University of Applied Sciences RheinMain
Prof. Dr. Adrian Ulges

03.11.2016

Machine Learning

Course Work 2

to complete by: 10.11.2016

Please execute this course work (as well as any following) in teams of two.

Exercise 2.1 (News Classifier)

In this exercise, you will implement a **news classifier** based on Logistic Regression. You will train and test the classifier on articles from the New York Times, and categorize them into sections (like *arts, technology, sports, ...*).

- Download the file `nytimes_classifier.zip` from the lecture homepage. You will find a folder `nytimes_data` containing 2903 news articles crawled from the New York Times (split into training and test data, as well as 8 categories). In `classifier.py` and `preprocess_documents.py`, you will find some code for reading input documents from the command line. Read through the code, particularly `classifier.py`, where the abstract class `Classifier` defines an interface for document classification. Internally, your classifier represents each document by a **bag-of-words** feature: a python dictionary mapping each term t to its number of appearances in the document.

- Have a look at the data. What error rate (= percentage of misclassified documents) would you expect a classifier to achieve on this problem? Write it down on a piece of paper ;-).

- Implement your own `LogisticRegressionClassifier` subclassing `Classifier`. Use the class `LogisticRegression` from `sklearn`. When calling the script with `--train`, train the model using `train()` and store the trained model to disk (use Python's `pickle` module for that). When calling the script with `--apply`, load the model and apply it to the documents specified.
  Hints: (1) Feel free to modify the code as much as you need it! (2) Use sklearn's `DictVectorizer` *to turn the feature dictionaries into numpy arrays (which sklearn can work with). (3) Use 'one-vs-rest' classification (check out sklearn's* `LogisticRegression` *docs).*

- The folder `nytimes_data` contains two subfolders, `train` und `test`. Train your classifier on the documents in `train`. Apply it to the documents in `test`. Measure the error rate on `test`. Also, output the incorrect documents' titles, plus the classifier's decision.

Exercise 2.2 (Text Classifier: Inspection)
Your `LogisticRegression` classifier assigns weights to each term, indicating how important the term is for the respective class (positive/negative weights indicate that the appearance of the term increases/decreases the likelihood of belonging to a certain class). Write code that picks the K most important (i.e., highest weighted) terms for each category and prints them. This way, check if your classifier learned reasonable weights!
Hints: (1) You find the weights in the `coef_` *attribute. Also, sklearn's* `DictVectorizer` *has a method* `get_feature_names()`*, which maps feature indices back into the original terms.*

Exercise 2.3 (Text Classifier: Bigrams)
Add bigrams (=2-word-ngrams) as features to the classifier. You will have to modify the code in `preprocess_documents.py` for that, and play around with the thresholds `min_frequency` and `min_doc_frequency` (try to reach $> 50k$ features).

- Does this improve accuracy? Test it both on `nytimes_data` and `nytimes_data_2016`.
- Again, inspect the classifier's highest weighted features. Have any bigrams been learned to be most important for your categories?

Exercise 2.4 (Report)
Put together a presentation of 2-3 slides, summarizing your results and any open questions.