Web Science: Assignment #9

Alexander Nwala

Apurva Modi

Wednesday, May 1, 2019

Contents

Problem 1

Using the data from A7:

1. Consider each row in the blog-term matrix as a 1000 dimension vector, corresponding to a blog.
2. 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.

   Use cosine distance metric (chapter 8) not euclidean distance. So you have to implement numpredict.cosine() instead of using numpredict.euclidean() in: https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py

SOLUTION :

1. I have used the "blogdata.txt" blog metric from A7 and code files from **Programming Collective Intelligence** text book.
2. Modified the "numpredict" file to add the cosine similarity function and created a new file "AssignmentNumPredict.py"
3. Created another script ,"kNNClusturing.py" to determine nearest neighbours using knnestimate() function for both the blogs.
4. Then, I Passed the title of the given two blogs, whose nearest neighbours need to be determined

Listing 1: AssignmentNumPredict.py

```
from random import random,randint
import math
from scipy import spatial

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

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

def cosine(v1,v2):

  result = 1 - spatial.distance.cosine(v1, v2)
  return result




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

  #import pdb
  #pdb.set_trace()

  # Loop over every item in the dataset
  for i, cal in enumerate(data):
    vec2 = cal

    # Add the distance and the index

    distancelist.append((cosine(vec1,vec2),i))


  # Sort by distance
  distancelist.sort(reverse=True)
  return distancelist

def knnestimate(data,vec1,k=5):
  # Get sorted distances
```

```
  distant=getdistances(data,vec1)


  return distant

# # Take the average of the top k results
# for i in range(k):
  # idx=dlist[i][1]
  # avg+=data[idx]['result']
# 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):
```

Problem 1 continued on next page...

```
  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):
```

Problem 1 continued on next page...

```
  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

from pylab import *

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()
```

Below code determines the nearest distance through cosine similarity using the modified numpredict.py:

Listing 2: kNNClusturing.py

```
from AssignmentNumPredict import *


def calculateData():

    fmeasure = 'F-Measure'
    wlblog = 'Web Science and Digital Libraries Research Group'
    bnames = {}
    mesf = []
    cesf = []
    with open("blogdata.txt", 'r', encoding='utf-8') as f:
        doctext = f.readlines()
        for i, line in enumerate(doctext):
            if i == 0:
                # skip header
                continue
            tuples = line.strip().split('\t')
            if tuples[0] == fmeasure:
                for i in range(1, len(tuples)):
                    mesf.append(float(tuples[i]))
            elif tuples[0] == wlblog:
                for i in range(1, len(tuples)):
                    cesf.append(float(tuples[i]))
            else:
                bnames[tuples[0]] = []
                for i in range(1, len(tuples)):
                    bnames[tuples[0]].append(float(tuples[i]))

    return bnames, mesf, cesf


def knnest(calval, mesvec, gpvec):
    nn = knnestimate(bnames.values(), mesvec)
    print("====================================" * 2)
    print("F-Measure")
    print("====================================" * 2)
    kvals = [1, 2, 5, 10, 20]
    for k in kvals:
        print('k =', k)
        for j in range(k):
            print('%s\t%.6f' % (list(bnames.keys())[nn[j][1]], nn[j][0]))

        print("------------------------------------" * 2)
    print()

    print("====================================" * 2)
    print("Web Science and Digital Libraries Research Group")
    print("====================================" * 2)
    nn = knnestimate(bnames.values(), gpvec)
    for k in kvals:
        print('k =', k)
        for j in range(k):
            print('%s\t%.6f' % (list(bnames.keys())[nn[j][1]], nn[j][0]))
```

Problem 1 continued on next page...

```
55          print("-------------------------------------" * 2)




60

if __name__ == "__main__":
    bnames, dvec, worvec = calculateData()
    knnest(bnames.values(), dvec, worvec)
```

Problem 1 continued on next page. . .

Listing 3: KNN Output

```
 ================================================================================
F-Measure
================================================================================
k = 1
Catering Harian Tangerang     nan
--------------------------------------------------------------------------------
k = 2
Catering Harian Tangerang     nan
Pursuing The Art of Life 0.707630
--------------------------------------------------------------------------------
k = 5
Catering Harian Tangerang     nan
Pursuing The Art of Life 0.707630
THE ORIFICE!!  0.700501
Start talking about these things.  0.696794
Witty Title Pending 0.683905
--------------------------------------------------------------------------------
k = 10
Catering Harian Tangerang     nan
Pursuing The Art of Life 0.707630
THE ORIFICE!!  0.700501
Start talking about these things.  0.696794
Witty Title Pending 0.683905
Life at the lake    0.672151
Jackson449     0.659178
Glitter Every Day   0.649249
Stony Bridge Farm   0.644484
Liz B. Quilting     0.632190
--------------------------------------------------------------------------------
k = 20
Catering Harian Tangerang     nan
Pursuing The Art of Life 0.707630
THE ORIFICE!!  0.700501
Start talking about these things.  0.696794
Witty Title Pending 0.683905
Life at the lake    0.672151
Jackson449     0.659178
Glitter Every Day   0.649249
Stony Bridge Farm   0.644484
Liz B. Quilting     0.632190
My Tenuous Grasp    0.629786
Random Thoughts of a Plastic Surgeon    0.623991
Cultural Media Literacy  0.623974
The Accidental CrossFitter    0.603577
Short Presents | Food. Fashion. Fun.    0.590743
Evil Ditties   0.587276
A Teacher's View    0.579137
Diary of a Faithful Mama 0.574603
Life, Faith, and Urban Farming     0.573619
Two Lovebird Locavores   0.559064
--------------------------------------------------------------------------------
```

Problem 1 continued on next page...

```
================================================================================
Web Science and Digital Libraries Research Group
================================================================================
k = 1
Catering Harian Tangerang     nan
--------------------------------------------------------------------------------
k = 2
Catering Harian Tangerang     nan
Cultural Media Literacy  0.629830
--------------------------------------------------------------------------------
k = 5
Catering Harian Tangerang     nan
Cultural Media Literacy  0.629830
Liz B. Quilting     0.556286
Random Thoughts of a Plastic Surgeon    0.548003
Life at the lake    0.538732
--------------------------------------------------------------------------------
k = 10
Catering Harian Tangerang     nan
Cultural Media Literacy  0.629830
Liz B. Quilting     0.556286
Random Thoughts of a Plastic Surgeon    0.548003
Life at the lake    0.538732
Indigeny & Energetics    0.533557
GardenSpotlight     0.532078
A Teacher's View    0.531536
Stony Bridge Farm   0.525594
Witty Title Pending 0.518742
--------------------------------------------------------------------------------
k = 20
Catering Harian Tangerang     nan
Cultural Media Literacy  0.629830
Liz B. Quilting     0.556286
Random Thoughts of a Plastic Surgeon    0.548003
Life at the lake    0.538732
Indigeny & Energetics    0.533557
GardenSpotlight     0.532078
A Teacher's View    0.531536
Stony Bridge Farm   0.525594
Witty Title Pending 0.518742
THE ORIFICE!!  0.510066
Pursuing The Art of Life 0.505948
Minutes with Coach  0.496287
Life, Faith, and Urban Farming     0.488257
Perfect...Not So Much    0.486288
Main Tourist Trips  0.485894
Jackson449     0.466494
Diary of a Faithful Mama 0.465566
Green on the Scene Wellness & Fertility 0.464534
Start talking about these things.  0.460376
--------------------------------------------------------------------------------
```

References

1. https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py
2. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html
3. https://cmry.github.io/notes/euclidean-v-cosine