Hw2

The task

- Class labels: three newsgroups
  - talk.politics.guns
  - talk.politics.mideast
  - talk.politics.misc
- Training data: 2700 instances (900 for each class)
- Test data: 300 instances (100 for each class)
- Features: words
- Task:
  - (Q1-Q3) Run Mallet DT learner
  - (Q4-Q5) Build your own DT learner

Use Mallet

- mallet import-svmlight --input train.vectors.txt --output svmltrain.vectors
  - Format of train.vectors.txt: label f1:v1 f2:v2 …

- mallet train-classifier --input train.vectors --trainer MaxEnt --output-classifier m1
  - Trains MaxEnt classifier and stores model

- mallet classify-svmlight --input test.vectors.txt --classifier m1 --output res
  - Tests on the new data and the result is written to “res”

- mallet import-svmlight --input test.vectors.txt --output test.vectors --use-pipe-from train.vectors

- vectors2classify --training-file train.vectors --testing-file test.vectors --trainer DecisionTree --report test:raw test:accuracy test:confusion train:confusion train:accuracy > de1.stdout 2>de1.stderr

Q4: build a DT learner

- Each node checks exactly one feature
- Features are all binary; that is, a feature is either present or non-present

  ➔ The DT is a binary tree
- Quality measure: Information gain

Programming assignments

- Programming languages: C, C++, Java, Perl, or Python
- Write a simple shell script
- Follow the instructions in the assignments, including
  - command line format:
    cat input | foo.sh arg1 arg2 … > output
  - file format
  - the probability model
  - Naming convention: hw1.notes
- Your code MUST run on Patas

Efficiency issue

- To select the best feature, you will need to calculate the info gain for each feature
- Therefore, you will need to calculate the counts of (c, f) and (c, not f) for each class label c and each feature f.
- Try to do this efficiently.
- Report running time in Tables 2 and 3.

Patas usage

- When testing your code, use small data sets and small depth values first.
- Use condor submit for Q4 and Q5.
- Always monitor your jobs.

Condor submit example

- For a command we can run as:

  mycommand -a -n <mycommand.in >mycommand.out
- The submit description file might look like this:

```
Executable = mycommand
getenv = true
input = mycommand.in
output = mycommand.out
error = mycommand.error
Log = mycommand.log
arguments = "-a -n"
transfer_executable = false
request_memory = 2*1024
Queue
```

Some useful Condor commands

- condor_submit — submit a job
- condor_status — list available nodes and their status
- condor_q — list the job queue
- condor_rm — remove a job from the queue

For more info, see
http://courses.washington.edu/ling572/winter2017/teaching_slides/class0-condor.pdf