How Do People categorize items?

Easy right?

Well, not really as it turns out.

Hugh O’Brien

My Model

- Based on a baysian perspective
- Our brains are the most powerful computers in existence
- However, not 100% rational
- Thus decision making not completely linear
- Weighting of scores depends on magnitude

My Model

- Look at past examples
- Calculate frequency of dimensions in context of each output class
- Get score/frequency for each input dimension wrt output class given
- Perform some form of aggregation/averaging.

My Model

- ‘Implicit supercomputer’
- Look at facts and past examples
- However, irrationality
- My theory: Weighting of scores depends on distance of score from 0.5
- People focus more on the extremes

A Quick Example

- INPUT
- Dim1 --> A
- Dim2 --> X
- Dim3 --> Y
- Output Classes --> A

A Quick Example

- Look at past frequencies for class A
- In dim1, ‘A’ dimension is seen in 8/10 cases, so 0.8 frequency
- Dim2, ‘X’ --> 2/10 = 0.2
- Dim3, ‘Y’ --> 5/10 = 0.5

A Quick Example

- Calcuate distance from 0.5 for each frequency
- Denoted by δ
- ‘A’ = 0.3, ‘X’ = 0.3, ‘Y’ = 0
- Sum these values = 0.6
- Weight of each frequence is thus, δ / Σδ
- So weight of ‘A’ will be 0.3/0.6 and so on

Conjunction Cases

- My theory: treated in a very similar manner to independent cases
- People tend to focus on extremes
- One independent case having a very strong score overshadows other low scores
- People weight very high scores disproportionately

My Code

- Language: Python 3
- Start by creating a dictionary with all desired frequencies

```
'A': {   'dim1': {   'A': 0.6666666666666666,
                     'C': 0.0,
                     'X': 0.16666666666666666,
                     'Y': 0.16666666666666666,
                     'Z': 0.0},
         'dim2': {   'A': 0.5,
                     'B': 0.16666666666666666,
                     'X': 0.16666666666666666,
                     'Y': 0.16666666666666666},
         'dim3': {   'B': 0.16666666666666666,
                     'C': 0.16666666666666666,
                     'X': 0.3333333333333333,
                     'Y': 0.3333333333333333},
         'length': 6},
```

My Code

- Next, average scores according to input given
- Weight according to distance from 0.5
- A proxy for ‘extremity’

```
weight_list = [abs(i - 0.5) for i in score_list]

weight_sum = sum(weight_list)

for i in score_list:

    #weight score according to its distance from 0.5
    #so very low values or very high values are weighted more highly
    #then those close to the center.
    weight = abs(i-0.5)/weight_sum
    total += weight*i
```

My Code

- Conjunction Cases:

```
max_score = max(score1, score2)
min_score = min(score1, score2)

weight = min_score/(max_score + min_score)

# under conjunction cases, does the test subject
# weight large independent values more heavily?

score = (weight*min_score + (1-weight)*max_score)
```

- Weight high scores more strongly than small ones.

My Attempt

```
Average Difference: 3.739559010340105
Cases w/ Difference > 0.5 [1, 5, 6, 9, 10, 12, 14, 16, 23, 27, 29]
Misclassified Cases [1, 10, 12, 14, 16, 27]
>>>
```

- Not perfect by any means
- 2 misclassified conjunction cases
- Many cases with a difference > 0.5
- ~ 12 cases

Results and Reflection

- Results not optimal
- Misclassification more important than absolute scores
- Misclassified 5/30 cases (~83% accuracy)
- Had better results with earlier version, but more arbitrary
- Seemed to perform well on conjunction cases

Results and Reflection

- Misclassified conjunction cases caused by misclassified independent cases
- Model’s weakness in independent cases
- My theory? Some validity.
- Definitely not confirmed however.
- Room for improvement
- Small dataset.

Questions? Thoughts?