L90 Practical: Part II

Simone Teufel[1]

Michaelmas 2018/19

[1]This part of practical based on a design by Helen Yannadoukakis

Procedure/Timeline

- Today:
  - Using Support Vector Machines
  - Using a validation corpus
  - Developing the extension system (doc2vec)
- Nov 16: Submit baseline report (get feedback Nov 24)
- Next Demonstrated Practical: Nov 21
  - Sanity check results
  - Analysis methods on results
- Jan 17: 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

Validation corpus

- 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 validation corpus.
- Validation corpus is a similar status to test and training corpus
- We first set parameters on the validation corpus, which is never used for training or testing.
- This is for generalisability.

What you should do

- Declare fold 1 (n=10 Round Robin Xval) as validation corpus
- You can now set all your parameters to your heart's content on this validation corpus, without risking overtraining.
  - Train on all remaining 90%
  - Test each parameter on the validation corpus
- After parameter setting, run an entirely new experiment, using only the information of what parameters work best.
- This entirely new experiment is a cross-validation as you did before.
- Note: you have lost some data, and your folds are now a bit smaller.

Standard way to use the validation corpus

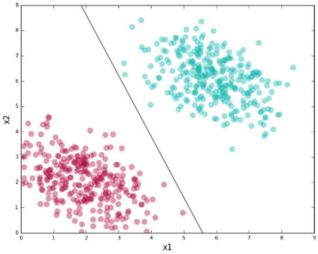
- Work with a 10-10-80 split (validation, test, training)
- Set your parameters by training on the 80% training split
- Choose the best parameters by comparing results on the validation split
- Then test the best system, with the supposedly best parameters, only once, on the test data.
- Not done here, as we want to compare to published cross-validated results.

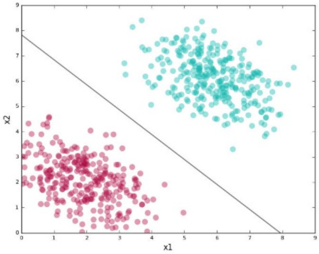
New: Support Vector Machines

- 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

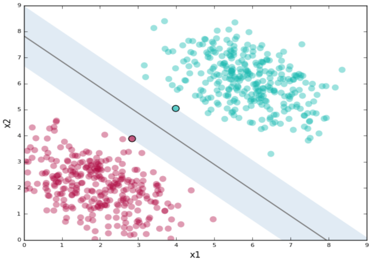
- A hyperplane in p-dimensions is a flat $p-1$-dimensional affine subspace
- Compute the distance between data points and various hyperplanes
- Select the one that creates the largest margin (best separation) between the two classes.
- Support vectors are data points lying on the margin.





Images from: https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

Support Vector Machines



- SVM-Light: implementation of Support Vector Machines (Joachims, 1999)
  - Easy to use (just download the binaries and convert your features to SVM-Light format)

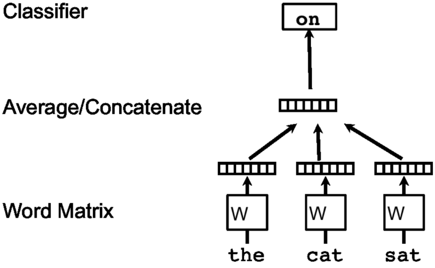
Doc2vec for Sentiment Analysis

- word2vec: learning neural word embeddings (Mikolov et al., 2013)
- doc2vec (Le and Mikolov, 2014):[2] 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:

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

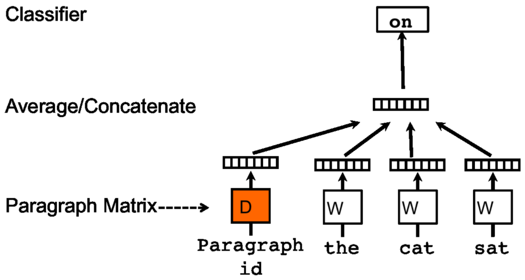
Softmax output layer:

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

$$y = b + U\,h(w_{t-k},\dots,w_{t+k};W)$$

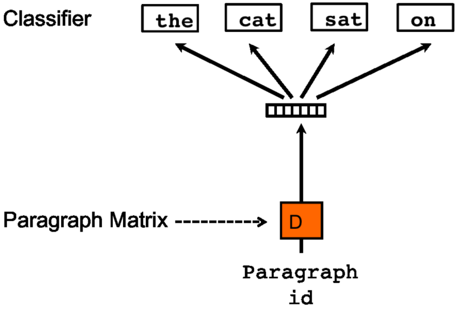
Images and formulas from paper though note inaccuracies...

Doc2vec: distributed memory (dm) architecture



- 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)

Doc2vec: how can we use it for sentiment analysis?

- Paragraph embeddings achieved through word2vec training can be used as features within a typical supervised machine learning framework

Questions?