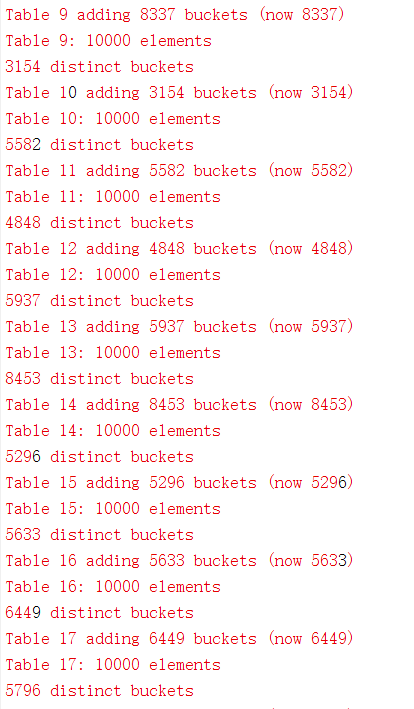
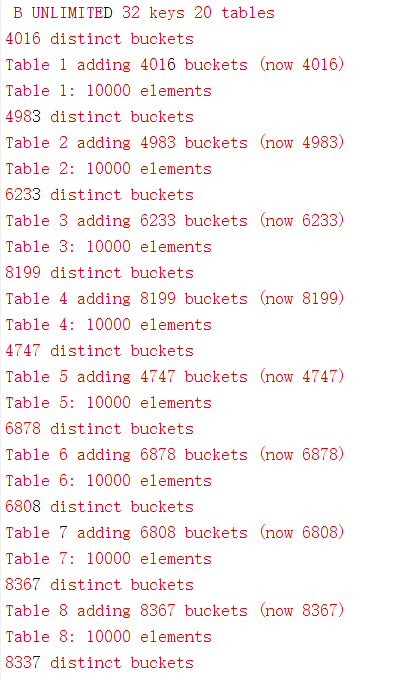
用cifar-10的数据运行，用32bit的k建立20个哈希表，用了traindata的前10000个数据。用了lsh方法。

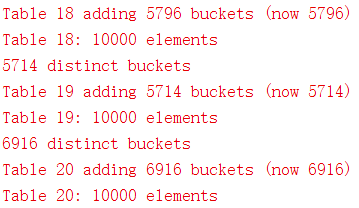
>>load('cifar\_10\_gist')

>>traindata=traindata';

>>T=lsh('lsh',20,32,size(traindata(:,1:10000),1),traindata(:,1:10000),'range',0.1);%type,l,k,d,x,varargin

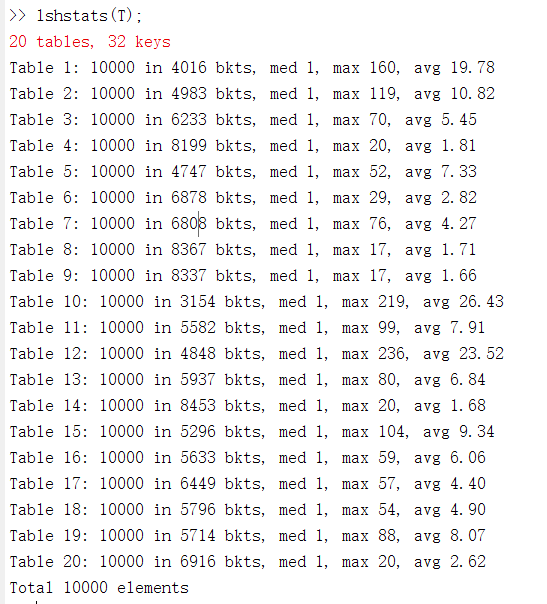
结果表示10000个数据全被索引到每个哈希表中，数据在哈希桶中的分布比较合理。





>> lshstats(T);

哈希桶里的数据都比较均匀，没有特别大的桶。



>> lshstats(T,'test',traindata,testdata',2);

用测试集测试一下，搜索2个最近邻。

平均查找了138.02个数据，大约1.3%，只有一个错误的。



>> lshstats(T(1:5),'test',traindata(:,1:10000),testdata',2);

如果只用前5个哈希表查找测试集2个最近邻，平均值用查找41.5个数据，错误率变高了。



>> tic; [nnlsh,numcand]=lshlookup(testdata(:,1),traindata,T,'k',11,'distfun','lpnorm','distargs',{1});toc

搜索一条数据的10个最近邻（也不是很明白在干啥）



下面用‘e2lsh’方法，也是20个哈希表，k=32，前10000个数据训练。然后用测试集测试。

>> Te=lsh('e2lsh',20,32,size(traindata(:,1:10000),1),traindata(:,1:10000),'range',0.1,'w',-4);

>> lshstats(Te,'test',traindata(:,1:10000),testdata,2);

查找率变低了，错误率增高。



关于ap值，lsh是无监督的，可以用测试的结果算出来一个查找率和错误率。找到了lsh的文章，他是这么定义lsh的误差的。

