
9      * 1.5 2.5
10     * 3.5 4.5
11     */
12     println("\nm1 + m2 : ")
13     println(m1 + m2)
14
15     /**
16     * 0.5 1.5
17     * 2.5 3.5
18     */
19     println("\nm1 - m2 : ")
20     println(m1 - m2)
21
22     /**
23     * 0.5 1.0
24     * 1.5 2.0
25     */
26     println("\nm1 *= m2 : ")
```

```
27 // 注意：乘号前面多了冒号
28 println(m1 :* m2)
29
30 /**
31  * 2.0  4.0
32  * 6.0  8.0
33  */
34 println("\nm1 :/ m2 : ")
35 // 注意：除号前面多了冒号
36 println(m1 :/ m2)
37
38 // 但是m1 和 m2并没有改变
39 /**
40  * 1.0  2.0
41  * 3.0  4.0
42  */
43 println("\nm1 : ")
44 println(m1)
45
46 /**
47  * 0.5  0.5
48  * 0.5  0.5
49  */
50 println("\nm2 : ")
51 println(m2)
52
53 // 如果想把最后的结果保存到m1上，需要加等号
54 /**
55  * 1.5  2.5
56  * 3.5  4.5
57  */
58 println("\nm1 += m2 : ")
59 println(m1 += m2)
60
61 /**
62  * 1.0  2.0
63  * 3.0  4.0
64  */
65 println("\nm1 -= m2 : ")
66 println(m1 -= m2)
67
68 /**
69  * 0.5  1.0
70  * 1.5  2.0
71  */
72 println("\nm1 :*= m2 : ")
73 // 注意：乘号前面多了冒号
74 println(m1 :*= m2)
75
76 /**
77  * 1.0  2.0
78  * 3.0  4.0
79  */
80 println("\nm1 :/= m2 : ")
81 // 注意：除号前面多了冒号
82 println(m1 :/= m2)
83 }
84 }
```

矩阵或向量与数值

- 加：+
- 减：-
- 乘：*
- 除：/

规则1：累加效果，加等号

注意：乘除号前不需要冒号，因为没有矩阵与数值的点积等计算

```
1 import breeze.linalg.{DenseMatrix, DenseVector}
2
3 object Test1 {
4   def main(args: Array[String]) {
5     val v1 = DenseVector(1.0, 2.0, 3.0, 4.0)
6
7     // DenseVector(1.5, 2.5, 3.5, 4.5)
8     println("v1 + 0.5 : ")
9     println(v1 + 0.5)
10
11    // DenseVector(0.5, 1.5, 2.5, 3.5)
12    println("\nv1 - 0.5 : ")
13    println(v1 - 0.5)
14
15    // DenseVector(0.5, 1.0, 1.5, 2.0)
16    println("\nv1 * 0.5 : ")
17    println(v1 * 0.5)
18
19    // DenseVector(2.0, 4.0, 6.0, 8.0)
20    println("\nv1 / 0.5 : ")
21    println(v1 / 0.5)
22
23    // v1依然不变
24    // DenseVector(1.0, 2.0, 3.0, 4.0)
25    println("\nv1 : ")
26    println(v1)
27
28    // DenseVector(1.5, 2.5, 3.5, 4.5)
29    println("\nv1 += 0.5 : ")
30    println(v1 += 0.5)
31
32    // DenseVector(1.0, 2.0, 3.0, 4.0)
33    println("\nv1 -= 0.5 : ")
34    println(v1 -= 0.5)
35
36    // DenseVector(0.5, 1.0, 1.5, 2.0)
37    println("\nv1 *= 0.5 : ")
38    println(v1 *= 0.5)
39
40    // DenseVector(5.0, 10.0, 15.0, 20.0)
41    println("\nv1 /= 0.1 : ")
42    println(v1 /= 0.1)
43
44    // DenseVector(5.0, 10.0, 15.0, 20.0)
45    println("\nv1 : ")
46    println(v1)
47
48    val m1 = DenseMatrix((1.0, 2.0), (3.0, 4.0))
49
50    /**
51     * 1.5  2.5
52     * 3.5  4.5
53     */
54    println("m1 + 0.5 : ")
55    println(m1 + 0.5)
56
57    /**
58     * 0.5  1.5
59     * 2.5  3.5
60     */
61    println("\nm1 - 0.5 : ")
62    println(m1 - 0.5)
63
64    /**
65     * 0.5  1.0
66     * 1.5  2.0
67     */
68    println("\nm1 * 0.5 : ")
69    println(m1 * 0.5)
70
71    /**
```

```

72      * 2.0  4.0
73      * 6.0  8.0
74      */
75      println("\nm1 / 0.5 : ")
76      println(m1 / 0.5)
77
78      // m1依然不变
79      /**
80      * 1.0  2.0
81      * 3.0  4.0
82      */
83      println("\nm1 : ")
84      println(m1)
85
86      /**
87      * 1.5  2.5
88      * 3.5  4.5
89      */
90      println("\nm1 += 0.5 : ")
91      println(m1 += 0.5)
92
93      /**
94      * 1.0  2.0
95      * 3.0  4.0
96      */
97      println("\nm1 -= 0.5 : ")
98      println(m1 -= 0.5)
99
100     /**
101     * 0.5  1.0
102     * 1.5  2.0
103     */
104     println("\nm1 *= 0.5 : ")
105     println(m1 *= 0.5)
106
107     /**
108     * 5.0  10.0
109     * 15.0 20.0
110     */
111     println("\nm1 /= 0.1 : ")
112     println(m1 /= 0.1)
113
114     /**
115     * 5.0  10.0
116     * 15.0 20.0
117     */
118     println("\nm1 : ")
119     println(m1)
120   }
121 }

```

矩阵与向量

- 加：+
- 减：-
- 乘：.*
- 除：./

规则1: 乘除前面，加冒号；单独的乘号和除号分别表示点积和线性求解

规则2: 累加效果，加等号

规则3: 必须有星号

规则4: 星号在左，逐行；星号在右，逐列；与向量是列向量还是行向量无关

规则5: 向量必须是列向量

```

1  import breeze.linalg.{*, DenseMatrix, DenseVector}
2
3  object Test1 {
4      def main(args: Array[String]) {

```

```

5
6     val m1 = DenseMatrix(
7         (1.0, 2.0),
8         (3.0, 4.0)
9     )
10
11     val v1 = DenseVector(1.0, 2.0)
12     //     val v1 = DenseVector(1.0, 2.0).t // 运行时异常, 规则5, 向量必须是列向量
13     //     val v1 = DenseVector(1.0, 2.0).t.t // 正确, 如果是一个列向量, 需要转换成行向量
14
15     // 规则4: 星号在左, 逐行; 星号在右, 逐列
16     println("-----星号在左边, 就逐行操作-----")
17
18     /**
19      * 2.0  4.0
20      * 4.0  6.0
21      */
22     println("\nm1(*, :) + v1 : ")
23     println(m1(*, :) + v1)
24
25     /**
26      * 0.0  0.0
27      * 2.0  2.0
28      */
29     println("\nm1(*, :) - v1 : ")
30     println(m1(*, :) - v1)
31
32     // 规则1: 乘除前面, 加冒号
33
34     /**
35      * 1.0  4.0
36      * 3.0  8.0
37      */
38     println("\nm1(*, :) :* v1 : ")
39     println(m1(*, :) :* v1)
40
41     /**
42      * 1.0  1.0
43      * 3.0  2.0
44      */
45     println("\nm1(*, :) :/ v1 : ")
46     println(m1(*, :) :/ v1)
47
48     println("-----星号在右边, 就逐列操作-----")
49
50     /**
51      * 2.0  3.0
52      * 5.0  6.0
53      */
54     println("\nm1(:, *) + v1 : ")
55     println(m1(:, *) + v1)
56
57     /**
58      * 0.0  1.0
59      * 1.0  2.0
60      */
61     println("\nm1(:, *) - v1 : ")
62     println(m1(:, *) - v1)
63
64     /**
65      * 1.0  2.0
66      * 6.0  8.0
67      */
68     println("\nm1(:, *) :* v1 : ")
69     println(m1(:, *) :* v1)
70
71     /**
72      * 1.0  2.0
73      * 1.5  2.0
74      */
75     println("\nm1(:, *) :/ v1 : ")

```

```

76     println(m1(:, *) :/ v1)
77
78     // 无论星号在哪, m1都不会改变
79     println("-----规则2: 累加效果, 加等号-----")
80
81     // 下面以整除为例, 注意星号的位置, 一个在左, 一个在右
82
83     /**
84      * 1.0  2.0
85      * 3.0  4.0
86      */
87     println("\nm1 : ")
88     println(m1)
89
90     /**
91      * BroadcastedRows(1.0  4.0
92      * 3.0  8.0 )
93      */
94     println("\nm1(*, :) :*= v1 : ")
95     println(m1(*, :) :*= v1)
96
97     /**
98      * 1.0  4.0
99      * 3.0  8.0
100     */
101     println("\nm1 : ")
102     println(m1)
103
104     /**
105      * BroadcastedColumns(1.0  4.0
106      * 1.5  4.0 )
107     */
108     println("\nm1(:, *) :/= v1 : ")
109     println(m1(:, *) :/= v1)
110
111     /**
112      * 1.0  4.0
113      * 1.5  4.0
114     */
115     println("\nm1 : ")
116     println(m1)
117   }
118 }

```

函数

统计

求和

```

1  import breeze.linalg.{Axis, DenseMatrix, sum}
2
3  object Test1 {
4    def main(args: Array[String]) {
5
6      val m1 = DenseMatrix(
7        (1.0, 2.0),
8        (3.0, 4.0)
9      )
10
11      // Axis._0 纵向
12      // 4.0  6.0
13      println(sum(m1, Axis._0))
14
15      // Axis._1 横向
16      // 3.0 7.0
17      println(sum(m1, Axis._1))
18    }
19  }

```

均值

```
1 import breeze.linalg.{Axis, DenseMatrix}
2 import breeze.stats.mean
3
4 object Test1 {
5   def main(args: Array[String]) {
6
7     val m1 = DenseMatrix(
8       (1.0, 2.0),
9       (3.0, 4.0)
10    )
11
12    // Axis._0 纵向
13    // 4.0 6.0
14    println(mean(m1, Axis._0))
15
16    // Axis._1 横向
17    // 1.5 3.5
18    println(mean(m1, Axis._1))
19  }
20 }
```

方差和标准差

```
1 import breeze.linalg.{Axis, DenseMatrix}
2 import breeze.stats.{stddev, variance}
3
4 object Test1 {
5   def main(args: Array[String]) {
6
7     val m1 = DenseMatrix(
8       (1.0, 2.0),
9       (3.0, 4.0)
10    )
11
12    // Axis._0 纵向
13    // 2.0 2.0
14    println(variance(m1, Axis._0))
15
16    // Axis._1 横向
17    // 0.5, 0.5
18    println(variance(m1, Axis._1))
19
20    // Axis._0 纵向
21    // 2.0 2.0
22    println(stddev(m1, Axis._0))
23
24    // Axis._1 横向
25    // 0.5, 0.5
26    println(stddev(m1, Axis._1))
27  }
28 }
```

N次方和开方

```
1 import breeze.linalg.DenseMatrix
2 import breeze.numerics.{pow, sqrt}
3
4 object Test1 {
5   def main(args: Array[String]) {
6
7     val m1 = DenseMatrix(
8       (1.0, 2.0),
9       (3.0, 4.0)
10    )
11
12    /**
13     * 1.0 4.0
14     * 9.0 16.0
15     */
16  }
```



```

15      */
16      println(pow(m1, 2))
17
18      /**
19      * 1.0      8.0
20      * 27.0    64.0
21      */
22      println(pow(m1, 3))
23
24      /**
25      * 1.0                      1.4142135623730951
26      * 1.7320508075688772    2.0
27      */
28      println(sqrt(m1))
29    }
30  }

```

E和log

```

1  import breeze.linalg.DenseMatrix
2  import breeze.numerics.{exp, log, log10, log1p}
3
4  object Test1 {
5    def main(args: Array[String]) {
6
7      val m1 = DenseMatrix(
8        (1.0, 2.0),
9        (3.0, 4.0)
10     )
11
12     /**
13     * 2.718281828459045    7.38905609893065
14     * 20.085536923187668  54.598150033144236
15     */
16     // e = 2.718281828459045
17     println(exp(m1))
18
19     /**
20     * 0.0                      0.6931471805599453
21     * 1.0986122886681098    1.3862943611198906
22     */
23     // 以e为底
24     println(log(m1))
25
26     /**
27     * 0.0                      0.3010299956639812
28     * 0.47712125471966244   0.6020599913279624
29     */
30     // 以10为底
31     println(log10(m1))
32
33     /**
34     * 0.6931471805599453    1.0986122886681096
35     * 1.3862943611198906    1.6094379124341003
36     */
37     // 以e为底
38     // log1p() 以返回 log(1 + x)，甚至当 x 的值接近零也能计算出准确结果。
39     println(log1p(m1))
40   }
41 }

```

三角

- sin, sinh, asin, asinh
- cos, cosh, acos, acosh
- tan, tanh, atan, atanh
- atan2
- sinc(x) == sin(x)/x
- sincpi(x) == sinc(x * Pi)

取整

```
1 import breeze.linalg.DenseVector
2 import breeze.numerics._
3
4 object Test1 {
5   def main(args: Array[String]) {
6     val a = DenseVector(1.4, 0.5, -2.3)
7
8     // 四舍五入
9     println(round(a))
10
11    // 向上取整
12    println(ceil(a))
13
14    // 向下取整
15    println(floor(a))
16
17    // 大于0, 为1; 小于0, 为-1
18    println(signum(a))
19
20    // 绝对值
21    println(abs(a))
22  }
23 }
```

示例

模拟逻辑回归

利用Breeze进行，归一化，添加截距项，预测

```
1 import breeze.linalg._
2 import breeze.numerics._
3 import breeze.stats._
4
5 object Work2 {
6   def main(args: Array[String]) {
7
8     // 随机产生数据
9     // val featuresMatrix = DenseMatrix.rand[Double](3, 3)
10    // val labelMatrix = DenseMatrix.rand[Double](3, 1)
11
12    // 测试数据
13    val featuresMatrix = DenseMatrix(
14      (1.0, 2.0, 3.0),
15      (4.0, 5.0, 6.0),
16      (7.0, 8.0, 9.0)
17    )
18
19    val labelMatrix = DenseMatrix(
20      1.0,
21      1.0,
22      0.0
23    )
24
25    // 均值
26    // DenseVector(4.0, 5.0, 6.0)
27    val featuresMean = mean(featuresMatrix(:, *)).toDenseVector
28    println("均值: ")
29    println(featuresMean)
30
31    // 标准差
32    // DenseVector(3.0, 3.0, 3.0)
33    val featuresStddev = stddev(featuresMatrix(:, *)).toDenseVector
34    println("\n标准差: ")
35    println(featuresStddev)
36
37    // 减去均值
```

```
38  /**
39  * -3.0 -3.0 -3.0
40  * 0.0 0.0 0.0
41  * 3.0 3.0 3.0
42  */
43  featuresMatrix(*, ::) -= featuresMean
44  println("\n减去均值: ")
45  println(featuresMatrix)
46
47  // 除以标准差
48  /**
49  * -1.0 -1.0 -1.0
50  * 0.0 0.0 0.0
51  * 1.0 1.0 1.0
52  */
53  featuresMatrix(*, ::) /= featuresStddev
54  println("\n除以标准差: ")
55  println(featuresMatrix)
56
57  // 生成截距
58  /**
59  * 1.0
60  * 1.0
61  * 1.0
62  */
63  val intercept = DenseMatrix.ones[Double](featuresMatrix.rows, 1)
64  println("\n截距: ")
65  println(intercept)
66
67  // 拼接成为最终的训练集
68  /**
69  * 1.0 -1.0 -1.0 -1.0
70  * 1.0 0.0 0.0 0.0
71  * 1.0 1.0 1.0 1.0
72  */
73  val train = DenseMatrix.horzcat(intercept, featuresMatrix)
74  println("\n训练集: ")
75  println(train)
76
77  // 参数
78  // 为方便检查结果, 这里全部设置为1
79  /**
80  * 1.0
81  * 1.0
82  * 1.0
83  * 1.0
84  */
85  val w = new DenseMatrix(4, 1, Array(1.0, 1.0, 1.0, 1.0))
86  // val w = DenseMatrix.rand[Double](4, 1) // 随机生成, 一定要指定类型
87  println("\n参数: ")
88  println(w)
89
90  /**
91  * -2.0
92  * 1.0
93  * 4.0
94  */
95  // 随机生成w时, 如果没有指定类型, A的计算结果虽然不会有错, 但是后面将无法计算, 除非通过a:
96  // 如果w指定了类型, 那么在idea中, 转换语句会是灰色的, 意思是这句话没有作用, 可以不写
97  val A = (train * w).asInstanceOf[DenseMatrix[Double]]
98  println("\nA: ")
99  println(A)
100
101  /**
102  * 0.11920292202211755
103  * 0.7310585786300049
104  * 0.9820137900379085
105  */
106  // Sigmoid函数
107  val probability = 1.0 / (exp(A * -1.0) + 1.0)
108  println("\nprobability: ")
```

```
109     println(probability)
110
111     /**
112      * MSE : 0.6041613548425021
113      */
114     val MSE = mean(pow(probability - labelMatrix, 2))
115     println("\nMSE: ")
116     println(MSE)
117
118     /**
119      * RMSE : 0.777278170825929
120      */
121     val RMSE = sqrt(MSE)
122     println("\nRMSE: ")
123     println(RMSE)
124 }
125
126 }
```

参考

- Quickstart
- Linear Algebra Cheat Sheet
- 炼数成金-黄美灵老师的Spark MLlib 机器学习算法与源码解析课程

顶

0

踩

0

上一篇

关联规则、支持度 (support)、置信度 (confidence)、并运用Spark RDD计算

下一篇

Spark大规模机器学习的性能瓶颈和解决方案

我的同类文章

Spark (7)

• Spark大规模机器学习的性...

2016-06-21

阅读 178

• Introducing Apache Spark ...

2016-06-16

阅读 218

• 结合Spark源码分析, combin...

2016-06-16

阅读 698

• Spark RDD、DataFrame和...

2016-06-16

阅读 2372

• 关联规则、支持度 (support...

2016-06-17

阅读 647

• Accuracy(准确率), Precisio...

2016-06-16

阅读 8579

• Spark join和cogroup算子

2016-06-16

阅读 519

参考知识库



机器学习知识库
5105 关注 | 308 收录



Apache Spark知识库
2844 关注 | 257 收录

猜你在找

- Spark 1.x大数据平台
- 阿里云机器学习算法应用实践
- Spark零基础入门 (1) : Scala基本数据类型及程序控#
- Spark零基础入门 (2) : 解析Scala集合操作
- 统计机器学习入门——线性模型选择与正则化2
- C++线性代数库Armadillo
- Eigen线性代数运算的C++模板库
- windows下MinGW编译cblas基本线性代数库
- 数值计算 高等数学 线性代数 免费库
- 线性代数导论17正交矩阵和Gram-Schmidt正交化



广告

[查看评论](#)

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop
- AWS
- 移动游戏
- Java
- Android
- iOS
- Swift
- 智能硬件
- Docker
- OpenStack
- VPN
- Spark
- ERP
- IE10
- Eclipse
- CRM
- JavaScript
- 数据库
- Ubuntu
- NFC
- WAP
- jQuery
- BI
- HTML5
- Spring
- Apache
- .NET
- API
- HTML
- SDK
- IIS
- Fedora
- XML
- LBS
- Unity
- Splashtop
- UML
- components
- Windows Mobile
- Rails
- QEMU
- KDE
- Cassandra
- CloudStack
- FTC
- coremail
- OPhone
- CouchBase
- 云计算
- iOS6
- Rackspace
- Web App
- SpringSide
- Maemo
- Compuware
- 大数据
- aptech
- Perl
- Tornado
- Ruby
- Hibernate
- ThinkPHP
- HBase
- Pure
- Solr
- Angular
- Cloud Foundry
- Redis
- Scala
- Django
- Bootstrap

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [银行汇款帐号](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) [webmaster@csdn.net](#) 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | [江苏知之为计算机有限公司](#) |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved 