Graphs, Topology and Discrete Geometry
Part II - Assignment 1

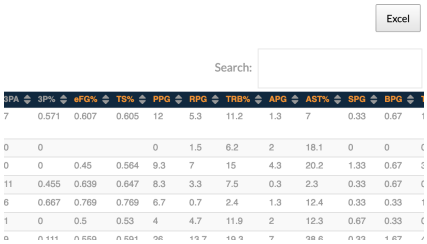
May 2022

In this exercise we are going to build a mutual k-nearest neighbor graph with NBA player of season 21/22 data, and use it as a classifier. Deliver a file in the format *your_NIU*.zip with the python code of preparation and the exercises as well as the generated images.

1 Preparation

1.1 Get the data

Go to nbastuffer website and in 'Regular Season' section click 'Excel' to download the dataset.



1.2 Preprocessing

Create a python script and load the necessary modules.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import networkx as nx
from sklearn import preprocessing
from sklearn.manifold import TSNE
```

Read the excel file with pandas and select the variables we are going to use to construct the k-NN graph for classification.

```
dataset = pd.read_excel('NBA Stats 202122 All Player Statistics \
in one Page.xlsx', header=1)

X = dataset[['RPGReboundsRebounds per game.','SPGStealsSteals \
per game.','BPGBlocksBlocks per game.','APGAssistsAssists \
per game.','2P%','3P%','FT%','TOPGTurnoversTurnovers per \
game.']].values

y = dataset['POS'].values
```

To inspect the first rows of the dataset use the `head` command

```
dataset.head()
```

Select randomly the rows which we suppose classified and consider the rest of data as non classified

```
num_objects_train = 100
num_objects_test=X.shape[0]-num_objects_train
np.random.seed(1)
random=np.random.choice(X.shape[0], num_objects_train, \
replace=False)
Index_train=np.isin(range(len(dataset)), random)

X_train=X[Index_train, :]
y_train=y[Index_train]

X_test=X[np.invert(Index_train),:]
y_test=y[np.invert(Index_train)]
```

Select the corresponding data element labels for visualization

```
X_train_labels = dataset['FULL NAME'][indices_train].values
X_test_labels = \
dataset['FULL NAME'][np.invert(indices_train)].values
```

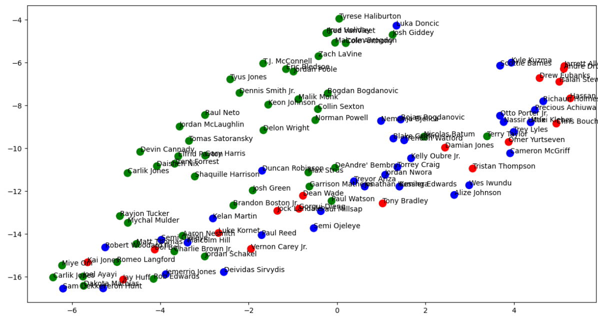


Figure 1: Player performance scatter plot. Centers are shown in red, forwards in blue and guards in green.

1.3 Visualization

Using scikit-learn's t-distributed stochastic neighbor embedding we can embed the data points into a 2-dimensional space to ease visualization.

```
X_embedded = TSNE(n_components=2,learning_rate='auto', \
init='random').fit_transform(X_train)

colors = {'C':'red', 'F':'blue',Vi 'G':'green','C-F':'red', \
'F-C':'red','G-F':'green','F-G':'blue'}

cols = [colors[i] for i in y_train]

plt.scatter(X_embedded[:n_examples_plot,0], \
X_embedded[:n_examples_plot,1] , c=cols[:n_examples_plot], s=100)

for i in range(n_examples_plot):
    txt = X_train_labels[i]
    plt.annotate(txt, (X_embedded[i,0],X_embedded[i,1]))

plt.show()
```

You should be able to see the plot with the labeled points of the dataset as in figure 1. To simplify we only consider three categories, green corresponding guards, red corresponding centers, and blue corresponding forwards.

We aim at classifying a player from the test set into a position based on the performance metrcs from the data. In this particular example, we take the first example and append it to the training features to build the graph.

```
X_clas = np.append(X_train, X_test[:1,:],axis=0)
X_clas=preprocessing.scale(X_clas)
X_clas= TSNE(n_components=2, \
learning_rate='auto',init='random').fit_transform(X_clas)
```

2 Exercises

2.1 Create distance matrix

Using only `numpy`, create a distance matrix between all the points of the previously sampled data `X_clas`

2.2 Determine K-nearest neighbors

Using numpy's `argsort` function, determine the indices of the $K = 3$ nearest neighbors of each data point.

2.3 Create the adjacency matrix of the mutual K-nn graph

With the previously obtained matrix, calculate the graph's adjacency matrix, taking in account that the number of iterations required should be the number of data points times $K = 3$.

2.4 Draw the K-NN graph

Using the `networkx` library, draw the K-NN graph for K=3, with each node in the corresponding position of the embeddings stored in X_{clas} two first columns, with the corresponding colors stored in the `cols` vector and with the name labels. The result like the one shown in figure 2

2.5 Assign the class to the new example

Using the adjacency matrix of the graph calculate the class `y_class` for the unseen example. Remember that this is done by counting the most occurring category in the neighbor examples. Check whether it is correctly classified (`y_class==y_test`).

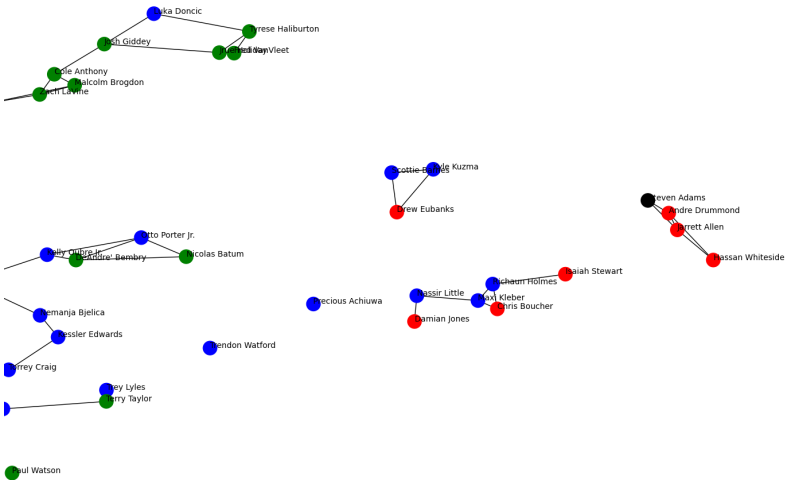


Figure 2: K-nn graph of players for K=3 with unseen example shown in black.