CS 760 project

Hanmo Li

May 4, 2018

Naive Bayes(NB)

Naive Bayes is a baseline algorithm to tackle the text classification and sentiment analysis problem. Its simplexity and robust makes it popular in the text processing domain. The critical function in Naive bayes is as follows.[1]

$$p(y|x_1, ..., x_n) = \frac{p(y)\prod_i p(x_i|y)}{p(x_1, ..., x_n)}$$

Problem here is to figure out the value of $p(x_i|y)$ from samples. There are two common ways to do this.

Gaussian Naive Bayes(GNB)

Under the Gaussian assumption, the $p(x_i|y)$ can be calculated as the equation follows.

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

The parameters σ_y and μ_y are estimated by the maximum likehood method.

Multinomial Naive Bayes(MNB)

This method is what we have learnt from class. That is, the $p(x_i|y)$ is calculated using the following equation.

$$p(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

α here is the smooth parameter.

TODO: compare the above two Naive Bayes algoritms.

Support Vector Machine(SVM)

SVM is another baseline algorithm for the sentiment analysis tasks. The tricky part for SVM is to choose a proper kernal. Here we tried several popular kernals and compared their model performances(shown in the following table).

TODO: compare the performances for different SVM kernals.

Naive Bayes - Support Vector Machine (NBSVM)

NB-SVM is a powerful algorithm introduced by Sida Wang and Christopher D. Manning(2012)[1], which is designed to tackle the sentiment and topic classification tasks. It's now one of the most popular algorithms used in Kaggle community for the text calssification competitions.

The intuitive insight for NB-SVM is that NB beats the SVM for short snippet sentiment task while SVM performs better for longer documents. Thus by combining these two baseline algorithms together, NB-SVM inherits both advantages for NB and SVM and outperforms most published algorithm in the sentiment analysis domain.

The process for NB-SVM is to firstly extract the NB features, then use them to fit the SVM model. The SVM model here can be replaced by other calssification methods, like logistic regression model etc..

The technical details about this algorithm is shown as follows[2](binary classification problem with $y = (1, -1)$).

Suppose $D = (d_1, d_2, ..., d_n)$ is a set of text documents and each text documents d_i contains a set of n-grams. We denote ngm $= (ngm_1, ngm_2, .., ngm_N)$ as the set of count vectors for all n-grams extracted from D. ngm_i^i represents the number of occurence of tth n-gram in document i. Defining $p = 1 + \sum_{i:y_i=1} ngm^i$ and $q = 1 + \sum_{i:y_i=-1} ngm^i$. To determine how important n-grams are for the classes y, defining the log ratio r:

$$r = log(\frac{p/||p||_1}{q/||q||_1})$$

Then to get better performance, we need to binarize ngm. That is, set ngm as $n\hat{g}m^i = 1(ngm^i > 0)$. To get the NB feature, we only need to product $\hat{r}$ and $n\hat{g}m$ together. Finally, by using the NB features as the input for SVM, we can get the final result of NB-SVM algorithm.

TODO: compare the performance of NBSVM with other algorithms.

references

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[1]. Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.

[2]. Lebret, Rémi, and Ronan Collobert. "N-gram-based low-dimensional representation for document classification." arXiv preprint arXiv:1412.6277 (2014).