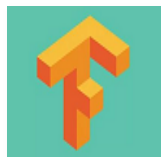


【python数据挖掘课程】二十.KNN最近邻分类算法分析详解及平衡秤TXT数据集读取

原创 Eastmount 最后发布于2017-12-08 00:15:43 阅读数 6317 ☆ 收藏

展开



Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相...



Eastmount

¥9.90

去订阅

这是《Python数据挖掘课程》系列文章，也是我这学期上课的部分内容及书籍的一个案例。本文主要讲述KNN最近邻分类算法、简单实现分析平衡秤数据集，希望这篇文章对大家有所帮助，同时提供些思路。内容包括：

- 1.KNN算法基础原理知识
- 2.最近邻分类算法分析预测坐标类型
- 3.Pandas读取TXT数据集
- 4.KNN分析平衡秤数据集
- 5.算法优化

本篇文章为基础性文章，希望对你有所帮助，如果文章中存在错误或不足支持，还请海涵~同时，推荐大家阅读我以前的文章了解基础知识。自己真的太忙了，只能挤午休或深夜的时间学习新知识，但每次写文内心都非常享受。

前文参考：

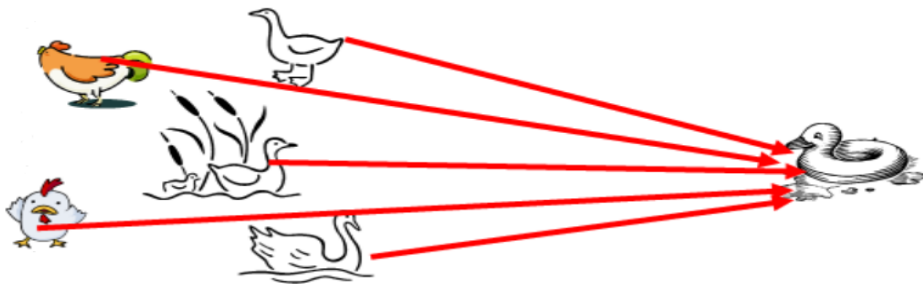
- 【Python数据挖掘课程】一.安装Python及爬虫入门介绍
- 【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
- 【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
- 【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
- 【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
- 【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
- 【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
- 【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐
- 【Python数据挖掘课程】九.回归模型LinearRegression简单分析氧化物数据
- 【python数据挖掘课程】十.Pandas、Matplotlib、PCA绘图实用代码补充
- 【python数据挖掘课程】十一.Pandas、Matplotlib结合SQL语句可视化分析
- 【python数据挖掘课程】十二.Pandas、Matplotlib结合SQL语句对比图分析
- 【python数据挖掘课程】十三.WordCloud词云配置过程及词频分析

- 【python数据挖掘课程】十四.Scipy调用curve_fit实现曲线拟合
- 【python数据挖掘课程】十五.Matplotlib调用imshow()函数绘制热图
- 【python数据挖掘课程】十六.逻辑回归LogisticRegression分析鸢尾花数据
- 【python数据挖掘课程】十七.社交网络Networkx库分析人物关系（初识篇）
- 【python数据挖掘课程】十八.线性回归及多项式回归分析四个案例分享
- 【python数据挖掘课程】十九.鸢尾花数据集可视化、线性回归、决策树花样分析

一. KNN算法基础原理知识

K最近邻（K-Nearest Neighbor，简称KNN）分类算法是数据挖掘分类技术中最简单常用的方法之一。所谓K最近邻，就是寻找K个最近的邻居的意思，每个样本都可以用它最接近的K个邻居来代表。本小节主要讲解KNN分类算法的基础知识及分析实例。

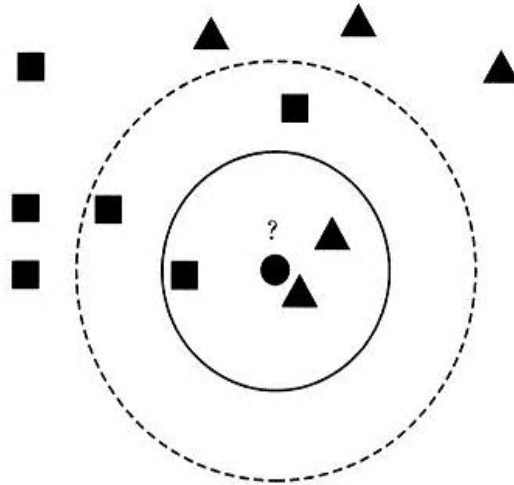
KNN分类算法是最近邻算法，字面意思就是寻找最近邻居，由Cover和Hart在1968年提出，简单直观易于实现。下面通过一个经典的例子来讲解如何寻找邻居，选取多少个邻居。下图是非常经典的KNN案例，需要判断右边这个动物是鸭子、鸡还是鹅？它涉及到了KNN算法的核心思想，判断与这个样本点相似的类别，再预测其所属类别。由于它走路和叫声像一只鸭子，所以右边的动物很可能是一只鸭子。



所以，KNN分类算法的核心思想是从训练样本中寻找所有训练样本X中与测试样本距离

（欧氏距离）最近的前K个样本（作为相似度），再选择与待分类样本距离最小的K个样本作为X的K个最邻近，并检测这K个样本大部分属于哪一类样本，则认为这个测试样本类别属于这一类样本。

假设现在需要判断下图中的圆形图案属于三角形还是正方形类别，采用KNN算法分析如下：



1.当K=3时，图中第一个圈包含了三个图形，其中三角形2个，正方形一个，该圆的则分类结果为三角形。

2.当K=5时，第二个圈中包含了5个图形，三角形2个，正方形3个，则以3:2的投票结果预测圆为正方形类标。

总之，设置不同的K值，可能预测得到不同的结果。

二. 最近邻分类算法分析预测坐标类型

KNN分类算法的具体步骤如下：

- 1.计算测试样本点到所有样本点的欧式距离dist，采用勾股定理计算。
- 2.用户自定义设置参数K，并选择离带测点最近的K个点。
- 3.从这K个点中，统计各个类型或类标的个数。
- 4.选择出现频率最大的类标号作为未知样本的类标号，反馈最终预测结果。

KNN在Sklearn机器学习包中，实现的类是neighbors.KNeighborsClassifier，简称KNN算法。构造方法为：

```
KNeighborsClassifier(algorithm='ball_tree',
                      leaf_size=30,
                      metric='minkowski',
                      metric_params=None,
                      n_jobs=1,
                      n_neighbors=3,
                      p=2,
                      weights='uniform')
```

KNeighborsClassifier可以设置3种算法：brute、kd_tree、ball_tree，设置K值参数为n_neighbors=3。

调用方法如下：

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3, algorithm="ball_tree")
```

它也包括两个方法：

训练：nbs.fit(data, target)

预测：pre = clf.predict(data)

下面这段代码是简单调用KNN分类算法进行预测的例子，代码如下。

```
# -*- coding: utf-8 -*-
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

X = np.array([[ -1, -1], [-2, -2], [1, 2], [1, 1], [-3, -4], [3, 2]])
Y = [0, 0, 1, 1, 0, 1]
x = [[4, 5], [-4, -3], [2, 6]]
knn = KNeighborsClassifier(n_neighbors=3, algorithm="ball_tree")
knn.fit(X, Y)
pre = knn.predict(x)
print pre
```

定义了一个二维数组用于存储6个点，其中x和y坐标为负数的类标定义为0，x和y坐标为正数的类标定义为1。调用knn.fit(X,Y)函数训练模型后，再调用predict()函数预测[4,5]、[-4,-3]、[2,6]三个点的坐标，输出结果分别为：[1, 0, 1]，其中x和y坐标为正数的划分为一类，负数的一类。

解释：它相当于分别计算[4,5]点到前面X变量六个点的距离，采用欧式距离，然后选择前3个（K=3）最近距离的点，看这三个点中属于0和1类的个数，则该[4,5]点预测的类型则属于较多的那个类别，即为1（正数）。

同时也可以计算K个最近点的下标和距离，代码和结果如下，返回距离k个最近的点和距离指数，indices可以理解为表示点的下标，distances为距离。

```
distances, indices = knn.kneighbors(X)
print indices
print distances

>>>
[[1 0 1]
 [[0 1 3]
  [1 0 4]
  [2 3 5]
  [3 2 5]
  [4 1 0]
  [5 2 3]]
 [[ 0.          1.41421356  2.82842712]
 [ 0.          1.41421356  2.23606798]
 [ 0.           1.          2.          ]
 [ 0.           1.          2.23606798]
 [ 0.          2.23606798  3.60555128]
 [ 0.           2.          2.23606798]]
>>>
```

KNN分类算法存在的优点包括：算法思路较为简单，易于实现；当有新样本要加入训练集中时，无需重新训练，即重新训练的代价低；计算时间和空间线性于训练集的规模。其缺点主要表现为分类速度慢，由于每次新的待分样本都必须与所有训练集一同计算比较相似度，以便取出靠前的K个已分类样本。整个算法的时间复杂度可以用 $O(m*n)$ 表示，其中m是选出的特征项(属性)的个数，而n是训练集样本的个数。同时，各属性的权重相同，影响了准确率，K值不好确定等会影响实验结果。


下面通过一个完整的实例结合可视化技术进行讲解，加深同学们的印象。

三. Pandas读取TXT数据集

作者最早想分析电影信息，如下图所示，根据电影中出现的打斗次数或亲吻次数来判断电影的类型，是属于动作片、爱情片还是科幻片，但是无赖数据集不好构造，这里修改为KNN分析平衡表数据集，但是其分析原理都是类似的，希望对您有所帮助。

电影名称	打斗次数	接吻次数	电影类型
California Man	3	104	Romance
He's Not Really into Dudes	2	100	Romance
Beautiful Woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action


该数据集来自于UCI网络公开数据集，成为Balance Scale Dataset平衡表数据集。
下载地址为：<http://archive.ics.uci.edu/ml/datasets/Balance+Scale>。



Balance Scale Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Balance scale weight & distance database



Data Set Characteristics:	Multivariate	Number of Instances:	625	Area:	Social
Attribute Characteristics:	Categorical	Number of Attributes:	4	Date Donated	1994-04-22
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	157064

Source:

Generated to model psychological experiments reported by Siegler, R. S. (1976). Three Aspects of Cognitive Development. Cognitive Psychology, 8, 481-520.

数据集主要来自于平衡秤的重量和距离相关数据，共625个样本，4个特征。这个数据集被生成来模拟心理实验结果。每个例子被分类为具有平衡尺度尖端向右，向左倾斜或平衡。属性是左侧重量，左侧距离，右侧重量和右侧距离。找到类的正确方法是（左距离*左权重）和（右距离*右权重）中的较大者。如果他们平等，就是平衡的。属性如下表所示：

列名	说明	值描述
Class Name	类名，三类：平衡、左边重、右边重	3 类，包括 L、B、R
Left Weight	左边重量	5 个等级，1-5
Left Distance	左边距离	5 个等级，1-5
Right Weight	右边重量	5 个等级，1-5
Right Distance	右边距离	5 个等级，1-5

下载数据集至本地data.txt文件，如下图所示。

data.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
B, 1, 1, 1, 1
R, 1, 1, 1, 2
R, 1, 1, 1, 3
R, 1, 1, 1, 4
R, 1, 1, 1, 5
R, 1, 1, 2, 1
R, 1, 1, 2, 2
R, 1, 1, 2, 3
R, 1, 1, 2, 4
```

这里作者采用Numpy扩展包中loadtxt()函数读取data.txt文件，注意数据集中每行数据都是采用逗号进行分割。

```
# -*- coding: utf-8 -*-
import os
import numpy as np
data = np.loadtxt("wine.txt", dtype=float, delimiter=",")
print data
```

输出内容如下所示，可以发现已经将数据读取并存储至data变量中。

```
[[ 'B' '1' '1' '1' '1']
 [ 'R' '1' '1' '1' '2']
 [ 'R' '1' '1' '1' '3']
 ...,
 [ 'L' '5' '5' '5' '3']
 [ 'L' '5' '5' '5' '4']
 [ 'B' '5' '5' '5' '5']]
```

由于数据中存在“B”、“R”、“L”三类字母，故需要转换为字符类型（string）。同时该数据集存在一个特点，第一列为类标，后面四列为对应的数据集，则使用split()划分第一列和剩余4列数据，代码如下：

```
# -*- coding: utf-8 -*-
import os
import numpy as np
data = np.loadtxt("wine.txt",dtype=float,delimiter=",")
print data

yy, x = np.split(data, (1,), axis=1)
print yy.shape, x.shape
print x
print yy[:5]
```

输出如下所示：

```
(625L, 1L) (625L, 4L)
[[ '1' '1' '1' '1']
 [ '1' '1' '1' '2']
 [ '1' '1' '1' '3']
 ...,
 [ '5' '5' '5' '3']
 [ '5' '5' '5' '4']
 [ '5' '5' '5' '5']]
[[ 'B']
 [ 'R']
 [ 'R']
 [ 'R']
 [ 'R']]
```


同时这里的类标为“B”、“R”、“L”，我也将其转换为数字，其中“L”表示0，“B”表示1，“R”表示2。

代码如下：

```
#从字符型转换为Int整型
X = x.astype(int)
print X
#字母转换为数字
y = []
i = 0
print len(yy)
while i<len(yy):
    if yy[i]=="L":
        y.append(0)
    elif yy[i]=="B":
        y.append(1)
    elif yy[i]=="R":
        y.append(2)
    i = i + 1
print y[:5]
```

输出内容如下所示，现在可以直接使用X数组和y数组进行KNN算法分析。

```
[[1 1 1 1]
 [1 1 1 2]
 [1 1 1 3]
 ...,
 [5 5 5 3]
 [5 5 5 4]
 [5 5 5 5]]
625
[1, 2, 2, 2, 2]
```

四. KNN分析平衡秤数据集

接下来开始进行KNN算法分类分析，其中KNN核心算法主要步骤包括五步：

- 1.为了判断未知实例的类别，以所有已知类别的实例为参照
- 2.选择参数K
- 3.计算未知实例与所有已知实例的距离
- 4.选择最近K个已知实例
- 5.根据少数服从多数的投票法则，让未知实例归类为K个最近邻样本中最多数的类标

调用SKlearn机器学习扩展包的核心代码如下所示：

```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier()
print knn
```

输出算法原型如下：

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                      weights='uniform')
```

接下来调用fit()函数对数据集进行训练，再调用predict函数对数据集进行预测，完整代码如下所示：

```
# -*- coding: utf-8 -*-
import os
import numpy as np
data = np.loadtxt("data.txt", dtype=str, delimiter=",")
print data
print type(data)

yy, x = np.split(data, (1,), axis=1)
print yy.shape, x.shape
print x
print yy[:5]

#从字符型转换为Int整型
X = x.astype(int)
print X
#字母转换为数字
y = []
i = 0
print len(yy)
while i<len(yy):
    if yy[i]=="L":
        y.append(0)
    elif yy[i]=="B":
```

```

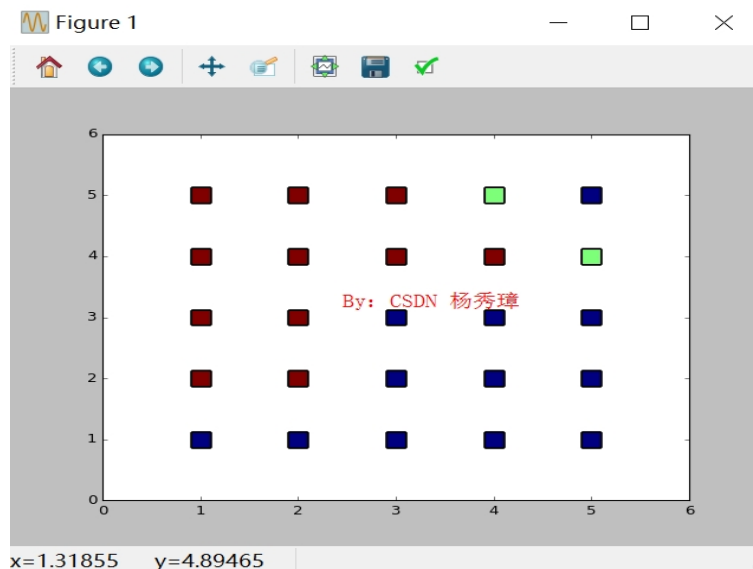
        y.append(1)
        elif yy[i]=="R":
            y.append(2)
        i = i + 1
    print y[:5]

#KNN分析
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier()
print knn
knn.fit(X,y)
pre = knn.predict(X)
print pre

#可视化分析
import matplotlib.pyplot as plt
L1 = [x[0] for x in X]
L2 = [x[2] for x in X]
plt.scatter(L1, L2, c=pre, marker='s',s=200)
plt.show()

```

输出预测结果如下所示，由于每列数据值为1到5，所以很多点出现重合。



最后简单评价KNN算法结果，代码如下：

```

#预测结果与真实结果比对

```

```

print sum(pre == y)    #输出准确率 召回率 F值
from sklearn import metrics
print(metrics.classification_report(y,pre))
print(metrics.confusion_matrix(y,pre))

```

输出如图所示图形，其中625组数据中，共预测正确540组，Precision值为83%，召回率Recall为86%，F1特征为84%，KNN算法总体分析结果较好。

```

540

```

	precision	recall	f1-score	support
0	0.85	0.97	0.90	288
1	0.00	0.00	0.00	49
2	0.95	0.91	0.93	288
avg / total	0.83	0.86	0.84	625

```

[[279  4  5]
 [ 41  0  8]
 [ 10 17 261]]

```

可以看到，该算法的优点为易于理解，简单容易实现，但是当数据量很大时，算法的效率不是很理想。

五. 代码优化

最后提供一段优化后的代码，提取其中的两列绘制相关的背景图。

```

# -*- coding: utf-8 -*-
import os
import numpy as np

#第一步 导入数据集
data = np.loadtxt("data.txt",dtype=str,delimiter=",")
print data
print type(data)

```

```

yy, x = np.split(data, (1,), axis=1) print yy.shape, x.shape
#从字符型转换为Int整型
X = x.astype(int)
#获取x两列数据,方便绘图 对应x、y轴
X = X[:, 1:3]
print X
#字母转换为数字
y = []
i = 0
print len(yy)
while i<len(yy):
    if yy[i]=="L":
        y.append(0)
    elif yy[i]=="B":
        y.append(1)
    elif yy[i]=="R":
        y.append(2)
    i = i + 1
print y[:5]

#第二步 KNN分析
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier()
print knn
knn.fit(X,y)
pre = knn.predict(X)
print pre

#第三步 数据评估
from sklearn import metrics
print sum(pre == y) #预测结果与真实结果比对
print(metrics.classification_report(y,pre)) #输出准确率 召回率 F值
print(metrics.confusion_matrix(y,pre))

#第四步 创建网格
x1_min, x1_max = X[:,0].min()-0.1, X[:,0].max()+0.1 #第一列
x2_min, x2_max = X[:,1].min()-0.1, X[:,1].max()+0.1 #第二列
xx, yy = np.meshgrid(np.arange(x1_min, x1_max, 0.1),
                     np.arange(x2_min, x2_max, 0.1)) #生成网格型数据
print xx.shape, yy.shape #(42L, 42L) (42L, 42L)
print xx.ravel().shape, yy.ravel().shape #(1764L,) (1764L,)
print np.c_[xx.ravel(), yy.ravel()].shape #合并 (1764L, 2L)
#ravel()拉直函数
z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
print z

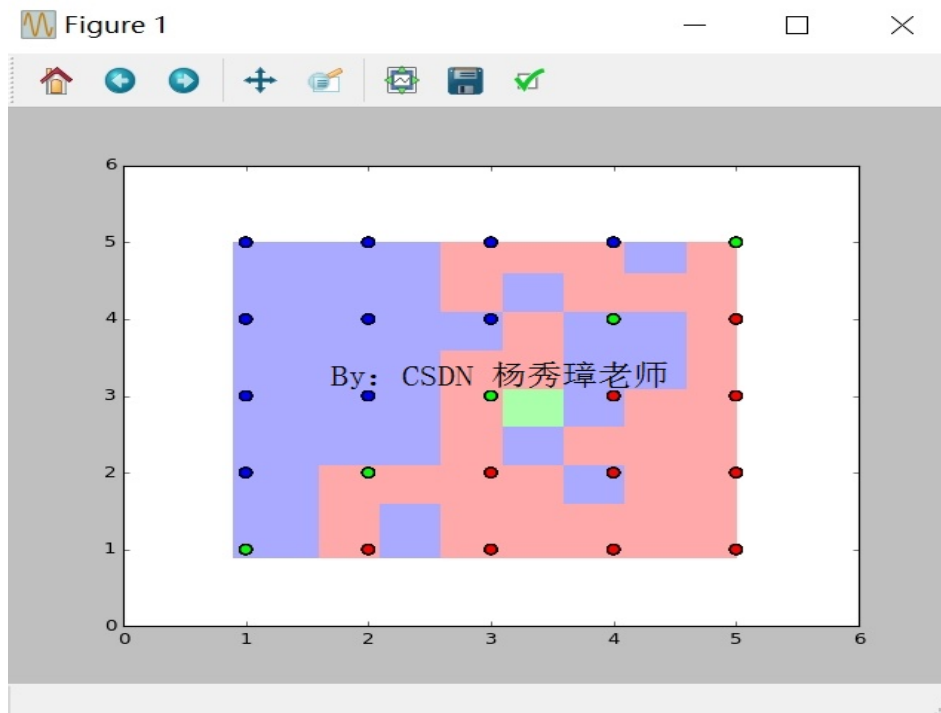
```

```

#第五步 绘图可视化
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF']) #颜色Map
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
plt.figure()
z = z.reshape(xx.shape)
plt.pcolormesh(xx, yy, z, cmap=cmap_light)
plt.scatter(X[:,0], X[:,1], c=y, cmap=cmap_bold, s=50)
plt.show()

```

输出如下所示，希望读者自行研究。



可以看到整个区域划分为三种颜色，绿色区域、红色区域和蓝色区域。同时包括散点图分布，对应数据的类标，包括绿色、蓝色和红色的点。可以发现，相同颜色的点主要集中于该颜色区域，部分蓝色点划分至红色区域或绿色点划分至蓝色区域，则表示预测结果与实际结果不一致。

感想杂谈：

遇一人白首，择一城终老。
经历风雨，慢慢变老。

去年的今天，你提着蛋糕赶着公交，来到花溪，鼓起勇气牵住了我的冰手；今年的这天，你在紧急出差遵义前，准备蛋糕，送我惊喜，倾世温柔。

很多人只看到了我的万字情书，制作的视频集锦，定期的狗粮，却不知道你背后的关爱与付出，还有我的亏欠。谢谢这一年你给我的温暖和安心，即使不在身边，几片文字，几段声音，摇首相望，也能感受到最纯真的爱情。况且还有这么多比韩剧还秀逗的剧情。来年我希望自己成长，学会担当，学会取舍，撑起一个新的家庭。

回首，以前的秀璋真的很少讲话，初中低头走路上学，高中打不出一个屁来，大四毕业也才发了第一条说说，为什么就改变呢？确实，光靠我的勇气，我可能还是那个闷不做声的小屁娃，但你的勇气和我的勇气加起来，我们足够应付整个世界，没有遇见你，我哪来的勇气，爱你就像爱生命。

PS：失眠之夜，生日快乐。致大家：一个人时，学会善待自己；两个人时，学会善待对方。

晚安，娜娜。晚安，贵阳。🤗

希望文章对你有所帮助，尤其是我的学生，如果文章中存在错误或不足之处，还请海涵。
(By:Eastmount 2017-12-08 深夜12点 <http://blog.csdn.net/eastmount/>)

👍 点赞 3 ☆ 收藏 🔄 分享 ...



Eastmount  博客专家

发布了444 篇原创文章 · 获赞 5908 · 访问量 484万+

他的留言板

关注