L90 Practical: Part II

Andreas Vlachos¹

Michaelmas 2019/20

¹This part of practical based on a design by Helen Yannadoukakis and Simone Teufel
Procedure/Timeline

- **Today:**
  - Using Support Vector Machines
  - Using a blind test set
  - Developing the extension system (doc2vec)
- **Nov 22:** Submit baseline report (get feedback Nov 29)
- **Next Demonstrated Practical: Nov 13**
  - Sanity check results
  - Analysis methods on results
- **Jan 14:** Submit report on the extension system
From last time, you should have...

- code for sign test
- code for feature manipulation (e.g., bigrams)
- code for NB + smoothing
- code for Round-Robin cross-validation
We will next add a superior classifier – Support Vector Machines

There are many parameters than can be set in SVMs, e.g. feature cutoff, kernels, . . .

You therefore need a training/validation corpus to train and tune hyperparameters and a separate blind test set

A sensible split is 90% for training/validation, 10% for testing

This is for generalisability

See instructions for more detail
SVM is a generalisation of simple maximal margin classifier and support vector classifier.

Both of these require that classes are separable by linear boundary.

Support Vector Machines can use non-linear boundaries (kernels).

Further extensions lead to multi-class SVMs.
Hyperplanes and support vectors

- A hyperplane in $p$-dimensions is a flat $p - 1$-dimensional affine subspace
- Compute the distance between data points and various hyperplanes
- Select the one that creates the largest margin (best separation) between the two classes.
- Support vectors are data points lying on the margin.
- The size of the margin = the SVM’s confidence.

Images from: https://www.saedsayad.com/support_vector_machine.htm
Doc2vec for Sentiment Analysis

- **word2vec**: learning neural word embeddings (Mikolov et al., 2013)
- **doc2vec** (Le and Mikolov, 2014): embeddings for *sequences* of words
- Agnostic to granularity: sentence, paragraph, document
- Learned ‘document’ vector effective for various/some tasks, including sentiment analysis

\(^2\) Or paragraph vectors, or document vectors ...
Distributed representation of words

Task: predict the next word given the context

Optimisation objective:

\[
\frac{1}{T} \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \ldots, w_{t+k})
\]

Softmax output layer:

\[
p(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{\exp y_{w_t}}{\sum_i \exp y_i}
\]

\[
y = b + U h(w_{t-k}, \ldots, w_{t+k}; W)
\]
Add paragraph token: each paragraph mapped to a unique vector

Paragraph vector now also contributes to the prediction task

- Shared across all contexts from the same paragraph

Works as a “memory” of context / topic
Doc2vec: distributed bag of words (dbow) architecture

Alternatively, train paragraph vector to predict words in a window (no word order); similar to Skip-gram model.
Doc2vec

- Our level of granularity: document / review
- Parameters:
  - Training algorithm (dm, dbow)
  - The size of the feature vectors (e.g., 100 dimensions good enough for us)
  - Number of iterations / epochs (e.g., 10 or 20)
  - Context window
  - Hierarchical softmax (faster version) . . .
- A number of available tools (e.g., gensim python library)
Paragraph embeddings achieved through word2vec training can be used as features within a typical supervised machine learning framework.
Questions?