NEWS ARTICLE CLASSIFICATION

Document/Text classification is one of the important and typical task in *supervised* machine learning (ML). Assigning categories of documents, which can be a web page, library book, media articles, gallery etc. has many applications like e.g. spam filtering, email routing, sentiment analysis etc. Here we demonstrate news article classification using python.

We can divide the classification problem into below steps:

1. Loading the data set in jupyter.
2. Extracting features from text files.
3. Running ML algorithms.
4. Grid Search for parameter tuning.

1.Loading the data set in jupyter

The data set will be using for this example is the famous "20 Newsgroup" data set.

2.Extracting features from text file

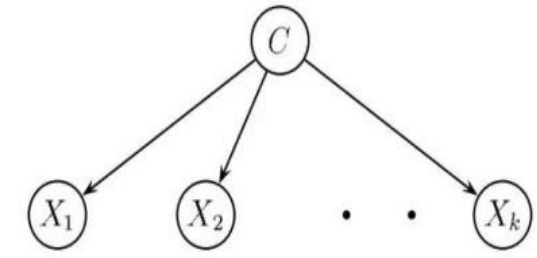
Text files are actually series of words (ordered). In order to run machine learning algorithms we need to convert the text files into numerical feature vectors. We will be using **bag of words** model for our example. Briefly, we segment each text file into words (for English splitting by space), and count # of times each word occurs in each document and finally assign each word an integer id.

Dataset Description

- 20,000 articles
- Hierarchical clustered category
- 20 categories at leaf level
- Evenly 1000 articles per category
- Train vs Test ratio: 80% vs 20%
- Randomly split training data into labeled and unlabeled data set.
- Labeled doc size: 200 ~ 5076
- Unlabeled doc fixed size: 10000

Mechanics - Naive Bayes Classifier

Multinomial Naive Bayes (with m-estimate smoothing)



$$p(C_k \mid x_1, \ldots, x_n) \propto p(C_k, x_1, \ldots, x_n)$$
$$\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots$$
$$\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

Training

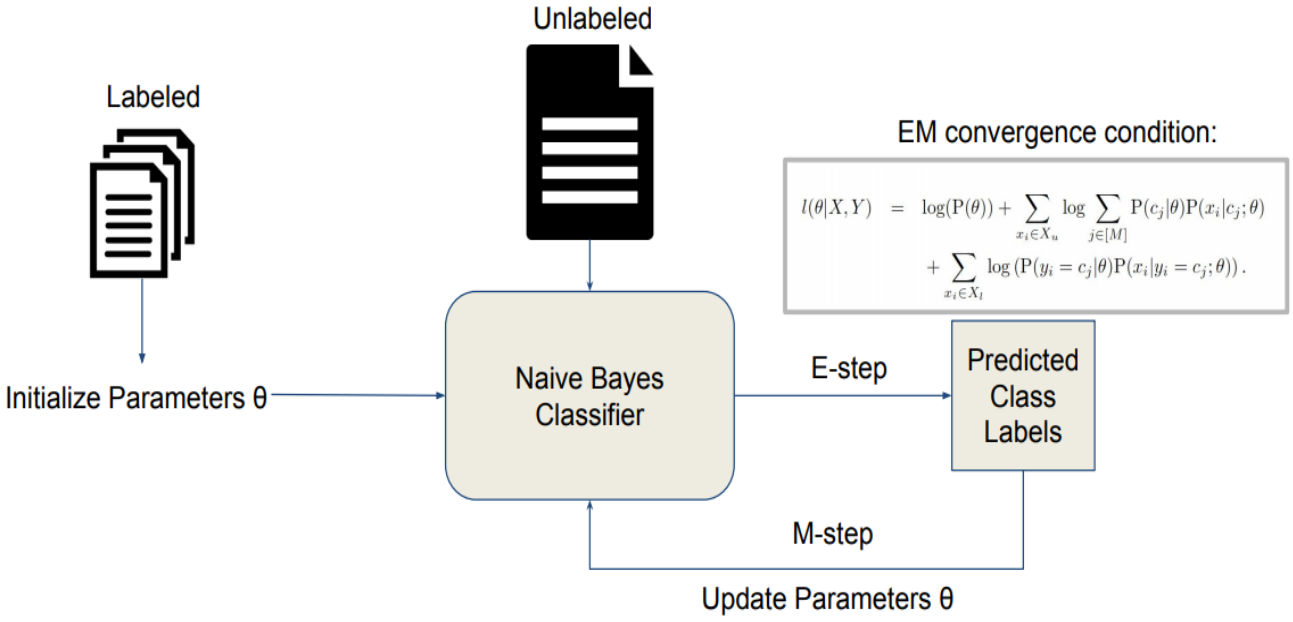
$$\frac{\square\square + \sum_{x_i \in X} \delta_{ij} x_{it}}{\square\square |\mathcal{X}| + \sum_{s=1}^{|\mathcal{X}|} \sum_{x_i \in X} \delta_{ij} x_{is}},$$

$$\frac{1 + \sum_{i=1}^{|X|} \delta_{ij}}{M + |X|}.$$

Tune parameter α using Grid Search

- **Inputs:** Collections X_l of labeled documents and X_u of unlabeled documents.
- Build an initial naive Bayes classifier, $\hat{\theta}$, from the labeled documents, X_l, only. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(X_l|\theta)\mathrm{P}(\theta)$ (see Equations 1.5 and 1.6).
- Loop while classifier parameters improve, as measured by the change in $l(\theta|X,Y)$ (the log probability of the labeled and unlabeled data, and the prior) (see Equation 1.8):
  - **(E-step)** Use the current classifier, $\hat{\theta}$, to estimate component membership of each unlabeled document, *i.e.*, the probability that each mixture component (and class) generated each document, $\mathrm{P}(c_j|x_i;\hat{\theta})$ (see Equation 1.7).
  - **(M-step)** Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta \mathrm{P}(X,Y|\theta)\mathrm{P}(\theta)$ (see Equations 1.5 and 1.6).
- **Output:** A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

Table 1.1 The basic EM algorithm for semi-supervised learning of a text classifier.



Mechanics - Text Preprocessing Dimension Reduction Use NLTK library and Regular Expression:

- Remove Noisy Symbols (punctuation, digit, web links, etc.)
- Stemming (reduce derived words to their root form)
- Lemmatization (reduce inflected word forms to single item)
- Remove stopwords and rare words

Conclusion Advantage:

- Massive cheap dataset in real case
- Significantly improve accuracy given small labeled dataset
- Labeled data for parameter initialization
- Simple implementation and parameter tuning
- Good scalability (online learning)
- Works well for categorical data (i.e. text in word vector)

Limitation:

- Strong assumption of feature independence
- Suffer noisy data distribution and highly correlated features
- Time consuming in computation of log likelihood over EM iterations