Hash

Lin Chang

- Hash
- Feature Hashing
- Hierarchical FeatureHashing

→Hash (散列)

Input (can be any length) ⟶ Output (fixed length)

An algorithm to map the data.

Hash (散列)

Application: fast lookup and encryption algorithm

Example

Mod

```
#! /usr/bin/env python
#coding=utf-8
#实现哈希表（线性地址再散列）
def ChangeKey(key,m,di):
    key01=(key+di) % m
    return key01
a=raw_input("Please entry the numbers:\n").split()
m=len(a)
print a
#print m  #长度为字符串的组数
dict01={}
for i in a:
    print i
    key=int(i)%m    #哈希表的构造必须用数来进行构造？
    if "%s"%key in dict01:
        NewKey=ChangeKey(key,m,1)
        while "%s"%NewKey in dict01:          #因为下面的 dict01 的 key 值是以字符
串来保存，因此这里作判断时也要用字符串格式
            NewKey=ChangeKey(NewKey,m,1)
        dict01["%s"%NewKey]=int(i)
    else:
        dict01["%s"%key]=int(i)
print dict01
```

```
Please entry the numbers:
23 32 32 54
['23', '32', '32', '54']
23
32
32
54
{'1': 32, '0': 32, '3': 23, '2': 54}
```

Conflict

```
Please entry the numbers:
1 4 8 6
['1', '4', '8', '6']
1
4
8
6
{'1': 1, '0': 4, '3': 6, '2': 8}
```

Chain

```
#实现哈希表（取余法构造的，拉链法解决冲突问题）

def chainHash(InputList):
    res={}
    for line in InputList:
        print line
        if line.split()[0] not in res:  #看键值是否已经存在
            temp=[]     #因为在拉链法中，键值包含多个对象，因此需要新建一个列表，把键值保存在这个列表中
            temp.append(line.split()[1])
            res["%s" % line.split()[0]]=temp
        else:
            res["%s" % line.split()[0]].append(line.split()[1])
return res
```

```
#主程序
a=raw_input("Please entry the numbers:\n").split()
m=len(a)
print a
#print m  #长度为字符串的组数
dict01=[]
for i in a:
    key=int(i)%m   #哈希表的构造必须用数来进行构造？
dict02=chainHash(dict01)
print dict02

#先构造一个这样的列表结构：  "key01 val01", "key02 val01", "key03 val01", "key01 val02", "key02
val02", "key01 val03", ...]
    key_str=str(key)
    i_str=str(i)
    combine_key_i=key_str+' '+i_str
    print combine_key_i
    dict01.append(combine_key_i)
print dict01
dict02={}
#将列表结构输入到拉链法中，解决冲突
```

Result

```
Please entry the numbers:
1 4 8 6
['1', '4', '8', '6']
1 1
0 4
0 8
2 6
['1 1', '0 4', '0 8', '2 6']
1 1
0 4
0 8
2 6
{'1': ['1'], '0': ['4', '8'], '2': ['6']}
```

Chain

```
Please entry the numbers:
1 4 8 6
['1', '4', '8', '6']
1 1
0 4
0 8
2 6
['1 1', '0 4', '0 8', '2 6']
1 1
0 4
0 8
2 6
{'1': ['1'], '0': ['4', '8'], '2': ['6']}
```

Mod

```
Please entry the numbers:
1 4 8 6
['1', '4', '8', '6']
1
4
8
6
{'1': 1, '0': 4, '3': 6, '2': 8}
```

→ Feature Hashing

High dimension to low dimension

Retain the expression ability of original features

→ Feature Hashing

High dimension to low dimension

Retain the expression ability of original features

Instead of building a **hash table** of the features encountered in training, instance of Feature Hasher apply a **hash function** to the features to determine their column index in sample matrice directly.

```
from sklearn.feature_extraction import FeatureHasher

f=open("ASLO-Features-SIFT.txt")
hashed=open("sift_hashing2.txt",'w') #创建一个文本文件，存储特征哈希后的结果
hash_list_dict=[] #进行哈希变换时要用到的，装有所有行字典的列表
num=0
for line in f:   #循环读取文件中的每一行
    #print line
    a=line.split()
    #print a
    hash_dict={} #每一行值构建一个字典
    feature_number=1  #键，这里是用列指数代表每行每一个特征值值对应的特征名称
    for j in a: #将每一行中的数值单个取出
        value=float(j)  #键值需要是数字
        feature_number_str=str(feature_number)
        hash_dict[feature_number_str]=value  #将键-值存入字典中
        feature_number+=1
    #print hash_dict
    hash_list_dict.append(hash_dict)

#print hash_list_dict
```

```
#进行特征哈希
h=FeatureHasher(n_features=60)
feature_hashed=h.transform(hash_list_dict)
print feature_hashed.toarray()
#写入数据
for m in feature_hashed.toarray():
    #将数组中的'['和']'去掉，方便后续计算
    a=str(m)[1:-1]
    b=a.split()
    for p in b:
        hashed.write(str(p))
        hashed.write('\t')
    hashed.write('\n')


f.close()
hashed.close()
```

Thank you