lab4_writeup_1003998757

October 15, 2020

0.1 Lab 4 - Prototypicality

This lab must be done **individually**. The required packages have been imported for you below.

[61]:
```
import matplotlib.pyplot as plt
import numpy as np
import pickle
from scipy import spatial
```

Optional: Uncomment the following line of code to sanity-check your `pearsonr` function below.

[62]:
```
# from scipy.stats import pearsonr
```

Data adapted from Leuven Concept Database (De Deyne et al., 2008).

Import data for the lab.

Hint: Check what these variables are by printing them out.

[63]:
```
with open("data-prototypicality.pickle", "rb") as f:
    birdnames, F, goodness, features = pickle.load(f)
    print(birdnames)
    print(F)
    print(goodness)
    print(features)
```

```
['eagle', 'dove', 'duck', 'pheasant', 'turkey', 'canary', 'chicken', 'crow',
'seagull', 'blackbird', 'sparrow', 'stork', 'parrot', 'parakeet', 'peacock',
'pelican', 'penguin', 'robin', 'woodpecker', 'ostrich', 'owl', 'falcon', 'swan']
[[0 0 0 … 0 0 0]
 [1 1 1 … 1 1 1]
 [4 0 0 … 1 3 0]
 …
 [1 1 0 … 1 1 1]
 [1 0 1 … 0 0 1]
 [0 1 1 … 0 0 0]]
[1.75 1.46 3.24 2.69 4.09 1.42 4.02 1.97 1.77 1.43 1.18 3.1  2.07 1.53
 3.31 2.98 4.53 1.02 1.78 4.12 2.96 1.96 3.16]
['\ufeffis attracted by shiny objects', 'is a scavenging animal', 'endangered
species', 'is bluish grey', 'builds nests', 'vomits pellets', 'brings children',
```

'hatches out other eggs', 'is brown/black', 'is a thief', 'carries a ring', 'floats on water', 'eats berries', 'eats bread', 'eats fruit', 'eats grain', 'eats insects', 'eats frogs', 'eats small animals', 'eats mice', 'eats fish', 'eats worms', 'eats seed', 'there are many kinds of it', 'filtrate food out of the water with its bill', 'sings (whistles)', 'is used in competitions', 'yellow', 'is eaten on festive dinners', 'is dangerous', 'is grey', 'grey-white', 'has two eyes', 'has downy hair', 'has a yellow bill', 'has a large wingspread', 'has a large tail', 'has a crest', 'has a hooked bill', 'has a long neck', 'has a pointed mouth', 'has a red breast', 'has a bill', 'has a tail', 'has a big bill', 'has big claws ', 'has large eyes', 'has claws', 'has short paws', 'has long paws', 'has a beautiful tail', 'has beautiful feathers', 'has an eye on its feathers', 'has a red dewlap', 'has sharp claws', 'has a bill with a crop', 'has two paws', 'has many colours', 'has feathers', 'has wings', 'is fin-footed', 'lives in a chicken coop', 'lives in a cage', 'is found in a pond', 'is grisly', 'is brown', 'the boss among the chickens', 'is dumb', 'dark colour', 'is an animal', 'is a pet', 'is a bird of prey', 'is a migratory bird', 'is a carnivore', 'is edible', 'is funny', '"is grey', 'is green', 'is big', 'is bigger than a chicken', 'is small', 'is tasty', 'is ugly', 'is sweet ', 'is beautiful', 'is not dangerous', 'is cute', 'is smart', 'is fast', 'is a symbol for peace', 'is white', 'is afraid of people', 'rises early', 'hunts', 'cackles', 'can be aggressive', 'can be a greeting', 'is good at diving', 'can be caught', 'can be fed', 'can sing beautifully', "can't fly", 'can be filled up', 'can talk', 'is not good at flying', 'can open its tail', 'can have different colors', 'can fly', 'can be bred', 'can turn his head very far', 'can swim', 'has a small bill', 'is in a cuckoo clock', 'occurs frequently', 'is found in Belgium', 'is found in the garden', 'appears in songs', 'is found in advertising or as brand ', 'appears in fairy tales and stories', 'crows', ' in the morning', 'has beady eyes', 'has peepers', 'quacks', 'has a long bill', 'lives by the sea', 'live alone', 'lives in Africa', 'lives in trees', 'live in the mountains', 'lives in nature', 'lives in the zoo', 'lives in Europe', 'herds', 'lives in the woods', 'lives in the wild', 'lives in cold areas', 'lives in tropical areas', 'lives in fields', 'lives in warm countries', 'lives in water', 'does not live in Belgium', 'also lives in the city', 'lives on a farm', 'lives on land ', 'sometimes lives in church towers ', 'lives among human beings', 'especially lives at the coast', 'lays eggs', 'lays eggs in the nests of other birds', 'does not lay eggs', 'lays big eggs', 'seems mean', "makes a 'coo' sound", "makes an 'oehoe' sound", 'makes holes in trees', 'makes a sound', 'makes an extremely irritating sound', 'makes a recognizable sound', 'builds nests at high altitude', 'makes a sharp sound', 'makes a lot of clamor', 'is messy', 'makes a strange noise', 'wakes up', 'is a male chicken', 'the male occurs in different colours', 'is mostly grey brown', "is a girl's name", 'is medium-sized', 'beautiful colours', 'nocturnal animal', 'can store food in its bill', 'has an orange bill', 'pecks', 'lives in the polar regions', 'is red', 'has a sharp view', 'graceful', 'sleeps on a stick', 'gabbles', 'jumps', 'opens its tail when it is scared', 'sometimes stands on one paw', 'sticks its head into the ground', 'stinks', 'appears in comics', 'cartoon figure', "makes a 'tok tok' sound", 'clucks on trees', 'does not migrate in the winter', 'is proud', 'chirps', 'is often released after a wedding', 'often sits still', 'flies high',

```
'flies in formation', 'is a bird', 'especially lives in the country',
'especially occurs in the winter', 'the female is brown', 'is the female of the
cock', 'waddles', 'used to carry around mail', 'is used in fights', 'is used in
shows', "is eaten when it's Christmas", 'is eaten on Thanksgiving', 'is kept by
a pigeon fancier', 'assosciated with wisdom', 'is not eaten', 'is hunted',
"often becoms a cat's prey", 'is a prototypical bird', 'says cuckoo', 'is rare',
'can see in the dark', 'sings', 'sits on threads of electricity', 'often sits in
parks', 'takes good care of her children', 'is black', 'is black and white',
'has black eyes', 'glides', 'flutters', 'has a beak', 'has air sacs', 'has
legs', 'has two wings', 'can walk', 'is able to reproduce', 'is found in trees',
'is sometimes kept as a pet', 'is sometimes eaten by man']
```

In this lab, you will construct a simple cognitive model that reproduces human judgments of prototypicality about birds (see corresponding lecture notes for details, particularly the slide that shows the big list of prototypicality ratings on different birds such as sparrows and penguins, in the experiment performed by Rosch).

All the essential variables have been imported for you, and please spend some time trying to understand their corresponding data structures, recorded in “birdnames”, “F”, “goodness”, and “features”, respectively. Specifically, you will be modelling prototypicality for the birds described in “birdnames”. The feature matrix “F” records all the features that people have come up with for these birds, as well as the applicability of the features to the birds. Each entry in this matrix records how many participants considered a feature applies to a bird: 0 indicates that no one considered a bird to have a certain feature, and a higher count (integer) indicates that at least 1 person considered a bird to have a certain feature. You will need to use all features for your model, and all features are annotated in “features”. Finally, “goodness” contains prototypicality ratings taken directly from Rosch’s list as shown on the lecture slide (mentioned above). The idea of the following exercise is for you to construct a prototypicality model, such that the model-constructed prototypicality ratings for all the birds in question should ultimately correlate with the ratings recorded in “goodness”. Note that the conceptual basis for the prototype model has been already discussed in class—see lecture notes posted.

0.1.1 Task 1 [0.5 pt]

Calculate the prototype feature vector from all birds, using the `F` variable.

[64]:
```python
# Write your code here.
# calculate entire thing or for each axis?
# https://numpy.org/doc/stable/reference/generated/numpy.mean.html
proto_feature_vec = np.mean(F,1)
print(proto_feature_vec)
```

```
[0.04347826 1.         0.95652174 0.2173913  3.73913043 0.17391304
 0.17391304 0.52173913 0.95652174 0.08695652 2.08695652 0.65217391
 1.04347826 2.04347826 1.04347826 1.73913043 1.91304348 0.34782609
 1.         0.56521739 0.91304348 1.82608696 1.65217391 1.47826087
 0.30434783 0.69565217 0.26086957 0.39130435 0.43478261 0.34782609
 0.86956522 0.56521739 4.         1.47826087 1.91304348 0.95652174
 0.43478261 0.17391304 1.2173913  0.73913043 0.82608696 0.17391304
 4.         3.95652174 0.52173913 0.43478261 0.2173913  1.91304348
 1.39130435 0.34782609 0.34782609 0.69565217 0.17391304 0.26086957
 1.17391304 0.17391304 4.         1.65217391 3.91304348 4.
 0.7826087  0.26086957 0.47826087 0.43478261 0.13043478 1.47826087
 0.         0.73913043 2.08695652 4.         0.60869565 0.39130435
 0.65217391 0.73913043 1.04347826 0.56521739 0.69565217 0.34782609
 1.56521739 1.86956522 1.52173913 1.         0.60869565 0.86956522
 2.04347826 3.17391304 0.65217391 0.47826087 1.73913043 0.17391304
 1.04347826 0.69565217 0.34782609 1.         0.2173913  1.56521739
 0.         0.47826087 0.91304348 1.86956522 0.60869565 0.86956522
 1.30434783 0.17391304 0.65217391 0.2173913  1.08695652 3.34782609
 1.2173913  1.2173913  0.82608696 1.47826087 0.         2.17391304
 2.7826087  0.91304348 0.26086957 0.17391304 0.69565217 0.43478261
 0.04347826 0.39130435 1.95652174 0.17391304 0.47826087 0.52173913
 1.43478261 1.17391304 1.69565217 0.39130435 3.56521739 1.43478261
 3.39130435 1.43478261 1.65217391 3.43478261 0.17391304 0.30434783
 0.52173913 0.91304348 0.82608696 0.73913043 0.7826087  0.69565217
 1.86956522 0.2173913  0.82608696 0.47826087 4.         0.
 0.         0.60869565 0.7826087  0.17391304 0.17391304 0.17391304
 4.         0.7826087  1.95652174 1.         1.26086957 1.82608696
 0.30434783 0.30434783 0.         0.         0.65217391 1.04347826
 0.17391304 1.60869565 0.86956522 0.2173913  0.17391304 1.95652174
 2.47826087 0.17391304 0.2173913  1.         0.39130435 0.86956522
 0.30434783 0.52173913 0.2173913  0.2173913  0.17391304 1.
 0.2173913  0.86956522 0.2173913  0.17391304 3.13043478 0.30434783
 0.43478261 0.17391304 1.13043478 0.69565217 0.17391304 4.
 1.08695652 0.26086957 1.04347826 0.17391304 0.56521739 0.17391304
 0.         0.39130435 0.30434783 0.17391304 0.17391304 0.17391304
 2.95652174 1.08695652 0.65217391 0.95652174 0.         1.7826087
 0.17391304 0.56521739 0.43478261 0.82608696 2.30434783 0.65217391
 0.34782609 2.82608696 0.73913043 0.47826087 1.         0.7826087
 1.         1.         1.         1.         0.73913043 0.43478261
 0.26086957]
```

0.1.2 Task 2 [1 pt]

Calculate Euclidean distance of each bird to the prototype feature vector.

Hint: use `spatial.distance.euclidean(x,y)`.

[65]:
```
# Write your code here.
newF = F.T

distances = []
for i in newF:
    euclidean = spatial.distance.euclidean(proto_feature_vec, i)
    print("Euclidean Distances for each bird: {}".format(euclidean))
    distances.append(euclidean)
```

```
Euclidean Distances for each bird: 17.909718982342103
Euclidean Distances for each bird: 16.587626062046706
Euclidean Distances for each bird: 17.78792309235422
Euclidean Distances for each bird: 13.273764222524095
Euclidean Distances for each bird: 16.42563582596834
Euclidean Distances for each bird: 13.889994494957271
Euclidean Distances for each bird: 16.287411493447934
Euclidean Distances for each bird: 11.514517835453935
Euclidean Distances for each bird: 13.405765765729143
Euclidean Distances for each bird: 13.910325821534434
Euclidean Distances for each bird: 13.123882615142145
Euclidean Distances for each bird: 17.317288148237118
Euclidean Distances for each bird: 15.565338838486614
Euclidean Distances for each bird: 12.955500397445254
Euclidean Distances for each bird: 14.828636781885033
Euclidean Distances for each bird: 17.776920531736717
Euclidean Distances for each bird: 18.17479714928806
Euclidean Distances for each bird: 12.389397259804355
Euclidean Distances for each bird: 12.015270686865067
Euclidean Distances for each bird: 15.437721704246691
Euclidean Distances for each bird: 16.828333745061055
Euclidean Distances for each bird: 16.13453598396215
Euclidean Distances for each bird: 15.331744253428962
```

0.1.3 Task 3 [2.5 pts]

Write a generic function that calculates Pearson correlation between two arrays `a` and `b`.

[66]:
```
# pearson correlation:
# https://wikimedia.org/api/rest_v1/media/math/render/svg/
 ↪f76ccfa7c2ed7f5b085115086107bbe25d329cec

def pearsonr(a,b):
    # Write your code here.
    num = np.cov(a,b)[0][1]
    denom = np.std(a)* np.std(b)

    corr = num / denom

    return corr

# Parameters :

# x : 1D array
# y : 1D array the same length as x

# Returns :

# (Pearson’s correlation coefficient, : 2-tailed p-value
```

0.1.4 Task 4 [1 pt]

Calculate and report the Pearson correlation between empirical ratings in “goodness” and distances of birds to prototype.

[67]:
```
# Write your code here.
# print(distances)
pearson = pearsonr(goodness, distances)
print(pearson)
```

```
0.6063551169150772
```

0.1.5 Task 5 & Task 6 [2 pts]

Task 5: Scatter-plot “goodness” ratings (y-axis) against distances of birds to prototype (x-axis); label both axes. [**1 pt**]

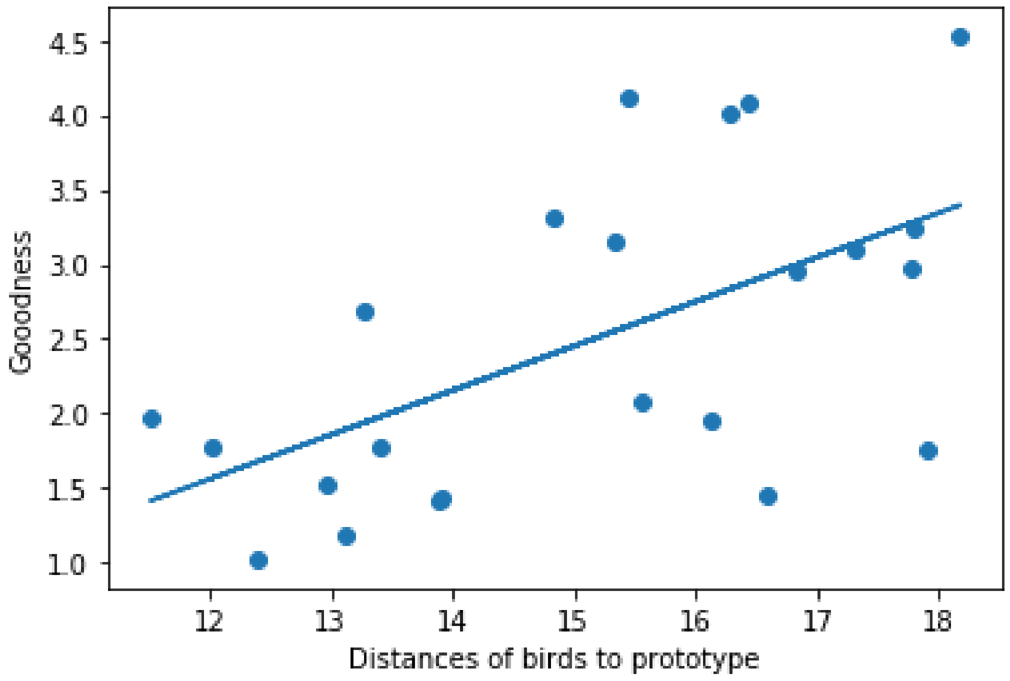
Task 6: Fit a line to these data points and show that line on the plot. [**1 pt**]

[68]:
```
# Write your code here.
plt.scatter(distances, goodness)
plt.xlabel("Distances of birds to prototype")
plt.ylabel("Gooodness")

fit = np.polyfit(distances, goodness,1)
# print(fit)

line = fit[0] * np.asarray(distances) + fit[1]
plt.plot(distances, line)
plt.show
```

[68]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



0.1.6 Task 7 [3 pts]

Task 7: a) Propose and justify a simple method of improving the correlation; b) Implement your proposal and show it works. [**3 pts**]

7. a) I propose that the prototype feature vectors should be calculated differently. Rather than calculating the average, it might be better to calculate the median. The F data contains many features and some rows have the value "0" which means the feature does not exist. This means that taking the average might not necessarily reflect the overall prototype features, as the mean can be influenced by extreme values. If we truely want a "middle value" to reflect the standard prototype, it may be better to use the median, as this captures the middle point of our data.

[73]:
```
# Write your solution here.
proto_feature_vec = np.median(F,1)
# print(proto_feature_vec)

distances = []
for i in newF:
    euclidean = spatial.distance.euclidean(proto_feature_vec, i)
    distances.append(euclidean)

new_pearson = pearsonr(goodness, distances)
print(new_pearson)
print("The original Pearson  corr is {}, and the new Pearson corr is {}.".
 ↪format(pearson, improved_corr))
print("It is evident that the newly computed correlation is better, as it is␣
 ↪closer to 1.")
```

```
0.6654645137464171
The original Pearson  corr is 0.6063551169150772, and the new Pearson corr is
0.6654645137464171.
It is evident that the newly computed correlation is better, as it is closer to
1.
```

Export and submit a **fully executable** Python Jupyter Notebook and a PDF copy of your notebook showing all results.

```
[ ]:
```