Brain_Activity_Decoder_Performance_Analysis

June 2, 2023

1 Using semantic vectors to decode brain activation

[1]:
```
# imports
import matplotlib.pyplot as plt
from copy import deepcopy
import numpy as np
```

1.1 Load the data

[ ]:
```
# Download the data extract learn_decoder.py
!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?
 ↪export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt␣
 ↪--keep-session-cookies --no-check-certificate 'https://docs.google.com/uc?
 ↪export=download&id=1xZaorRH-xxjfochvSesAhOTUg82_Xq56' -O- | sed -rn 's/.
 ↪*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1xZaorRH-xxjfochvSesAhOTUg82_Xq56" -O␣
 ↪files.zip && rm -rf /tmp/cookies.txt
!unzip files.zip
!rm files.zip

from learn_decoder import *
```

[7]:
```
# extract the data
data = read_matrix("imaging_data.csv", sep=",")
vectors = read_matrix("vectors_180concepts.GV42B300.txt", sep=" ")
concepts = np.genfromtxt('stimuli_180concepts.txt', dtype=np.dtype('U')) #The␣
 ↪names of the 180 concepts
```

1.2 What are the Accuracy scores?

Define a function that computes rank-based accuracy score, then, iterate over the 18 folds. For each fold, train the decoder **using the `learn_decoder` function** (the function is already imported from `learn_decoder.py`) on the fold train data, obtain the predicions on the fold test data, and store both the accuracy score of each concept (use the labels from `concepts`) as well as the average score of the 10 concepts.

[8]:
```
# constants
ITEM, VALUE = 0, 1
```

```
# functions from Pset 3, question 3
def cosine_similarity(v1, v2):
  return (v1 @ v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))


def find_rank(v_hat, v, vectors):
  vectors_sorted = sorted(vectors, key=lambda u: cosine_similarity(u, v_hat),␣
 ↪reverse=True)

  for i, u in enumerate(vectors_sorted):
    if np.array_equal(u, v):
      return i+1


def evaluate_fold(M, fold_data, fold_vectors, fold_concepts, vectors,␣
 ↪fold_rank_dict, concept_rank_dict, i):
  rank_sum = 0

  for data, v, concept in zip(fold_data, fold_vectors, fold_concepts):
    v_hat = data @ M

    rank = find_rank(v_hat, v, vectors)
    concept_rank_dict[concept] = rank

    rank_sum += rank

  avg_rank = rank_sum / len(fold_concepts)
  fold_rank_dict[i] = avg_rank


def get_folds(dataset, i, fold_size):
  train_folds = np.concatenate((dataset[ : (i-1)*fold_size],␣
 ↪dataset[i*fold_size : ]))
  test_fold = dataset[(i-1)*fold_size : i*fold_size]

  return train_folds, test_fold


def eval_model(data, vectors, concepts, fold_num):
  fold_rank_dict, concept_rank_dict = {}, {}
  fold_size = int(len(data) / fold_num)

  for i in range(1, fold_num + 1):
    train_data, test_data = get_folds(data, i, fold_size)
    train_vectors, test_vectors = get_folds(vectors, i, fold_size)
    train_concepts, test_concepts = get_folds(concepts, i, fold_size)
```

```
    M = learn_decoder(train_data, train_vectors)

    evaluate_fold(M, test_data, test_vectors, test_concepts, vectors,␣
  ↪fold_rank_dict, concept_rank_dict, i)

  return fold_rank_dict, concept_rank_dict
```

[9]:
```
fold_rank_dict, concept_rank_dict = eval_model(data, vectors, concepts,␣
 ↪fold_num=18)
```

[10]:
```
fold_rank_dict
```

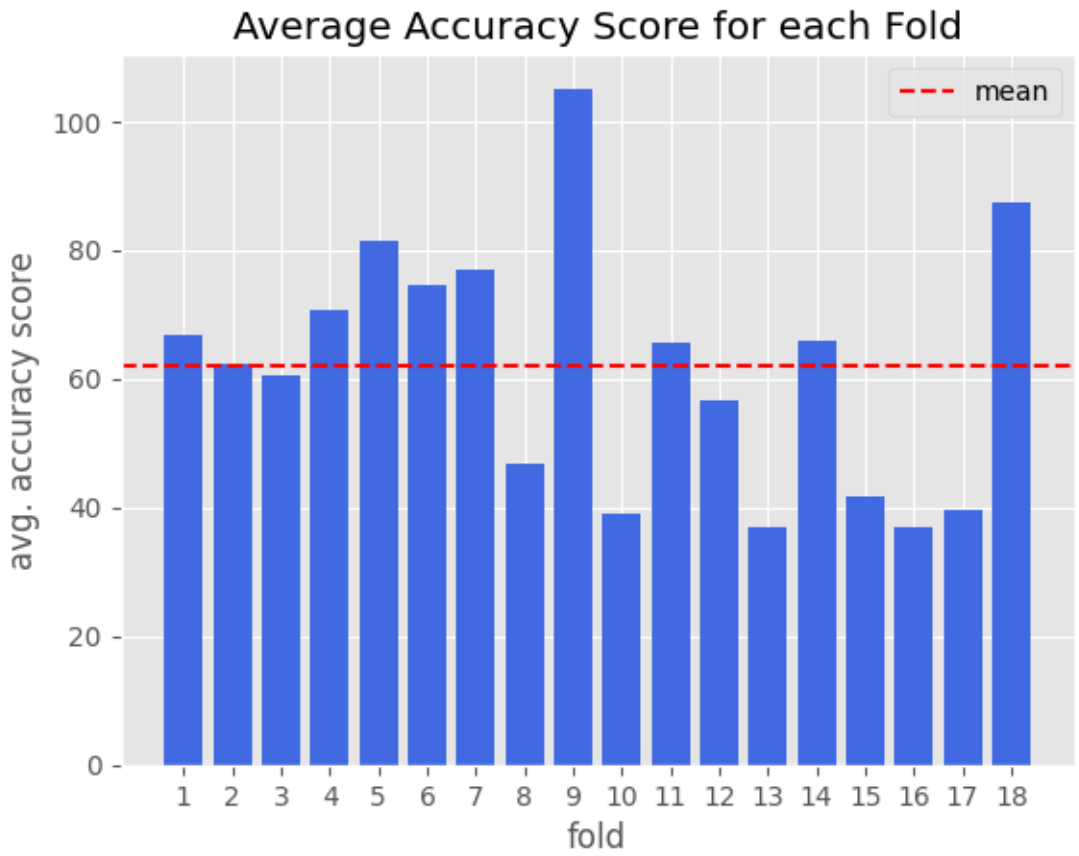
[10]:
```
{1: 66.7,
 2: 62.3,
 3: 60.4,
 4: 70.6,
 5: 81.3,
 6: 74.5,
 7: 77.0,
 8: 46.7,
 9: 105.1,
 10: 39.1,
 11: 65.6,
 12: 56.5,
 13: 36.9,
 14: 66.0,
 15: 41.7,
 16: 36.8,
 17: 39.7,
 18: 87.5}
```

Now, let us plot the average accuracy score for each fold.

[15]:
```
def display_fold_stats(fold_rank_dict):
  # create bar plot with average score for each fold
  x_vals, y_vals = list(fold_rank_dict.keys()), list(fold_rank_dict.values())
  plt.bar(x_vals, y_vals, color='royalblue')

  plt.style.use("ggplot")
  plt.title("Average Accuracy Score for each Fold")
  plt.xlabel("fold")
  plt.ylabel("avg. accuracy score")

  fold_num = len(fold_rank_dict.keys())
  mean_score = sum(fold_rank_dict.values()) / fold_num
```

```
    plt.axhline(y = mean_score, color = 'red', linestyle = '--', label="mean")
    plt.xticks([i for i in range(1, fold_num + 1)])
    plt.legend()

    plt.show()

    # print descriptive stats
    print(f"Mean accuracy score for all folds: {mean_score:.3f}")
    print(f"Lowest (best) score: {min(y_vals)}, for fold: {np.argmin(y_vals) +␣
↪1}")
    print(f"Highest (worst) score: {max(y_vals)}, for fold: {np.argmax(y_vals) +␣
↪1}")
```

[16]:
```
display_fold_stats(fold_rank_dict)
```




```
Mean accuracy score for all folds: 61.911
Lowest (best) score: 36.8, for fold: 16
Highest (worst) score: 105.1, for fold: 9
```

As we can see:

- The average score for all folds is 61.911.
- The lowest (and best) score is 36.8, and is received for fold number 16.
- The highest (and worst) score is 105.1, and is received for fold number 9.

1.3 Which concepts can be decoded with more or less success?

First, notice that the expected accuracy score of a random classifier is 90 for each concept. Thus, we will consider the model's success on each concept based on the following:

- Our model is considered to decode a concept "with more success" if its accuracy score on the concept is less than 90 (better than a random classifier).
- Our model is considered to decode a concept "with less success" if its accuracy score on the concept is greater or equal to 90 (not better than a random classifier). We consider the accuracy score 90 on a concept to be "with less success" since a simple random classifier (that requires no training at all) can perform just as good.

[17]:
```
less_success_concepts, more_success_concepts = [], []

for concept, score in concept_rank_dict.items():
  if score < 90:
    more_success_concepts.append(concept)

  else:
    less_success_concepts.append(concept)

print(f'Concepts "with more success" (a total of {len(more_success_concepts)}␣
 ↪concepts):')
print(more_success_concepts)

print()

print(f'Concepts "with less success" (a total of {len(less_success_concepts)}␣
 ↪concepts):')
print(less_success_concepts)
```

```
Concepts "with more success" (a total of 134 concepts):
['ability', 'accomplished', 'angry', 'apartment', 'argument', 'art', 'attitude',
'bag', 'bar', 'bear', 'beat', 'beer', 'big', 'bird', 'blood', 'body', 'brain',
'broken', 'building', 'business', 'carefully', 'challenge', 'charming', 'code',
'collection', 'computer', 'construction', 'cook', 'crazy', 'damage', 'dance',
'dangerous', 'delivery', 'device', 'dig', 'dinner', 'disease', 'do', 'doctor',
'dog', 'dressing', 'economy', 'election', 'emotion', 'event', 'experiment',
'extremely', 'feeling', 'fight', 'fish', 'flow', 'food', 'gold', 'great',
'hair', 'help', 'hurting', 'impress', 'investigation', 'job', 'lady', 'land',
'laugh', 'law', 'left', 'level', 'light', 'magic', 'marriage', 'material',
'mechanism', 'medication', 'money', 'mountain', 'movement', 'music', 'nation',
'news', 'pain', 'personality', 'philosophy', 'picture', 'pig', 'plan', 'plant',
'play', 'pleasure', 'poor', 'professional', 'protection', 'quality', 'reaction',
```

```
'read', 'relationship', 'religious', 'road', 'sad', 'science', 'seafood',
'sell', 'sexy', 'shape', 'ship', 'show', 'sign', 'silly', 'skin', 'smart',
'smiling', 'solution', 'soul', 'sound', 'spoke', 'star', 'student', 'stupid',
'successful', 'sugar', 'suspect', 'table', 'taste', 'team', 'texture', 'time',
'tool', 'toy', 'tree', 'tried', 'typical', 'unaware', 'useless', 'war', 'wear',
'word']

Concepts "with less success" (a total of 46 concepts):
['applause', 'argumentatively', 'ball', 'bed', 'burn', 'camera', 'charity',
'clothes', 'cockroach', 'counting', 'deceive', 'dedication', 'deliberately',
'dessert', 'dissolve', 'disturb', 'driver', 'electron', 'elegance',
'emotionally', 'engine', 'garbage', 'gun', 'ignorance', 'illness', 'invention',
'invisible', 'jungle', 'kindness', 'king', 'liar', 'mathematical', 'movie',
'noise', 'obligation', 'prison', 'residence', 'sew', 'sin', 'trial', 'usable',
'vacation', 'wash', 'weak', 'weather', 'willingly']
```

As we can see, 134 concepts were decoded with more success, and 46 with less success. Thus, many more concepts are decoded successfully with our model.

Let us display the top 5 most/least successfully decoded concepts:

[18]:
```python
sorted_concepts = sorted(
    [(concept, score) for concept, score in concept_rank_dict.items()],
    key=lambda x: x[1])

sorted_concepts = [x[0] for x in sorted_concepts]

print("Top 5 most successfully decoded concepts:")
print(sorted_concepts[:5])

print("\nTop 5 least successfully decoded concepts:")
print(sorted_concepts[-5:])
```

```
Top 5 most successfully decoded concepts:
['do', 'food', 'time', 'great', 'laugh']

Top 5 least successfully decoded concepts:
['electron', 'deceive', 'applause', 'cockroach', 'argumentatively']
```

1.4 Are the results satisfactory, in your opinion? Why or why not?

[20]:
```python
# let us display the average score for all folds again
print(f"Average fold accuracy score: {sum(fold_rank_dict.values()) /␣
 ↪len(fold_rank_dict.keys()):.3f}")
```

```
Average fold accuracy score: 61.911
```

As we can see, the average fold accuracy score is 61.911, which gives us a good estimate for the model's performance. Since this score is much lower than 90, we can conclude that our model is

much better than a random classifier. Thus, our model possesses a significant amount of predictive power.