University of Canberra

Faculty of Science and Technology

**Programming for Data Science G (11521)**

**Week 7 Tutorial**

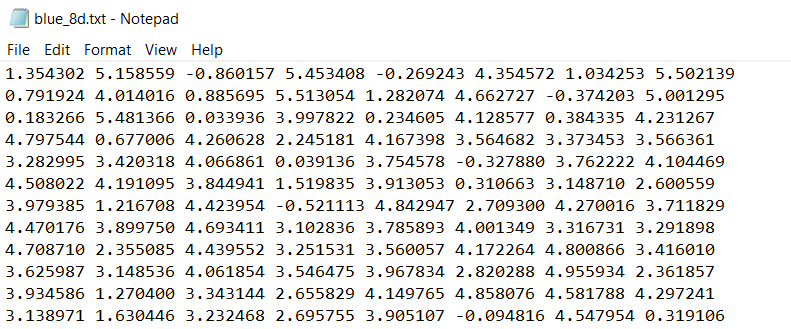
**Drawing on Canvas**

**Tasks**

* To transform data to fit a given canvas size
* To use different drawing functions in tkinter.

**Transform data to fit a given canvas size**

* Some samples in a 8-dimensional data set:



* Each line in this file has 8 values of a data sample
* Canvas is a plane that can display 2-dimensional data samples like (x1, y1), (x2, y2), etc.
* To display a multi-dimensional data set on canvas, we need to select any 2 values from 8 values in a data sample, the first value for x and the second value for y
* The index of values in a data sample starts from 0. For example, if data sample = (1.354302, 5.158559, -0.860157, 5.453408, -0.269243, 4.354572, 1.034253, 5.502139), the first value 1.354302 has index 0, the second value 5.158559 has index 1, and so on.
* If we select the first value (index = 0) for x and the third value (index = 2) for y, we will have a pair (x, y), for example, (1.354302, -0.860157)
* As the values in a data set can be very small or negative like the above data set, or can be very large, to display data samples on a canvas, we need to scale data to the canvas size.
* The function **transform\_data\_for\_canvas\_display** will return 4 values **sx** (scaling on x), **sy** (scaling on y), **tx** (translation on x) and **ty** (translation on y). A pair (x, y) will be transformed to (xx, yy) as follows **xx = x\*sx + tx** and **yy = y\*sy + ty**.
* Function header **transform\_data\_for\_canvas\_display(data\_list, idx=0, idy=1, canvas\_width=800, canvas\_height=600)** where data\_list contains multi-dimensional data samples, idx is index for x and idy is index for y value. The idx and idy are to let the function know where to get 2 values for x and y from multiple values in each data sample in the data set.
* The function **display\_data** is to display data set on canvas using the 4 values from the **transform\_data\_for\_canvas\_display** function
* Function header: **display\_data(data\_list, xi=0, yi=1, colour='red', shape='circle', canvas=None, r=5, sx=150, sy=150, tx=300, ty=200)**, where xi and yi are the indices of value for x and y, respectively, r is the radius of a small dot that represents a data sample.
* Below is the implementation of the two functions

#Function to transform data to display on canvas

def **transform\_data\_for\_canvas\_display**(data\_list, idx=0, idy=1, canvas\_width=800, canvas\_height=600):

#data\_list is a list of nD samples, n > 2

maxW = -1000000.0 #max width

minW = 1000000.0 #min width

maxH = -1000000.0 #max height

minH = 1000000.0 #min height

for sample in data\_list:

if maxW < sample[idx]:

maxW = sample[idx]

if minW > sample[idx]:

minW = sample[idx]

if maxH < sample[idy]:

maxH = sample[idy]

if minH > sample[idy]:

minH = sample[idy]

sx = canvas\_width / (maxW - minW)

tx = canvas\_width \* minW / (minW - maxW)

sy = canvas\_height / (maxH - minH)

ty = canvas\_height \* minH / (minH - maxH)

return (sx, sy, tx, ty)

#Function to display data on canvas

def **display\_data**(data\_list, xi=0, yi=1, colour='red', shape='circle', canvas=None, r=5, sx=150, sy=150, tx=300, ty=200):

for sample in data\_list:

x = sample[xi]

y = sample[yi]

x = x\*sx + tx

y = y\*sy + ty

if canvas != None:

if shape == 'circle':

canvas.**create\_oval**(x-r, y-r, x+r, y+r, outline = colour, fill=colour)

elif shape == 'square':

canvas.**create\_rectangle**(x-r, y-r, x+r, y+r, outline=colour, fill=colour)

elif shape == 'triangle':

canvas.**create\_polygon**(x, y-r, x-r, y+r, x+r, y+r, outline=colour, fill=colour)

else: #if input shape is unknown, draw circle

canvas.**create\_oval**(x-r, y-r, x+r, y+r, outline = colour, fill=colour)

#end of function

**Use different drawing functions in tkinter**

* Below are sample programs to show how to use the two functions above (note the functions are in **io\_data\_module** and the data sets are in the **datasets** folder)

**Program 1**

import io\_data\_module as iodata

import tkinter

data\_list = iodata.read\_multi\_dim\_data\_file('datasets/data\_4c\_2d.txt')

window = tkinter.Tk()

cvs\_width = 800

cvs\_height = 600

canvas = tkinter.Canvas(window, bg="white", height=cvs\_height, width=cvs\_width)

#Display data

r = 4 #radius

ix = 0 #index of value in a data sample for x

iy = 1 #index of value in a data sample for y

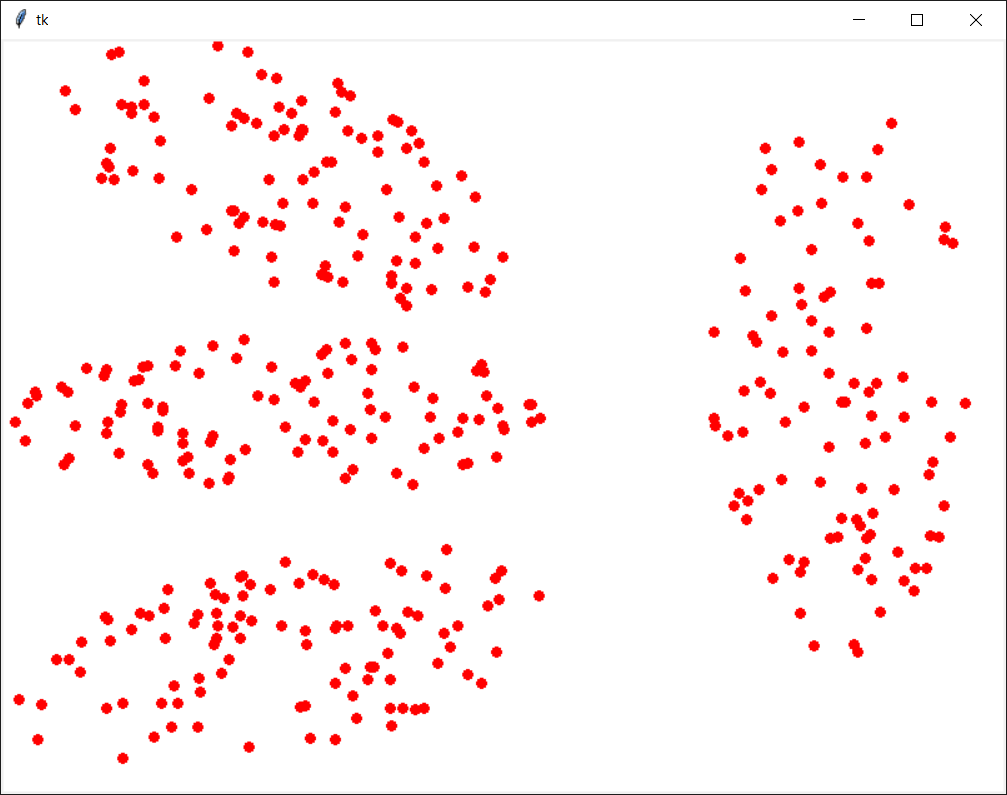
(sx,sy,tx,ty) = iodata.transform\_data\_for\_canvas\_display(data\_list, idx=ix, idy=iy, canvas\_height=cvs\_height, canvas\_width=cvs\_width)

#Show data list

iodata.display\_data(data\_list, ix, iy, 'red', 'circle', canvas, r, sx, sy, tx, ty)

canvas.pack()

window.mainloop()



**Program 2**

import io\_data\_module as iodata

import tkinter

data\_list = iodata.read\_multi\_dim\_data\_file('datasets/data\_2c\_4d.txt')

window = tkinter.Tk()

cvs\_width = 800

cvs\_height = 600

canvas = tkinter.Canvas(window, bg="white", height=cvs\_height, width=cvs\_width)

#Display data

r = 4 #radius

ix = 0 #index of value in a data sample for x

iy = 1 #index of value in a data sample for y

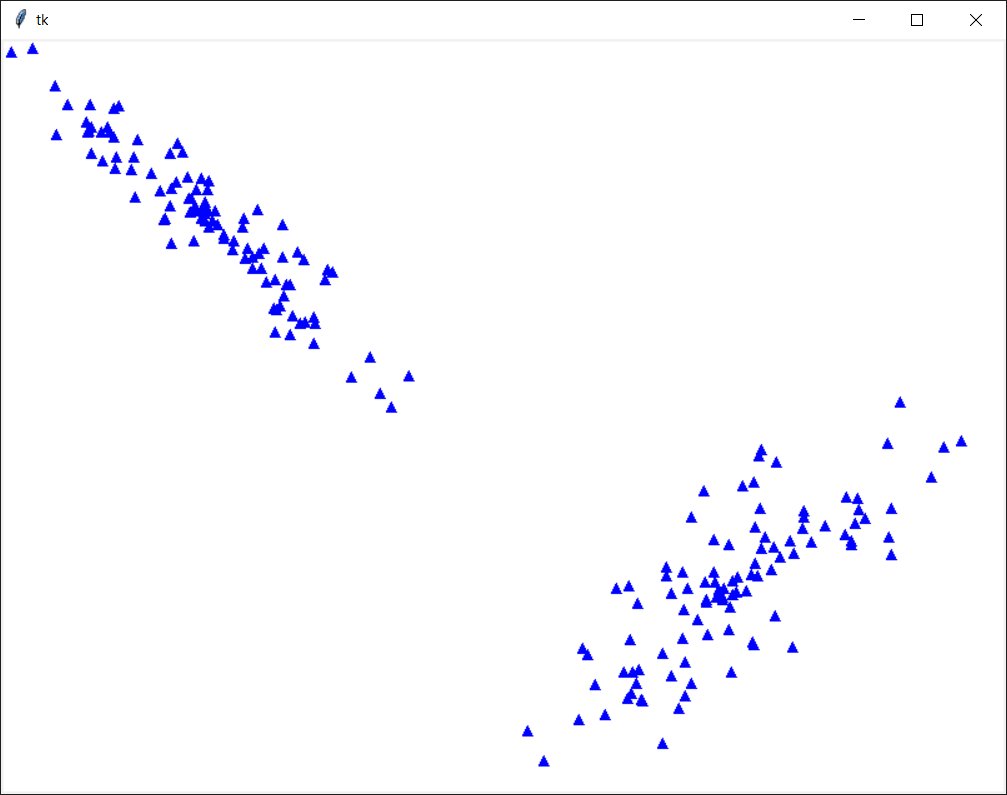
(sx,sy,tx,ty) = iodata.transform\_data\_for\_canvas\_display(data\_list, idx=ix, idy=iy, canvas\_height=cvs\_height, canvas\_width=cvs\_width)

#Show data list

iodata.display\_data(data\_list, ix, iy, 'blue', 'triangle', canvas, r, sx, sy, tx, ty)

canvas.pack()

window.mainloop()



**Program 3 (from Week 6 Tutorial, find\_nearest\_centre is the find\_nearest\_neighbour function in Week 4 Tutorial)**

import io\_data\_module as iodata

import tkinter as tk

centre\_1 = (5.1, 3.0, 1.1, 0.5)

centre\_2 = (4.4, 3.2, 2.8, 0.2)

centre\_3 = (5.7, 3.9, 3.9, 0.8)

centre\_list = [centre\_1, centre\_2, centre\_3]

data\_list = iodata.**read\_multi\_dim\_data\_file**('datasets/iris.data')

print(data\_list)

list\_1 = []

list\_2 = []

list\_3 = []

for sample in data\_list:

nearest\_centre = iodata.**find\_nearest\_centre**(sample, centre\_list)

if nearest\_centre == centre\_1:

list\_1.append(sample)

elif nearest\_centre == centre\_2:

list\_2.append(sample)

else:

list\_3.append(sample)

new\_centre\_1 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_1:

sum += sample[index]

new\_centre\_1[index] = sum / len(list\_1)

print(new\_centre\_1)

new\_centre\_2 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_2:

sum += sample[index]

new\_centre\_2[index] = sum / len(list\_2)

print(new\_centre\_2)

new\_centre\_3 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_3:

sum += sample[index]

new\_centre\_3[index] = sum / len(list\_3)

print(new\_centre\_3)

new\_centre\_list = [new\_centre\_1, new\_centre\_2, new\_centre\_3]

#Create canvas

window = tk.Tk()

cvs\_width = 800

cvs\_height = 600

canvas = tk.Canvas(window, bg="white", height=cvs\_height, width=cvs\_width)

#Display data

r = 4 #radius

ix = 0 #index of value in a data sample for x

iy = 1 #index of value in a data sample for y

(sx,sy,tx,ty) = iodata.transform\_data\_for\_canvas\_display(data\_list, idx=ix, idy=iy, canvas\_height=cvs\_height, canvas\_width=cvs\_width)

lists123 = [list\_1, list\_2, list\_3]

#Show data

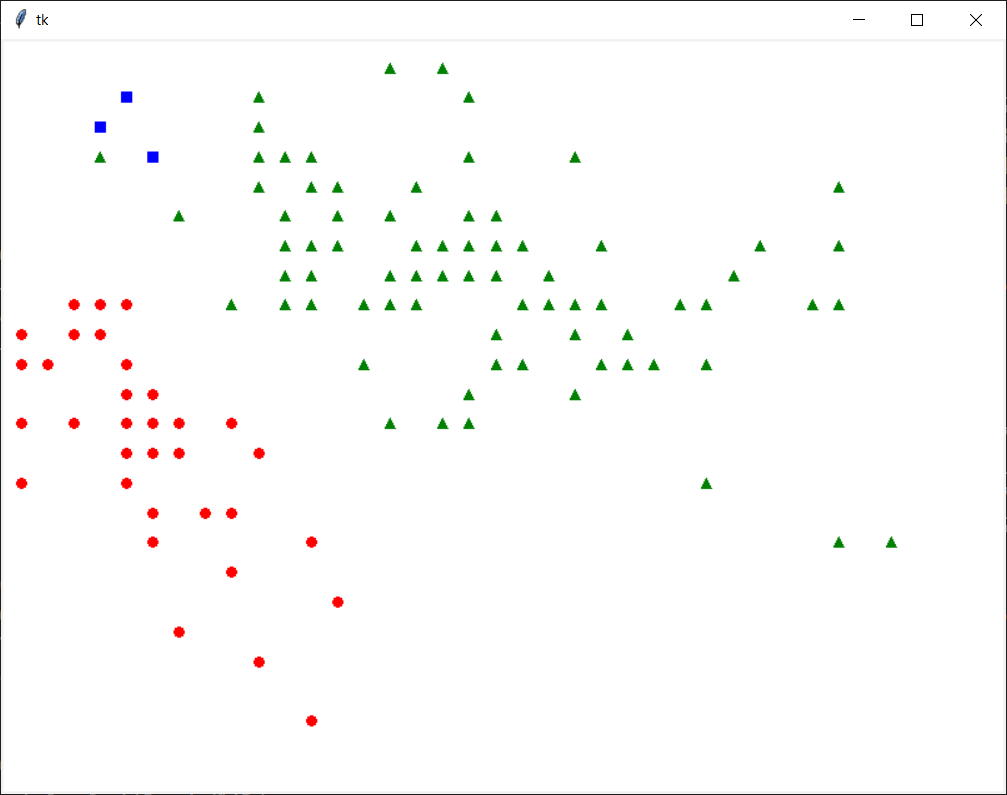
iodata.display\_data(list\_1, ix, iy, 'red', 'circle', canvas, r, sx, sy, tx, ty)

iodata.display\_data(list\_2, ix, iy, 'blue', 'square', canvas, r, sx, sy, tx, ty)

iodata.display\_data(list\_3, ix, iy, 'green', 'triangle', canvas, r, sx, sy, tx, ty)

canvas.pack()

window.mainloop()



* You can write a function to draw lines from a data sample (or cluster centre) to other data samples.