> For each feature from the following: [cylinders, displacement, horsepower, weight, acceleration, model_year, origin] indicate how you can represent it so as to make classification easier and get good generalization on unseen data, by choosing one of: 'drop' - leave the feature out, 'raw' - use values as they are, 'standard' - standardize values by subtracting out average value and dividing by standard deviation, 'one-hot' - use a one-hot encoding. There could be multiple answers that make sense for each feature; please mention the tradeoffs between each answer. Write down your choices.

- cylinders

Raw as value is meaningful, standard

- displacement

Standard as larger variance.

- Horsepower Standard as larger variance.
- Weight Standard as larger variance.
- Acceleration Raw, or standard
- model_year Perhaps change to num of years till 2022. Use raw or standard.
- origin One-hot

> 1C) How can car name, a textual feature, be transformed into a feature which can be used by the perceptron algorithm?

1) Tokenize all words in names. And use 'one-hot' that allows several selected. 2) Factor names: extract producer name. That is take the first word. Use 'one-hot' 3) Use unsupervised learning to get clusters of names 4) Order, numerate, standardize.

> 1D)
>
> - For each feature representation, compute a classifier by running the algorithm on all the data and compare the number of errors of that classifier on the data.

Not good as classifier will fit to the training data.

> Split the data into two sets: training is 90%, test is 10%. For each feature representation, compute a classifier by running the algorithm on the training data, compare the sum of the number of errors of that classifier on the training and test data.

Not good as comparing the sum so it is similar to the above.

> Split the data into two sets: training is 90%, test is 10%. For each feature representation, compute a classifier by running the algorithm on the training data, compare the number of errors of that classifier on the test data.

Good.

> For each feature representation, perform 10-fold cross-validation and compare the resulting average error rate.

Good.

2)

> 2C) Talk with your partner about the weaknesses of the bag-of-words approach seen above. For instance, how would you interpret the feature vector (1,1,1,1,1,0,1,0)(1,1,1,1,1,0,1,0)(1,1,1,1,1,0,1,0)? (Who is selling what to whom?)

Weaknesses:

- Sentence units (subject, predicate, objective, etc.) lost, i.e. meaning is lost.
- Doesn't count frequency. So 'one plus one plus one plus one' is the same as 'one plus one' but 4 is not 2.
- Takes less important words like adjectives, conjuctions, etc.

> 2D) Words like "the", "to", "a", "and" and so on occur with very high frequency in English sentences. Do they help in detecting the sentiment in a review?

Remove them manually. Or perhaps use some threashold: remove all top ones till they some large difference.

3)

> With the help of your partner, write down the matrix corresponding to the image shown here:

```
11011
10011
11011
11011
10001

11011 10011 11011 11011 10001
```

> 3C) How well would you expect the perceptron algorithm to work on classifying images if you simply represented them as vectors? Is there any information lost when representing images this way?

Might not be linearly separable. Losing dimension info might result in bad predictions? Example is adding 11 at beginning and removing 01 at end:

```
11110
11100
11110
11110
11100
```

Seems ok? It should be classified '1'. Will perceptron classify such digits?

> 3D) What approaches might you suggest to extract more meaningful features from the images, such that the perceptron algorithm might do a good job of classification? (Hint: What makes the image of the digit '1' different compared to the image of the digit '3'?)

2 dimentions change, not 1? More dots? Larger, longer? Number of 'ends'? 3 for 3, 2 for 1.

> 4) 4A) How well would you expect the perceptron algorithm to work, if it was given these raw sound clips as input?

If it might be not lin. separable then add extra dim. Then how?

- Long to converge? What if we have very long and very large amplitude: 0.00000001 and 1? Then it will increment very slowly. So to standardize the data? So that those far will be closer?
- What if the sound is shifted towards beginning or end? But we can shift the sample to solve it in the case when we take an 1 sec interval from a larger sound clip.

> 4B) What makes a cat's meow different from a dog's bark? If you had a black box which can take in a sound clip and return the 555 most dominant frequencies heard in the sound clip, how would you use the black box and extract features from the sound clips? How would you

> dominant frequencies heard in the sound clip, how would you use the black box and extract features from the sound clips? How would you represent those features as a vector which will then be fed to the perceptron algorithm?

A meow is longer than a dog's bark. Also the bark with higher max amplitude. Use that black box to get those 5 frequencies. Then add 'height' feature (diff between min and max) in absolute value, median feature in absolute value, the top frequent one.