# jsloan3_Homework1

February 7, 2020

# 1 COSC 5010 - Data Mining

## 1.1 Joshua Sloan

## 1.2 Homework 1

### 1.2.1 Question 1)

**2)**

**a.** Discrete, quantitative, ratio

(While I do see a binary element with the AM and PM times and time is generally on a continuous scale, I am interpretting this as more than just either AM or PM, but also with an associated time. For this reason, I am referring to this as discrete given that times are normally only given with respect to the minute (i.e. 7:45 PM) such that there are 1440 (or 60*24) different possible combinations between both AM and PM times. Additionally, from our lecture notes we can see that time is a ratio.

**b.** Continuous, quantitative, ratio

**c.** Discrete, qualitative, ordinal

**d.** Continuous, quantitative, ratio

**e.** Discrete, qualitative, ordinal

**f.** Continuous, quantitative, ratio

(Ratio because sea level can be defined as an origin, such that you can make statements like "I am 2x times higher above sea level than you")

**g.** Discrete, quantitative, ratio

**h.** Discrete, qualitative, nominal

(While there is some form of ordering to the ISBN numbering system, you cannot definitively say that the ordering is in relation to any quality of the book which can be regarded as "greater than or less than" other values and therefore, is nominal)

**i.** Discrete, qualitative, ordinal

**j.** Discrete, qualitative, ordinal

**k.** Continuous, quantitative, ratio

(Ratio because you can measure distances from the origin, such that you can make statements like "I am 2x closer to the center of campus than you")

**l.** Continuous, quantitative, ratio

**m.** Discrete, qualitative, nominal

(Discrete under the assumption that there is a limit to the number of invited guests to the party, and consequently the number of possible coat check numbers that you can receive)

**5)**

One example where identification number can be useful for prediction might be for a grocery store predicting when they are going to need to restock on a certain product. In this manner, they are able to use the corresponding identification number of a given item to determine how often is is being purchased from transaction data in conjunction with a stock counter associated with the item. From this, the store could predict how quickly the stock will reduce to 0 and ensure that they give themselves ample time to order more of that product such that it will arrive before they are out of stock accordingly.

**13)**

**a.**

For starters, this algorithm states to sort the distances for each object in DECREASING order (such that the first instance of the list will be the largest distance and the last instance of the list will be the shortest distance (i.e. nearest neighbor), yet then for the sorted nearest neighbors list, the algorithm states to take the FIRST k elements of the list for every object (which would be giving the k-furthest neighbors not k-nearest).

Beyond this however, there is a larger problem if there are duplicates in the list. Assuming that this algorithm does in fact return the k-NEAREST neighbors, then if there are duplicate values, that value will have the shortest distance (i.e. 0 distance). Any data point with a duplicate value will automatically have a nearest neighbor that does not provide any true insight and any value with a k-nearest neighbor that is a duplicate value will have multiple duplicates of the same data point within the list of its k-nearest neighbors (provided k is big enough to accomodate all duplicate points). With enough duplicate values, then all of the k values may be entirely comprised of the same data point for all points.

**b.**

To fix this, I would simply preprocess the data such that duplicate points are removed from the list prior to running the algorithm. In this case, all nearest neighbors will have a distance $> 0$ and no nearest neighbor for any object will be a duplicate of itself. *NOTE: I am relying on the assumption that at least one attribute value of should be unique for what that object represents. In other words, there is some form of identifying feature of each object which maps only to one instance (such as an ID number or name). In this regard, two objects with the same ID number would belong the the same object, letting us know that this object is a duplicate, rather than 2 separate objects that just so happen to contain the same values (such as an anonymous response to a survey). This could lead to complications, should this uniquely identifying feature be an email (for which there can be many mapping to a single person), but I am also assuming that this knowledge would either be accounted for, or would not create duplicate objects for both values associated with a uniquely identifiable person.*

**15)**

Sampling technique A takes into account the number of objects within each group and samples a preportional number of objects from each group, whereas sampling technique B simply randomly samples objects from accross all groups.

In this manner, Sampling technique A is guarenteed to have objects from all groups present (provided a non-trivial value for n), where technique B can only guarentee membership from all groups if n=m (with our goal being to obtain a sample such that n<m). From this, sampling technique A is a better representation of objects from all groups, where technique B can be skewed by an overabundance of objects from a particular group (especially when considered in conjunction with the assumption of sampling with replacement). In this regard, groups with a low number of objects are less likely to be selected than objects from larger groups when using technique B.

**16)**

**a.**

Given this transformation, we are simply weighting the frequency of the word by the number of documents that the word appears in. If a term only appears in one document, then it is given a weight of log(m/1) = log(m) (which is the maximum weight that can be attributed to it). If a word appears in every document, then the weight becomes log(m/m) = log(1) = 0.

**b.**

This might be useful for determining what the content of a document is. When using this transformation, common words (such as 'a', 'and', 'the', etc.) which are likely to appear in most (if not all) documents are given a very low (if not 0) weight, whereas unique words which are exclusive to a particular document are given the maximum weight. From this, the unique words from each document (which are the ones most likely to give insight to document specific content/ideas) will be weighted higher after this transformation, helping us determine what the contents of a particular document is based off of a word count.

### 1.2.2 Question 2)

When is sampling with replacement appropriate and when is sampling without replacement more preferable? Provide two examples where each is more appropriate.

Sampling with replacement is appropriate when we are trying to sample from a population where we know the data is independent (i.e. the probability of selection does not depend on any prior selections) and repetition is allowed. For example, if we are trying to sample to emulate a dice toss, then we should randomly sample from the population [1,2,3,4,5,6] with replacement, such that every sample value is equally likely (1/6th) regardless of any previous samples, just as in a real world dice toss.

Sampling without replacement is appropriate when we are sampling with the knowledge that we cannot have repetition and as the number of data points in our smaple space decreases, the probability of each indiviudal instance being selected increases. One example might be if we are randomly select 3 winners for a contest from a population (with the knowledge that each individual can only win once). In this case, everytime an individual is sampled from the pool, they must be removed from the population space such that 2 new unique winners can be selected (now with an increased probability of selection).

Samples obtained uniformly at random can miss anomalies (underrepresented data points) in datasets. How can we sample more systematically using PCA?

You can use PCA to get rid of samples with low variance. By doing this, it is likely that we will remove dimensions which are less likely to have anomalies (given the low variance) and therefore when sampling from the remaining dimensions, we are more likely to sample among dimensions which will provide anomalous data points. This point can be seen clearly through the idea of Gibbs sampling, such that fixing all other features except one where values largely take on similar values does not explore the sample space as well as sampling from a feature where we are more likely to sample a value containing a much different value then the previous sample (respective to that feature).

### 1.2.3 Question 3)

Due to curse of dimensionality distances become meaningless in high-dimensional spaces. That is, the minimum distance between random pairs of nodes becomes really close to the maximum distance between such pairs. In this problem, you are going to verify this phenomenon. Write a program in Java, C, C++, Python, or MATLAB.

[1]:
```
# importing libraries utilized
import numpy as np
import scipy
from sklearn import decomposition
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numba
from numba import jit, cuda, prange
import math

# creating a test matrix for gamma computation
n = 100
d = 10
temp = np.random.randint(low=0, high=100, size=(n,d)).astype(float)
```

[2]:
```
# L2 norm (Euclidean) distance function
@jit(nopython=True,parallel = True,fastmath=True)
def minAndMaxDistance(matrix):
    currentMin = 1000000
    currentMax = 0
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if(i == j):
                pass
            else:
                calcVal = np.linalg.norm(matrix[i]-matrix[j])
                #calcVal = scipy.spatial.distance.
↪cdist(matrix[i],matrix[j],'euclidean')
                #print(calcVal)
```

```
                if(currentMin > calcVal):
                    currentMin = calcVal
                if(currentMax < calcVal):
                    currentMax = calcVal
    return currentMin, currentMax
```

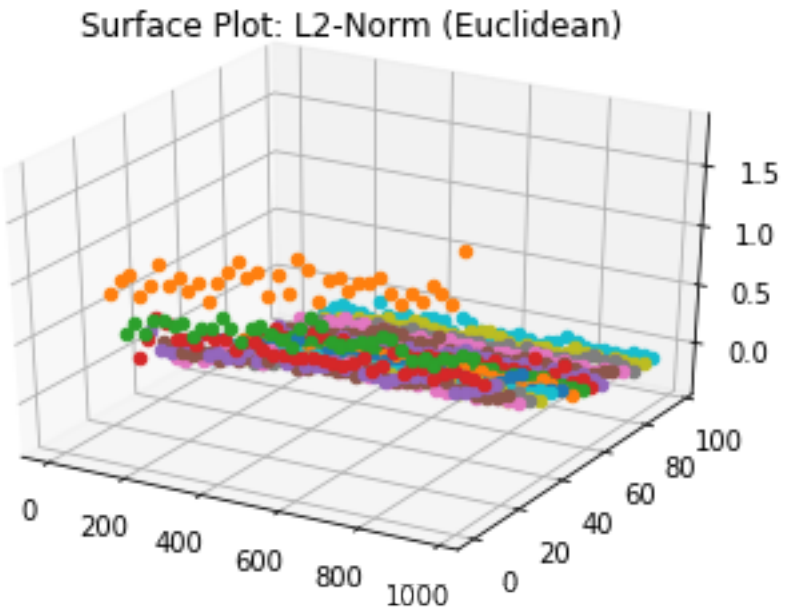
[3]:
```
# L1 norm distance function
@jit(nopython=True,parallel = True,fastmath=True)
def minAndMaxDistanceL1(matrix):
    currentMin = 1000000
    currentMax = 0
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if(i == j):
                pass
            else:
                calcVal = np.linalg.norm(matrix[i]-matrix[j],1)
                #calcVal = scipy.spatial.distance.
 ↪cdist(matrix[i],matrix[j],'cityblock')
                #print(calcVal)
                if(currentMin > calcVal):
                    currentMin = calcVal
                if(currentMax < calcVal):
                    currentMax = calcVal
    return currentMin, currentMax
```

[4]:
```
# un-used function
@jit(nopython=True,parallel = True)
def gammaCalc(n,d):
    tMat = np.random.randint(low=0, high=100, size=(i,j))
    tMin, tMax = minAndMaxDistance(tMat)
    if(tMin != 0.0):
        tGamma = math.log10((tMax - tMin)/tMin)
        return tGamma
    else:
        return math.inf
```

[5]:
```
# test
minimum, maximum = minAndMaxDistance(temp)
print("_____________")
print("Max:",maximum)
print("Min:",minimum)

gamma = math.log10((maximum - minimum)/minimum)
print("Gamma:",gamma)
```

```
---------------
Max: 222.24760966093652
Min: 41.09744517606904
Gamma: 0.644223907925862
```

[6]:
```python
# Euclidean distance gamma plot
ax = plt.axes(projection='3d')

# needed for surface plot
X = np.arange(100,1000,25)
Y = np.arange(1,100,5)
Z = np.empty((len(X),len(Y)))

#indRow = 0
for i in X:
    #indCol = 0
    for j in Y:
        #gammaCalc(i,j)
        tMat = np.random.randint(low=0, high=100, size=(i,j)).astype(float)
        tMin, tMax = minAndMaxDistance(tMat)
        if(tMin != 0.0):
            tGamma = math.log10((tMax - tMin)/tMin)
            #print(tGamma)
            ax.scatter3D(i,j,tGamma)
            #Z[indRow,indCol] = tGamma
        else:
            ax.scatter3D(i,j,math.inf)
            #Z[indRow,indCol] = math.inf
        #indCol = indCol + 1
    #indRow = indRow + 1
#print(Z)
#X, Y = np.meshgrid(X, Y)
#ax.plot_surface(X, Y, Z,cmap='viridis', edgecolor='none')
ax.set_title('Surface Plot: L2-Norm (Euclidean)')
plt.show()
```

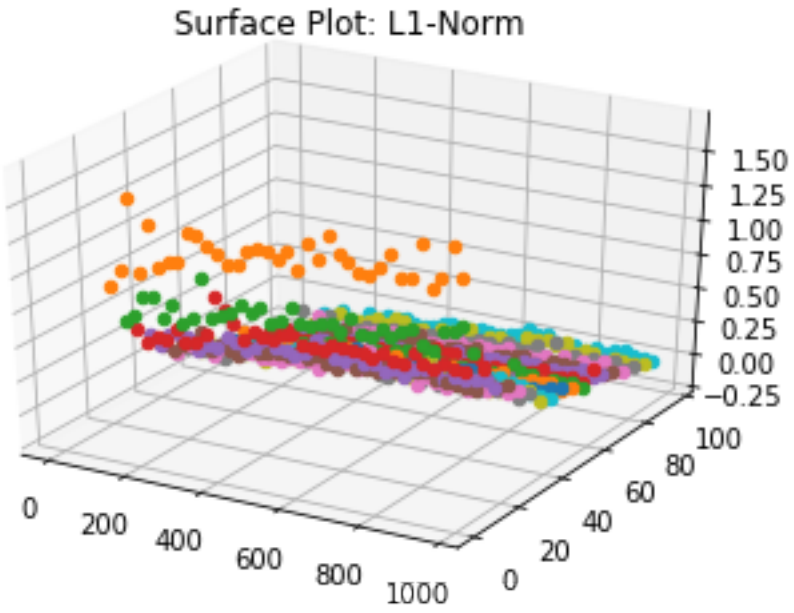



We can see from this plot that changes to n are not nearly as impactful when determining the gamma value as the increase in dimensionality and tends to matter more in lower dimensional space where the extreme maximum and minimum distances or more likely to occur in relation to the increased number of randomized instances to measure between.

As the dimensionality increases, the maximum and minimum values converge causing gamma calculations to stagnate (as can be seen by the flattening of the curve with respect to the dimension).

[7]:
```
# L1 distance gamma plot
ax = plt.axes(projection='3d')

# needed for surface plot
X = np.arange(100,1000,25)
Y = np.arange(1,100,5)
Z = np.empty((len(X),len(Y)))

#indRow = 0
for i in X:
    #indCol = 0
    for j in Y:
        #gammaCalc(i,j)
        tMat = np.random.randint(low=0, high=100, size=(i,j)).astype(float)
        tMin, tMax = minAndMaxDistanceL1(tMat)
        if(tMin != 0.0):
            tGamma = math.log10((tMax - tMin)/tMin)
            #print(tGamma)
            ax.scatter3D(i,j,tGamma)
```

```
            #Z[indRow,indCol] = tGamma
        else:
            ax.scatter3D(i,j,math.inf)
            #Z[indRow,indCol] = math.inf
        #indCol = indCol + 1
    #indRow = indRow + 1
#print(Z)
#X, Y = np.meshgrid(X, Y)
#ax.plot_surface(X, Y, Z,cmap='viridis', edgecolor='none')
ax.set_title('Surface Plot: L1-Norm')
plt.show()
```




**NOTE: Python is unable to plot surfaces for spaces where length(x-dimension) != length(y-dimension)**

(See https://stackoverflow.com/questions/46607106/python-3d-plot-with-different-array-sizes for details)

For this reason, I opted to simply use a scatter plot to plot each indiviudal (x,y,z) (or in this case: (n,d,gamma)) at various points within the interval 100 <= n <= 1000 and 1 <= d <= 100 to capture the surface space.

### 1.2.4 Question 4)

*For each of the following, determine how it can be done using Weka and submit as part of your homework, the proper command and parameters.*

**Center data (having zero mean): Center**

Command: weka.filters.unsupervised.attribute.Center

**Removing attribute 2 to 4**

Command: weka.filters.unsupervised.attribute.Remove -R 2-4

**Removing all attributes but the last**

Command: weka.filters.unsupervised.attribute.Remove -V -R 5

**Reordering attributes 1,2,3,4,5 as 5,4,1,2,3**

Command: weka.filters.unsupervised.attribute.Reorder -R 5,4,1,2,3

**Removing instances with missing values:**

Command: weka.filters.unsupervised.instance.RemoveWithValues -S 0.0 -C 4 -L first-last -V -M

*Note: -C 4 means that this is only check for missing values in the 4th attriubute column. If we want to check for missing values in the first column, we would have to change this value to 1/first, 2 for second column, etc.*

**How is a missing value denoted in an ARFF file?**

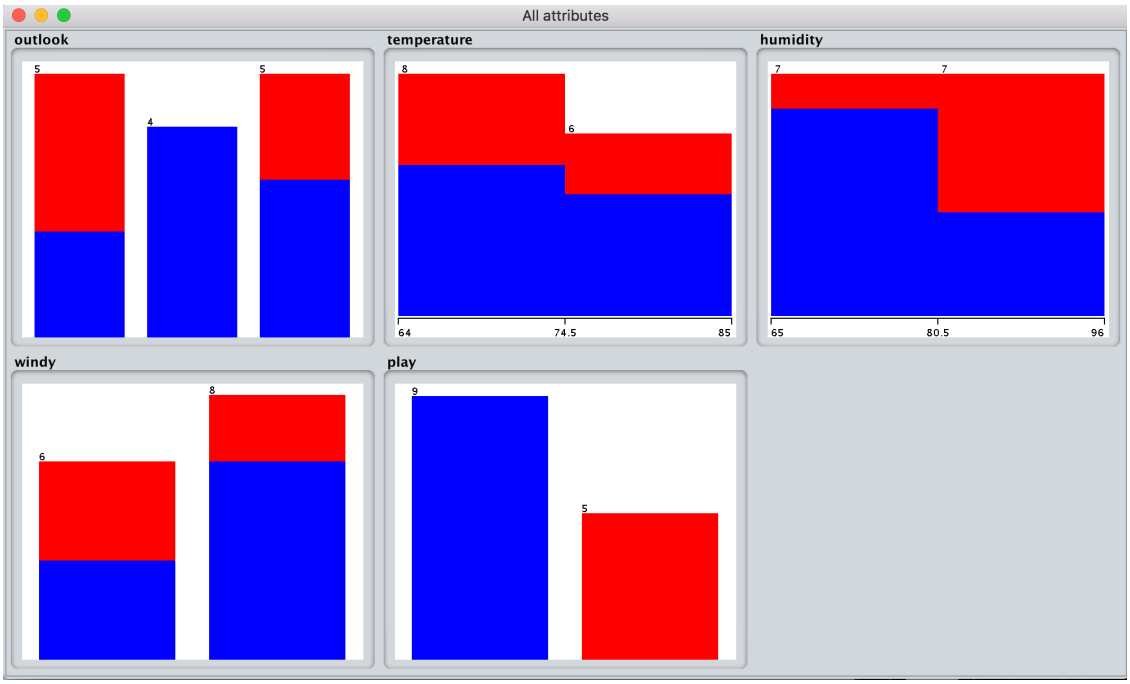
Missing values are denoted as a single '?' character, such as:

@data 4.4,?,1.5,?,Iris-setosa

for an instance of the iris data set where attributes 2 and 4 are missing.

**What does "visualize all" do?**

Visualize all allows you to look at each individual attribute value and how it is distrubuted among the data. If the data is labeled, you can see what preportion of each attribute value is associated with the corresponding label value (see picture below for weather data example).



**Removing all instances where the 3rd feature value is equal to 'x'.**

Command: weka.filters.unsupervised.instance.RemoveWithValues -S 66.0 -C 3 -L first-last

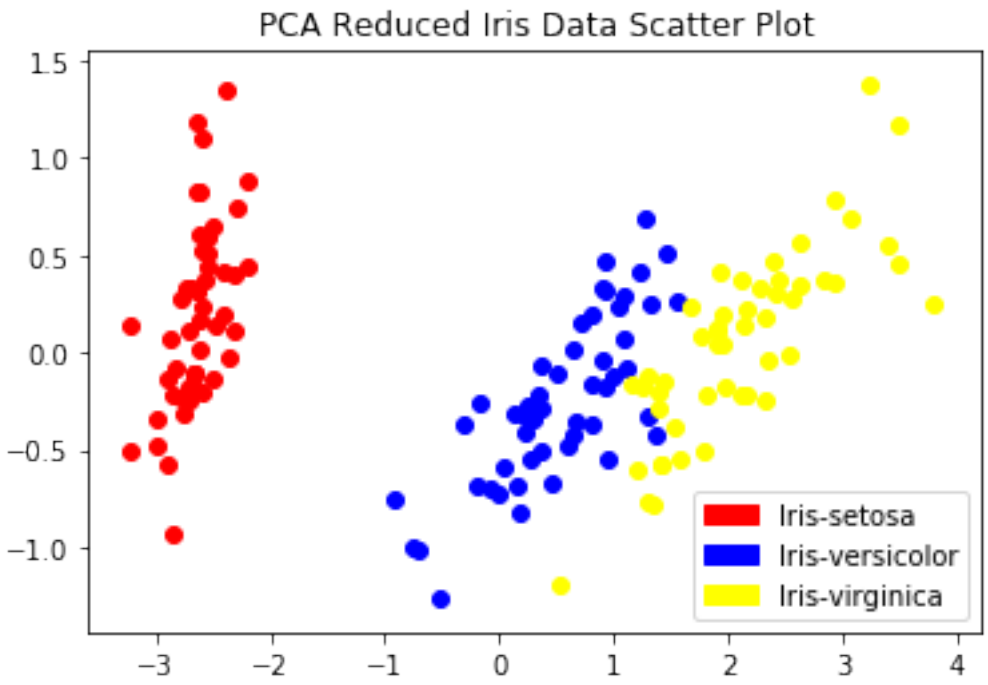
*Note: this will remove instances where the value of the 3rd attribute is less than 66.0. As far as I could tell, Weka does not have a filter which will remove only instances with a specific value, but rather removes all instances lower than (or greater than if you invert selection) a specific value. See https://weka.sourceforge.io/doc.dev/weka/filters/unsupervised/instance/RemoveWithValues.html?is-external=true for details.*

### 1.2.5 Question 5)

[8]:
```
irisSet = np.loadtxt('iris.data',delimiter = ',', dtype = str)
irisLabel = irisSet[:,-1]
irisData = irisSet[:,0:-1].astype(float)
#print(irisData.shape)
#print(irisLabel)
```

[9]:
```
pca = decomposition.PCA(n_components=2)
pca.fit(irisData)
irisData = pca.transform(irisData)
#print(irisData.shape)
```

[10]:
```
plt.title("PCA Reduced Iris Data Scatter Plot")
for i in range(len(irisData)):
    if(irisLabel[i] == 'Iris-setosa'):
        plt.scatter(irisData[i,0],irisData[i,1], color='red')
    elif(irisLabel[i] == 'Iris-versicolor'):
        plt.scatter(irisData[i,0],irisData[i,1], color='blue')
    elif(irisLabel[i] == 'Iris-virginica'):
        plt.scatter(irisData[i,0],irisData[i,1], color='yellow')
    else:
        print("ERROR: Invalid Iris Data Label")

# used for legend
import matplotlib.patches as mpatches

legend_dict = { 'Iris-setosa' : 'red', 'Iris-versicolor' : 'blue',␣
 ↪'Iris-virginica' : 'yellow' }

patchList = []
for key in legend_dict:
        data_key = mpatches.Patch(color=legend_dict[key], label=key)
        patchList.append(data_key)

plt.legend(handles=patchList)
plt.show()
```




From the above plot, we can see that Iris-setosa is the easiest to distinguish from the others (given that Iris-versicolor and Iris-virginica, while distinguishable, are clustered more closely when PCA reduced to 2 dimensions).