CS771: Machine learning: tools, techniques, applications
Assignment #1: Naive Bayes, NN, LDF

Due on: 31-1-2016, 23.59 21-1-2016
MM:225

You are free to use any open source or free platform or library. Even different ones for different questions. Avoid priced s/w like MATLAB.

Cross validation will be discussed class.

1. Consider the spam data set available on the course ftp site in the directory 'assgnData', file spam.tar.gz. It has 10 sub-directories containing spam and non-spam mails, along with the subject. The spam mails have file names starting with 'spm'. The data set is not balanced. Spam mails are much fewer than non-spam ones. Use the naive Bayes algorithm to build a classifier to classify mail as spam or non-spam. Report 10-fold cross validated results using nine sub-directories for training and the tenth one for testing in each instance.
   (a) First do it on the mail subject+body as is.
   (b) Then repeat after stop words (fluff or non-content words) have been removed.
   (c) Repeat a third time after removing stop words and lemmatizing the email. This means retaining only the lemma or root of each word. So, for example inflectional markers (e.g. tense) for a verb will be lemmatized to just the root verb.
   Stop word lists and lemmatizers are available on the internet.

[75]

2. Build a spam/non-spam classifier using a linear discriminant function method using a bag of words (BoW) representation in three different ways:
   (a) Use a binary BoW representation. Value is 1 if the word occurs in the subject/body of the mail, 0 if it does not.
   (b) Use a term frequency based BoW representation. Term frequency counts the number of times a word occurs in the document (normalized by the length of the document).
   (c) Use the tf-idf BoW representation.

   Again you should report 10-fold cross validated results.

   Note on text representations: Vector space representations for text define a vector space with dimension $|\mathcal{V}|$, where $\mathcal{V}$ is the vocabulary for the documents (that is the set of words in the documents). There are different ways of encoding a document as a vector. The common ones are:

   1. Binary frequency: The entry corresponding to the word in the vector is 1 if the word occurs in the document and 0 otherwise.
   2. Term frequency $tf(t,d)$: Let the raw frequency of the term t in document d be $f(t,d)$. There are numerous definitions for $tf(t,d)$ using $f(t,d)$. One commonly used one that normalizes for the length of the document is defined as:

$$tf(t,d) = 0.5 + 0.5 \times \frac{f(t,d)}{max\{f(t',d)|t' \in d\}}$$

Another one is log scaled: if $f(t,d) \neq 0$ then $1 + log(f(t,d))$ else 0.

3. Inverse document frequency $idf(t, \mathcal{D})$: If a word (or term) occurs in every document of a collection of documents $\mathcal{D}$ then it does not contain much information about the document (e.g. the word 'the'). To correct for this the inverse document frequency $idf(t, \mathcal{D})$ is defined. This is the log scaled ratio $log(\frac{|\mathcal{D}|}{|\mathcal{D}_t|})$, where $\mathcal{D}_t$ is the subset of documents from $\mathcal{D}$ that contain the term t. Once again many slightly different definitions exist for $idf(t, \mathcal{D})$. A commonly used definition is $idf(t, \mathcal{D}) = log(1 + \frac{|\mathcal{D}|}{|1+\mathcal{D}_t|})$. 1 is added in the denominator to avoid division by 0.

The entry for term t in the vector for document d is given by $tf(t,d) \times idf(t, \mathcal{D})$. This is high whenever the term is frequent in d but occurs infrequently in $\mathcal{D}$.

The vocabulary $\mathcal{V}$ for a collection of documents is usually constructed from the set of all words present in $\mathcal{D}$ after the stop words have been removed and lemmatization has been done. So, a vector contains essentially content words. Note that the BoW representation does not take order of word occurrence into account - that is the reason it is called a bag. A query document may contain words not in $\mathcal{V}$. These words are just ignored since the query document must also be represented in the same vector space. [75]

3. Build a k-NN classifier for the MNIST data set available on the ftp site in directory 'assgnData'. This data consists of images of handwritten numerals from 0 to 9. There are 4 files. Training and test data are separate and the images and labels are in separate files giving a total of 4 files. You should use the training data as the model and the test data for testing. Test with different values of k (say 1 to 4) to see which one works best. Also, try with different metrics - at least 3 different metrics.

   Note: MNIST is a very popular data set for trying out algorithms. Visit http://yann.lecun.com/exdb/mnist/ for more information especially related to performance of different kinds of algorithms. [75]