# Mosaic Machine Learning Exercise

Xiaoyu Bai

In the classification of *20Newsgroups* dataset, I use python and scikit-learn package for implementation. Since there is built-in function in scikit-learn package to fetch the data, I download the data through this package instead of manually doing so from the website.

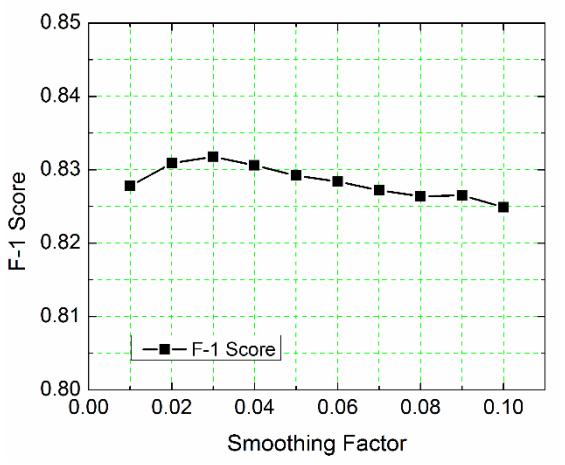
**Tokenization**

First of all, we need to transform the text data into a numerical vector for classification. For Q1 (implementation of 4 traditional ML approaches), I am using tf-idf as the tokenization method. The pros of tf-idf includes that the inverse-document-frequency will alleviate the impact of some common meaningless words; the vector is normalized so the classification is based on cosine similarity and the length of the article will not affect topic classification. The cons is that it is still based on bag-of-words so the correlation of adjacent terms will not be taken into consideration.

## 1. Multinomial Naive Bayes Model

To begin with, multinomial naive Bayes model may serve as a good baseline with the assumption that among features there is no correlation.

The tokenization construct vector dimension of 12979 however each vector is actually in sparse format so it can fit into the memory. The training set size is 11314 and test set size is 7532. The only parameter tuned in multinomial naive Bayes is the additive smoothing factor to estimate the probability.




The F-1 score for multiple classes is calculated as the unweighted mean of all classes.

With optimum smoothing factor of 0.03, the multinomial naive Bayes model achieves:

$$F-1\ score = 0.8386$$

$$Test\ Accuracy = 0.8336$$

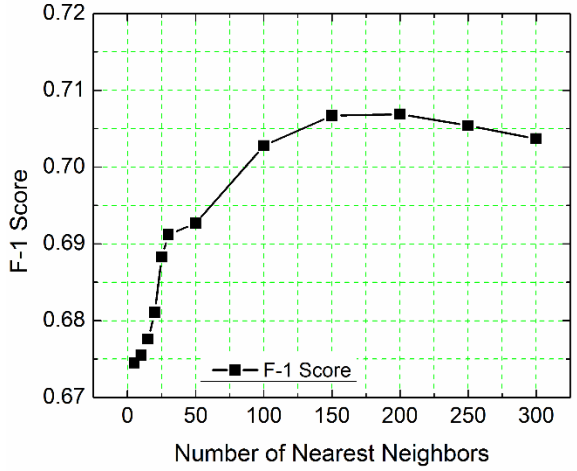
$$Training\ Accuracy = 0.9949$$

Another interesting thing to look at is the features which plays most important role in classification for each topic. The table below lists the top 10 most informative features for each topic based on Naive Bayes model.

| Topic | Top 10 most informative features |
|---|---|
| alt.atheism | *people sgi atheism com livesey atheists caltech god edu keith* |
| comp.graphics | *polygon subject com university lines files 3d image edu graphics* |
| comp.os.ms-windows.misc | *com ax ms drivers driver files dos file edu windows* |
| comp.sys.ibm.pc.hardware | *disk pc com controller bus edu card ide drive scsi* |
| comp.sys.mac.hardware | *scsi simms monitor centris se quadra drive edu apple mac* |
| comp.windows.x | *application x11r5 xterm widget edu com server mit motif window* |
| misc.forsale | *lines condition distribution university new shipping offer 00 edu sale* |
| rec.autos | *organization subject oil writes article engine edu cars com car* |
| rec.motorcycles | *writes ca article motorcycle bikes ride edu dod com bike* |
| rec.sport.baseball | *article com runs games players game team year baseball edu* |
| rec.sport.hockey | *games season players play nhl edu ca game team hockey* |
| sci.crypt | *crypto nsa government escrow keys com chip encryption clipper key* |
| sci.electronics | *university ca organization circuit power subject lines use com edu* |
| sci.med | *science dyer cs com msg gordon banks geb edu pitt* |
| sci.space | *gov digex access orbit alaska moon henry edu nasa space* |
| soc.religion.christian | *faith people christ christian bible edu church christians jesus god* |
| talk.politics.guns | *firearms weapons batf stratus fbi people com guns edu gun* |
| talk.politics.mideast | *armenia people arab armenians edu armenian turkish jews israeli israel* |
| talk.politics.misc | *state government article writes clinton people optilink cramer com edu* |
| talk.religion.misc | *objective people morality kent christian com edu jesus sandvik god* |

**2. K-Nearest-Neighbors**

Another simplistic model will be KNN in which case the training process takes very little computational cost. However the prediction process may take longer compared with other classifiers. The only parameter tuned in KNN model is the number of nearest neighbors.




With optimum K value of 200, the KNN model achieves:

$$F-1\ score = 0.7069$$

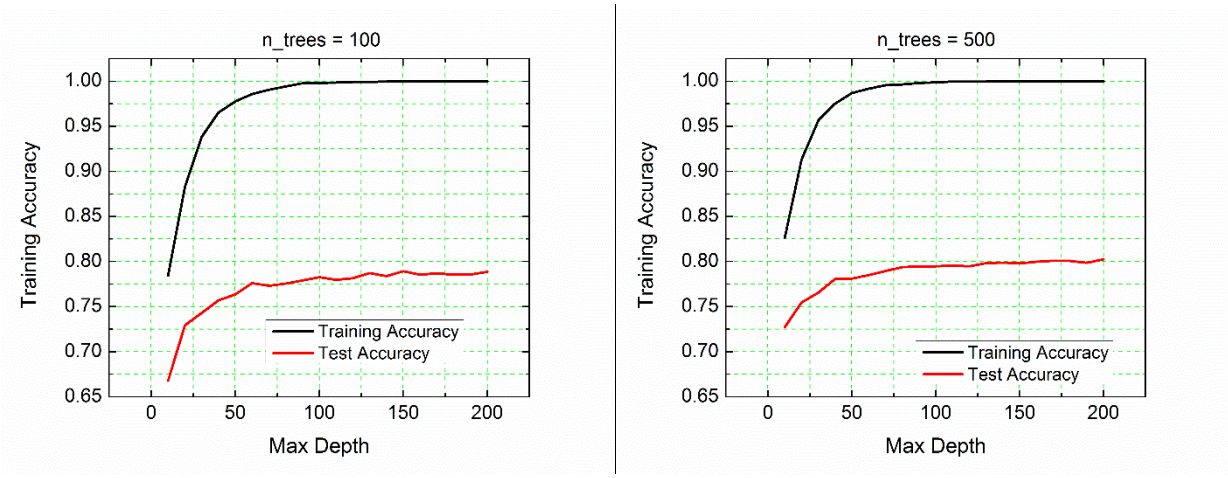
$$Test\ Accuracy = 0.7246$$

$$Training\ Accuracy = 0.7792$$

The model is under-fitted since the training accuracy is also very low. But due to the limitation of the KNN model, it is hard to achieve a very high accuracy.

## 3. Random Forest

Next implemented classifier is random forest. Random forest is known for low bias but high variance. Unlike the usual sequential boosting methods, random forest implements multiple decision tree in parallel so it can be considered as an ensemble learning method.




Two models with different number of decision trees in the forest has been compared. In each model, the max depth of the tree is set as a tuning parameter. The best achievable test accuracy is about 80% however this already suffers from a nearly 100% training accuracy, which means the random forest model is not suitable for this problem.

In optimal case (number of decision trees in forest is 500 and maximum depth of the tree is 200), the random forest model achieves:

$$F-1\ score = 0.7884$$

$$Test\ Accuracy = 0.8016$$

$$Training\ Accuracy = 0.9999$$

## 4. Support Vector Machine with Linear Kernel

SVM is known to be good at text classification based on bag-of-words because the text vector has high dimensional feature space, most features are relevant, and text vectors are sparse [1]. Since most text

categorization problems are linearly separable, here a SVM model with linear kernel is implemented. Hinge loss and L-2 norm is applied.

The performance of SVM is as below:

$$F-1\ score = \ 0.8458$$

$$Test\ Accuracy = 0.8523$$

$$Training\ Accuracy = 0.9948$$

It will be interesting to look at the top 10 most informative features again and compare this with the naive Bayes model. Although the naive Bayes model already did a good job, this time the top 10 word stems seem make even more sense.

| Topic | Top 10 most informative features |
|---|---|
| alt.atheism | *mangoe rushdie jaeger cobb okcforum islamic wingate atheists keith atheism* |
| comp.graphics | *polygon images cview animation tiff 3do pov image 3d graphics* |
| comp.os.ms-windows.misc | *win winqvt risc file ini driver ax win3 cica windows* |
| comp.sys.ibm.pc.hardware | *scsi pc bus irq controller 486 vlb bios ide gateway* |
| comp.sys.mac.hardware | *lciii adb iisi lc duo centris quadra powerbook apple mac* |
| comp.windows.x | *expo xlib xpert lcs xterm window server x11r5 widget motif* |
| misc.forsale | *00 interested asking condition wanted sell offer shipping forsale sale* |
| rec.autos | *callison mazda auto ford toyota engine dealer automotive cars car* |
| rec.motorcycles | *dog helmet bmw ride motorcycles riding motorcycle bikes bike dod* |
| rec.sport.baseball | *uniforms braves tigers giants stadium pitching cubs sox phillies baseball* |
| rec.sport.hockey | *finals wings pens cup flyers leafs penguins playoff nhl hockey* |
| sci.crypt | *gtoal cryptography crypto nsa security pgp tapped key encryption clipper* |
| sci.electronics | *kolstad uv 2600 adcom tv cooling voltage 8051 electronics circuit* |
| sci.med | *syndrome medical pitt dyer treatment cancer photography disease msg doctor* |
| sci.space | *funding sci planets rockets shuttle prb launch moon orbit space* |
| soc.religion.christian | *christianity geneva fisher easter christians church rutgers christ clh athos* |
| talk.politics.guns | *feustel handgun fbi cathy firearms weapons waco atf guns gun* |
| talk.politics.mideast | *holocaust gt1091a argic serdar armenia armenian armenians turkish israel israeli* |
| talk.politics.misc | *jobs ipser drieux tax drugs clinton optilink gay cramer kaldis* |
| talk.religion.misc | *biblical tyre promise thyagi christian psyrobtw hudson beast weiss morality* |

The general performance of each traditional classifier is summarized as below:

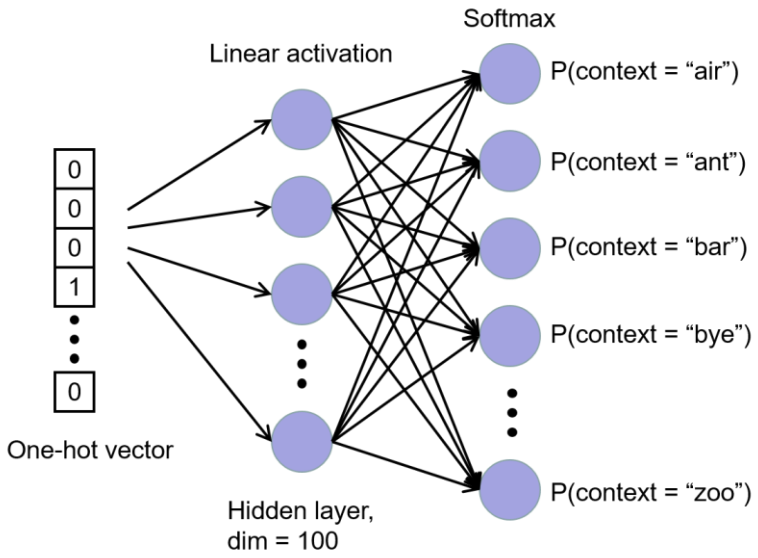
| Classifier | F-1 Score | Computational Time (s) |
|---|---|---|
| Multinomial Naive Bayes | 0.8318 | 0.20 |
| K-Nearest Neighbors | 0.7069 | 11.05 |
| Random Forest | 0.7771 | 74.55 |
| Support Vector Machine | 0.8458 | 8.90 |

## 5. Neural Network

In this section, a convolutional neural network has been trained for classification. The neural network model is based on Keras with TensorFlow backend. The network structure is derived from the structure in [2]. After running the original model, it is found that the original model suffers from severe overfitting issue. Therefore dropout layers have been added to prevent overfitting.

For all the above methods, the feature extraction is based on 1-hot-like vector which does not reflect the relation between words. For example, “dinner” and “supper” should be very close in terms of meaning. However in the previous feature extraction method, there is no way to distinguish that “dinner” has closer meaning to “supper” than “computer”. Therefore, the CNN deep learning model with word-embedding is expected to achieve better performance.

Instead of starting from training the embedding layer from scratch, I initialize with a pre-trained model Global Vectors for Word Representation (GloVe) which is available in [3]. The structure of embedding layer is shown in the figure below:

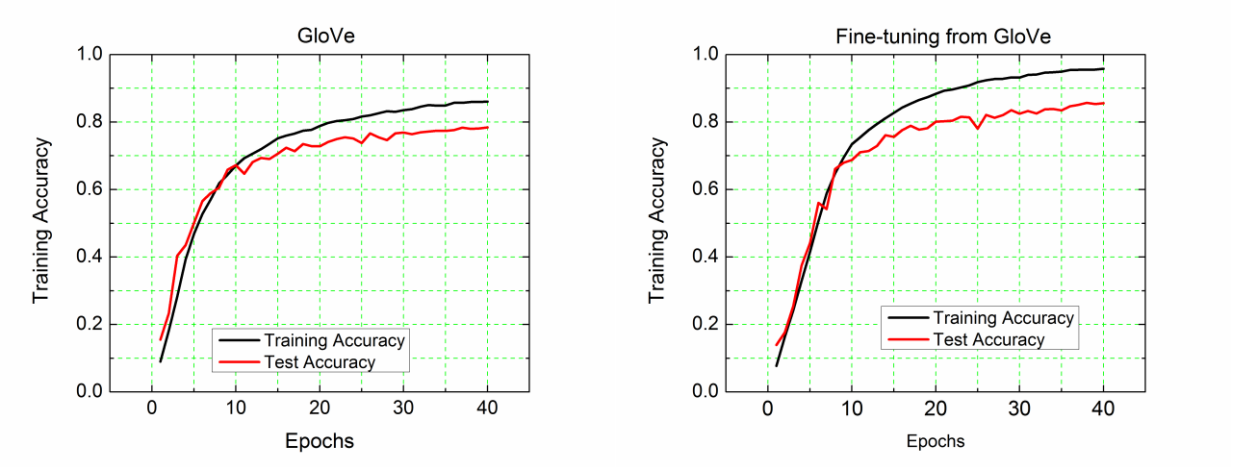



Reference about word-embedding for constructing this model can be found in [4-6]. Specifically in this implementation, the maximum number of words (with most frequent appearance) is set to be 20k and maximum sequence length is set to be 1000.

1. When using GloVe, a frozen layer (weight does not change during training) using the GloVe model is set in the CNN structure to implement word-embedding. However there are many people raise the question that their claim of >95% accuracy cannot be met after a few epochs [8-9]. And since my training set is smaller than the one used in [4], the accuracy is not expected to reach 95%.

2. In contrast, I also tried fine-tuning based on GloVe matrix weight. The only difference in the coding perspective is that I change the frozen layer in network to be trainable. So the weights in the embedding layer will also be trained in the training process.

The training process of approach 1 and 2 are displayed in the following figures:




However the best result from CNN does not win over SVM significantly, but the computational cost is much more expensive. After 40 epochs training in CNN, the performance is as blow:

$$F-1\ score = 0.8579$$

$$Test\ Accuracy = 0.8672$$

$$Training\ Accuracy = 0.9738$$

Further improvement can be achieved by increasing the maximum number of words and maximum length of sequences. However the computational cost will also increase correspondingly.

**Define, evaluate and compare the performance of the above models for doing this text classification.**

Unweighted average F-1 score has been used in the previous evaluation for each model. The definition of F-1 score for each class is:

$$F-1 = 2 \cdot \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

| Classifier | F-1 Score | Computational Time (s) |
|---|---|---|
| Multinomial Naive Bayes | 0.8318 | 0.20 |
| K-Nearest Neighbors | 0.7069 | 11.05 |
| Random Forest | 0.7771 | 74.55 |
| Support Vector Machine | 0.8458 | 8.90 |
| Convolutional Network | 0.8579 | ≈1 hour after 40 epochs |

**What if the text is unlabeled short sentence, would you still consider to use the above models and why?**

All the above models are supervised learning and labeled data are required so they do not applied any more. Since the data is unlabeled, clustering has to be performed beforehand. Cosine similarity will be a good standard for text clustering.

**What would be your solution to classify any new unlabeled text?**

Compare the performance and computational cost, SVM with linear kernel will be a good choice to classify new unlabeled text.

**Code**

'main_Q1.py': script that contains the four conventional machine learning model construction and prediction. Installation of scikit-learn package is required to run.

'main_Q2.py': script that contains the deep learning model. Installation of scikit-learn, TensorFlow and Keras is required to run.

'glove.6B.100d.txt': a pre-trained model for vector representation of words.

**Reference**

[1] Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pp.137-142.

[2] https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[3] https://nlp.stanford.edu/projects/glove/

[4] http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/

[5] https://github.com/dennybritz/cnn-text-classification-tf/issues/69

[6] http://adventuresinmachinelearning.com/word2vec-tutorial-tensorflow/

[7] https://github.com/fchollet/keras/issues/5826

[8] https://datascience.stackexchange.com/questions/17885/getting-low-accuracy-on-keras-pretrained-word-embeddings-example

[9] https://github.com/fchollet/keras/issues/5826