DATA.ML.200 Pattern Recognition and Machine Learning

TAU Signal Processing
Exercise Week 2: Conventional machine learning

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. `python` Nearest neighbor classifier (30 points)

   We continue with the same data as in the previous exercise: 20 Newsgroups.

   In the previous exercise the vocabularity included all words found in the training data. This made the Bag-of-Word (BoW) feature vectors unnecessarily long. Moreover, a curated set of English 'stop words' were used to clean uninformative words. In this exercise, the BoW feature is investigated more in detail.

   By using the 'max_features' feature of `CountVectorize()` you can limit the dictionary to the N most frequently appearing terms. Note that these would be words such as 'the' and 'and' without the stop word filtering.

   a) Run Scikit-Learn nearest neighbor classifier for different sizes of vocabulary (10, 100, 1000, etc.) to study what vocabulary size is sufficient for this task. Moreover, study whether the stop words improve accuracy at all.

   b) Plot results as graphs that show i) the full vocabulary accuracy with and without stop words (two horizontal lines of different colour) and ii) the accuracy with limited vocabularity (w/ and w/o stop words). The plot x-axis is the vocabulary size and y axis is the classification accuracy. Use plot colours and line types (or markers) so that your plots are as informative and easy to interpret as possible.

   c) Add one more graph, but in this case you do TF-IDF weighting of the feature vectors. This can be done by adding `TfidTransformer()` after the CountVectorizer(). This applies weights to the vectors and possibly improves the results. Add these results to the plot. (**Note:** you may need to convert TfidTransformer outputs from sparse to normal array for fast computation)

   Return the following items:

   - Python code: text_classifier.py that does all above.
   - PNG image: your full desktop screenshot that includes a terminal where the python file is executed:
     text_classifier_screenshot.png
   - PNG image: the final plot: text_classifier_plot.png