INTRO. TO WEB SCIENCE: CS 532: A10

Due on Monday, May 1, 2017

Dr. Nelson

Udochukwu Nweke

Contents

Problem 1

Listing 1: Computing K Nearest Neighbours Code

```
from numpredict import knnestimate, cosineDistance

def readfile(filename):
    return do_readfile(open(filename).readlines())

def do_readfile(lines):
    colnames = lines[0].strip().split('\t')[1:]
    rownames = []
    data = []

    for line in lines[1:]:
        p = line.strip().split('\t')
        rownames.append(p[0])
        data.append([float(x) for x in p[1:]])

    return rownames, colnames, data

def createVectorData(blognames, data):

    if( len(blognames) != len(data) ):
        print('mismatch exiting')
        return

    vectors = []
    for i in range(0, len(blognames)):

        blogDict = {}
        blogDict['name'] = blognames[i]
        blogDict['input'] = data[i]
        blogDict['result'] = 0
        vectors.append(blogDict)

    return vectors

def getIntFromList(lst):

    lst = lst.split('\t')
    for i in range(0, len(lst)):
        lst[i] = int(lst[i])

    return lst


webSciDLVector = '8 39   1    286  2    175  194  65   0'
webSciDLVector = getIntFromList(webSciDLVector)
```

```
fMeasureVector = '1 4    4    1    3    23   1    9    5'
fMeasureVector = getIntFromList(fMeasureVector)

blognames, words, data = readfile('./blogMatrix.txt')
vectors = createVectorData(blognames, data)

k = 5
knnestimate(vectors, fMeasureVector, k+1)
```

Listing 2: Pci Code

```
from random import random,randint
import math
#from pylab import *

def wineprice(rating,age):
  peak_age=rating-50

  # Calculate price based on rating
  price=rating/2
  if age>peak_age:
    # Past its peak, goes bad in 10 years
    price=price*(5-(age-peak_age)/2)
  else:
    # Increases to 5x original value as it
    # approaches its peak
    price=price*(5*((age+1)/peak_age))
  if price<0: price=0
  return price


def wineset1():
  rows=[]
  for i in range(300):
    # Create a random age and rating
    rating=random()*50+50
    age=random()*50

    # Get reference price
    price=wineprice(rating,age)

    # Add some noise
    price*=(random()*0.2+0.9)

    # Add to the dataset
    rows.append({'input':(rating,age),
                 'result':price})
  return rows

#cosine distance code - begin
def vecDotProduct(vecA, vecB):

  product = 0
```

```
  for i in range(0, len(vecA)):
    product += vecA[i] * vecB[i]

  return product

def vecMagnitude(vec):
  summation = 0
  for i in range(0, len(vec)):
    summation += vec[i] * vec[i]

  return math.sqrt(summation)

def cosineSimilarity(vecA, vecB):

  denom = vecMagnitude(vecA) * vecMagnitude(vecB)

  if( denom == 0 ):
    return 0
  else:
    return vecDotProduct(vecA, vecB)/denom

def cosineDistance(vecA, vecB):
  return 1 - cosineSimilarity(vecA, vecB)

#cosine distance code - end

def euclidean(v1,v2):
  d=0.0
  for i in range(len(v1)):
    d+=(v1[i]-v2[i])**2
  return math.sqrt(d)


def getdistances(data,vec1):
  distancelist=[]

  # Loop over every item in the dataset
  for i in range(len(data)):
    vec2=data[i]['input']

    # Add the distance and the index
    #distancelist.append((euclidean(vec1,vec2),i))
    distancelist.append((cosineDistance(vec1,vec2),i))

  # Sort by distance
  distancelist.sort()
  return distancelist

def knnestimate(data,vec1,k=5):
  # Get sorted distances
  dlist=getdistances(data,vec1)
  avg=0.0

```

```
  # Take the average of the top k results
  for i in range(k):
    idx=dlist[i][1]
    avg+=data[idx]['result']
    print('\ti:', i, data[idx]['name'])
  avg=avg/k
  return avg

def inverseweight(dist,num=1.0,const=0.1):
  return num/(dist+const)

def subtractweight(dist,const=1.0):
  if dist>const:
    return 0
  else:
    return const-dist

def gaussian(dist,sigma=5.0):
  return math.e**(-dist**2/(2*sigma**2))

def weightedknn(data,vec1,k=5,weightf=gaussian):
  # Get distances
  dlist=getdistances(data,vec1)
  avg=0.0
  totalweight=0.0

  # Get weighted average
  for i in range(k):
    dist=dlist[i][0]
    idx=dlist[i][1]
    weight=weightf(dist)
    avg+=weight*data[idx]['result']
    totalweight+=weight
  if totalweight==0: return 0
  avg=avg/totalweight
  return avg

def dividedata(data,test=0.05):
  trainset=[]
  testset=[]
  for row in data:
    if random()<test:
      testset.append(row)
    else:
      trainset.append(row)
  return trainset,testset

def testalgorithm(algf,trainset,testset):
  error=0.0
  for row in testset:
    guess=algf(trainset,row['input'])
    error+=(row['result']-guess)**2
    #print row['result'],guess
```

```
  #print error/len(testset)
  return error/len(testset)

def crossvalidate(algf,data,trials=100,test=0.1):
  error=0.0
  for i in range(trials):
    trainset,testset=dividedata(data,test)
    error+=testalgorithm(algf,trainset,testset)
  return error/trials

def wineset2():
  rows=[]
  for i in range(300):
    rating=random()*50+50
    age=random()*50
    aisle=float(randint(1,20))
    bottlesize=[375.0,750.0,1500.0][randint(0,2)]
    price=wineprice(rating,age)
    price*=(bottlesize/750)
    price*=(random()*0.2+0.9)
    rows.append({'input':(rating,age,aisle,bottlesize),
                 'result':price})
  return rows

def rescale(data,scale):
  scaleddata=[]
  for row in data:
    scaled=[scale[i]*row['input'][i] for i in range(len(scale))]
    scaleddata.append({'input':scaled,'result':row['result']})
  return scaleddata

def createcostfunction(algf,data):
  def costf(scale):
    sdata=rescale(data,scale)
    return crossvalidate(algf,sdata,trials=20)
  return costf

weightdomain=[(0,10)]*4

def wineset3():
  rows=wineset1()
  for row in rows:
    if random()<0.5:
      # Wine was bought at a discount store
      row['result']*=0.6
  return rows

def probguess(data,vec1,low,high,k=5,weightf=gaussian):
  dlist=getdistances(data,vec1)
  nweight=0.0
  tweight=0.0

  for i in range(k):
```

```
    dist=dlist[i][0]
    idx=dlist[i][1]
    weight=weightf(dist)
    v=data[idx]['result']

    # Is this point in the range?
    if v>=low and v<=high:
      nweight+=weight
    tweight+=weight
  if tweight==0: return 0

  # The probability is the weights in the range
  # divided by all the weights
  return nweight/tweight

def cumulativegraph(data,vec1,high,k=5,weightf=gaussian):
  t1=arange(0.0,high,0.1)
  cprob=array([probguess(data,vec1,0,v,k,weightf) for v in t1])
  plot(t1,cprob)
  show()


def probabilitygraph(data,vec1,high,k=5,weightf=gaussian,ss=5.0):
  # Make a range for the prices
  t1=arange(0.0,high,0.1)

  # Get the probabilities for the entire range
  probs=[probguess(data,vec1,v,v+0.1,k,weightf) for v in t1]

  # Smooth them by adding the gaussian of the nearby probabilites
  smoothed=[]
  for i in range(len(probs)):
    sv=0.0
    for j in range(0,len(probs)):
      dist=abs(i-j)*0.1
      weight=gaussian(dist,sigma=ss)
      sv+=weight*probs[j]
    smoothed.append(sv)
  smoothed=array(smoothed)

  plot(t1,smoothed)
  show()
```

Support your answer: include all relevant discussion, assumptions, examples, etc.

Using the data from A8:

Consider each row in the blog-term matrix as a 1000 dimension vector, corresponding to a blog.

- From chapter 8, replace numpredict.euclidean() with cosine as the distance metric. In other words, you'll be computing the cosine between vectors of 1000 dimensions.

- Use knnestimate() to compute the nearest neighbors for both:

`http://f-measure.blogspot.com/`
`http://ws-dl.blogspot.com/`
for k=1,2,5,10,20.

Solution 1:

1. I extracted vectors of `http://f-measure.blogspot.com/` and `http://ws-dl.blogspot.com/` from blog-term matrix and considered each row as a 1000 dimension vector. *blogMatrix* was created in Assignment 8. This was achieved by using *do_readfile()* and *createVectorData()* in listing 1.

2. In order to computer cosine similarity between vectors of 1000 dimension (in blogMatix.txt) and `http://f-measure.blogspot.com/` and `http://ws-dl.blogspot.com/` blog, I modified Pci *numpredic.py* code by adding line 40-65 in listing 2 which replaced euclidean distance function with cosine similarity function.

3. In order to compute K nearest neigbbors for `http://f-measure.blogspot.com/` and `http://ws-dl.blogspot.com/`, I used *knnestimate()* in listing 1 to computer for k=1,2,5,10,20 nearest neighbors respectively. The result for `http://f-measure.blogspot.com/` K Neighbors is seen from Table 1 to Table 5. The result for `http://ws-dl.blogspot.com/` K Neighbors is seen from Table 6 to Table 10.

Table 1: K= 1 Nearest Neighbors f-measure

K=1	Nearest Neighbors
1	the fast break of champions

Table 2: K= 2 Nearest Neighbors f-measure

K=2	Nearest Neighbors
1	the fast break of champions
2	The Jeopardy of Contentment

Table 3: K= 5 Nearest Neighbors f-measure

K=5	Nearest Neighbors
1	the fast break of champions
2	The Jeopardy of Contentment
3	The Girl at the Rock Show
4	I/LOVE/TOTAL/DESTRUCTION
5	Cherry Area

Table 4: K= 10 Nearest Neighbors f-measure

K=10	Nearest Neighbors
1	the fast break of champions
2	The Jeopardy of Contentment
3	The Girl at the Rock Show
4	I/LOVE/TOTAL/DESTRUCTION
5	Cherry Area
6	SunStock Music
7	In the Frame Film Reviews
8	CardrossManiac2
9	Encore
10	The Stark Online

Table 5: K= 20 Nearest Neighbors f-measure

K=20	Nearest Neighbors
1	the fast break of champions
2	The Jeopardy of Contentment
3	The Girl at the Rock Show
4	I/LOVE/TOTAL/DESTRUCTION
5	Cherry Area
6	SunStock Music
7	In the Frame Film Reviews
8	CardrossManiac2
9	Encore
10	The Stark Online
11	Steel City Rust
12	Diagnosis: No Radio
13	www.doginasweater.com Live Show Review Archive
14	Some Call It Noise....
15	15 Cuz Music Rocks
16	She May Be Naked
17	GLI Press
18	Morgan's Blog
19	the fast break of champions
20	Pithy Title Here

Table 6: K= 1 Nearest Neighbor WSDL

K=1	Nearest Neighbors
1	tmacthemost

Table 7: K= 2 Nearest Neighbors WSDL

K=2	Nearest Neighbors
1	tmacthemost
2	the traveling neighborhood

Table 8: K= 5 Nearest Neighbors WSDL

K=5	Nearest Neighbors
1	tmacthemost
2	the traveling neighborhood
3	MarkFisher's-MusicReview
4	Bonjour Girl
5	Avidd Wallows' Blog

Table 9: K= 10 Nearest Neighbors WSDL

K=10	Nearest Neighbors
1	tmacthemost
2	the traveling neighborhood
3	MarkFisher's-MusicReview
4	Bonjour Girl
5	Avidd Wallows' Blog
6	STATUS
7	Cherry Area
8	A2 MEDIA COURSEWORK JOINT BLOG
9	The Stark Online
10	Pithy Title Here

Table 10: K= 20 Nearest Neighbors WSDL

K=20	Nearest Neighbors
1	tmacthemost
2	the traveling neighborhood
3	MarkFisher's-MusicReview
4	Bonjour Girl
5	Avidd Wallows' Blog
6	STATUS
7	Cherry Area
8	A2 MEDIA COURSEWORK JOINT BLOG
9	The Stark Online
10	Pithy Title Here
11	Floorshime Zipper Boots
12	juanbook
13	Punk Rock Teaching
14	Chantelle Swain A2 Media Studies
15	Myopiamuse
16	She May Be Naked
17	Kid F
18	the fast break of champions
19	tDiagnosis: No Radio
20	20 Mile In Mine

References

[1] Blog Time Now. http://blogtimenow.com/blogging/find-blogger-blog-id-post-id-unique-id-number/. Accessed: 2017-10-04.

[2] Toby Segaran. Programming Collective Intelligence, 2007.