CS 534 – Implementation Assignment 2

Due 11:59pm, October 18th, 2016

---

1 Group members

- Chunxiao Wang (Contribution: 50%)
- Miao Yang (Contribution: 50%)

2 Basic implementation

2.1 Algorithm

Suppose we have N tweets in the training set, and denote $y_n = \begin{cases} 1 \text{ if it comes from Donald Trump} \\ 0 \text{ if it comes from Hillary Clinton} \end{cases}$ for $n = 1, \ldots, N$. Denote N_0, N_1 to be the total number of tweets from Hillary and Trump, respectively. Let $\mathbf{x}^n$ be the document vector for n^{th} tweet. The MLE for $P(y=0)$ would be $\frac{N_0}{N}$.

In **Bernoulli model**, $\mathbf{x}^n$ will be a vector of 1 and 0, representing whether i^{th} word is in this document or not. The Laplace smoothing estimators are

$$\begin{aligned} P(x_i = 1 \,|\, y = 0) &= \frac{N_{i|0}+1}{N_0+2}, P(x_i = 0 \,|\, y = 0) = \frac{N_0 - N_{i|0}+1}{N_0+2} \\ P(x_i = 1 \,|\, y = 1) &= \frac{N_{i|1}+1}{N_1+2}, P(x_i = 0 \,|\, y = 1) = \frac{N_1 - N_{i|1}+1}{N_1+2} \end{aligned}$$

with $N_{i|0}$ being the number of Hillary's tweets that contain i^{th} word, and $N_{i|1}$ the number of Trump's tweets containing i^{th} word.

In **multinomial model**, $\mathbf{x}^n$ is a vector of integers representing the number of times the i^{th} word appears in n^{th} document. The Laplace smoothing estimators are

$$\begin{aligned} P(x_i \,|\, y = 1) &= \frac{\text{total numer of word } i \text{ in Trump's tweets } + 1}{\text{total number of words in Trump's tweets} + |V|}, \\ P(x_i \,|\, y = 0) &= \frac{\text{total numer of word } i \text{ in Hillary's tweets } + 1}{\text{total number of words in Hillary's tweets} + |V|} \end{aligned}$$

After the learning procedure, we can compute $P(y \,|\, \mathbf{x})$ and predict with $\arg\max_y P(y \,|\, \mathbf{x}) = \arg\max_y \log P(y \,|\, \mathbf{x})$. Note that by Bayes Rule, we have

$$\begin{aligned} \log P(y \,|\, \mathbf{x}) &= \log \frac{P(\mathbf{x} \,|\, y) P(y)}{P(\mathbf{x})} \\ &= \log P(\mathbf{x} \,|\, y) + \log P(y) - \log P(\mathbf{x}) \end{aligned}$$

Therefore $\widehat{y} = \arg\max_y \log P(y \,|\, \mathbf{x}) = \arg\max_y \{\log P(\mathbf{x} \,|\, y) + \log P(y)\}$.

2.2 Testing accuracy

Using Laplace smoothing method for both Bernoulli and multinomial models on the training and test data sets, we get the testing accuracy as follows:

Bernoulli	94.099%
multinomial	94.565%

Both models are doing a great job with testing accuracy $> 94\%$. Though multinomial model is slightly better than Bernoulli, their difference is very negligible.

2.3 Comparison of confusion

In the test data, there are 644 tweets, with 320 from Hillary and 324 from Trump. The confusion matrix for both models are given in Table 1, 2.

		Predicted	
		Hillary	Trump
Truth	Hillary	312	8
	Trump	30	294

Table 1: Confusion matrix from Bernoulli model

		Predicted	
		Hillary	Trump
Truth	Hillary	310	10
	Trump	25	299

Table 2: Confusion matrix from multinomial model

Both confusion matrices are diagonal dominated, meaning most predictions are correct. In terms of the mismatched ones, both models tend to predict Trump's tweets to Hillary's more often ($30/38$ in Bernoulli and $25/35$ in multinomial model). This means Trump's tweets are more confusing than Hillary's.

2.4 Top ten words

Top ten words with highest probability are given in Table 3.

	1	2	3	4	5	6	7	8	9	10
Hillary	.		to	the	”	a	and	of	Trump	for
Trump	.	!		the	”	to	and	I	in	:

Table 3: Top ten words from Hillary and Trump's selected tweets

3 Priors and overfitting

In multinomial model, denote $\theta_i = p(x_i), i = 1, \ldots, |V|$. Suppose we are now using a *Dirichelet* $(1+\alpha, \ldots, 1+\alpha)$ as the prior, of which the pdf is given as

$$p(\theta) = \frac{\Gamma(|V|\alpha)}{\{\Gamma(\alpha)\}^{|V|}} \left\{\prod_{k=1}^{|V|} \theta_i\right\}^{\alpha}$$

Then the posterior becomes

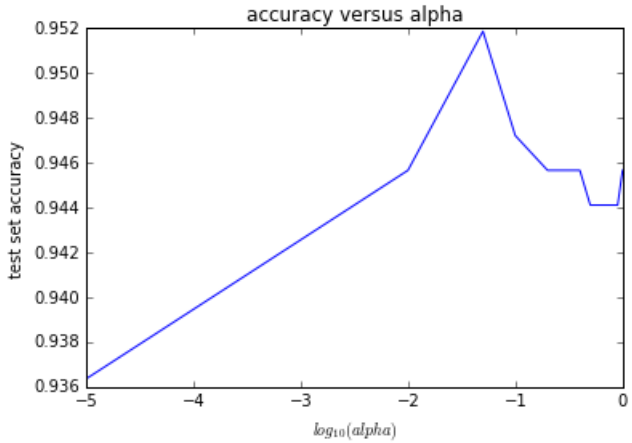
$$\begin{aligned} p(\theta|y=1) &\propto p(\theta)\, p(y=1|\mathbf{x}) \\ &\propto \prod_{k=1}^{|V|} \theta_i^{\alpha} \theta_i^{N_{i|1}-1} \\ &\propto \prod_{k=1}^{|V|} \theta_i^{\alpha+N_{i|1}-1} \\ &\backsim \textit{Dirichelet}\left(\alpha+N_{1|1}, \alpha+N_{2|1}, \ldots, \alpha+N_{|V||1}\right) \end{aligned}$$

Note that here $N_{i|1}, N_{i|0}$ denote the total number of word i in Trump's and Hillary's tweets, respectively. The posterior mode for $(\theta_i|y=1) = \dfrac{\alpha+N_{i|1}}{\sum_{i=1}^{|V|}(\alpha+N_{i|1})} = \dfrac{\alpha+N_{i|1}}{|V|\alpha+N_1}$. Similarly we have the posterior mode for $(\theta_i|y=0) = \dfrac{\alpha+N_{i|0}}{\sum_{i=1}^{|V|}(\alpha+N_{i|0})} = \dfrac{\alpha+N_{i|0}}{|V|\alpha+N_0}$. When $\alpha = 1$, we get the exact same result as we did in Laplace smoothing.

We choose different α, and repeat the multinomial model for each α. We choose a list of α to be

$$10^{-5}; 10^{-3}; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.$$

The figure of testing accuracy vs $\log_{10}\alpha$ is given below:



The trend from the figure can be described as follows: as α increases, the testing accuracy first increases and then decreases. We think this is because when α is small, the probability of new words tends to be negligible, then the accuracy becomes small. When α is large, we are pulling the probabilities of existing words to be smaller, thus decreases testing accuracy.

4 Identifying important features

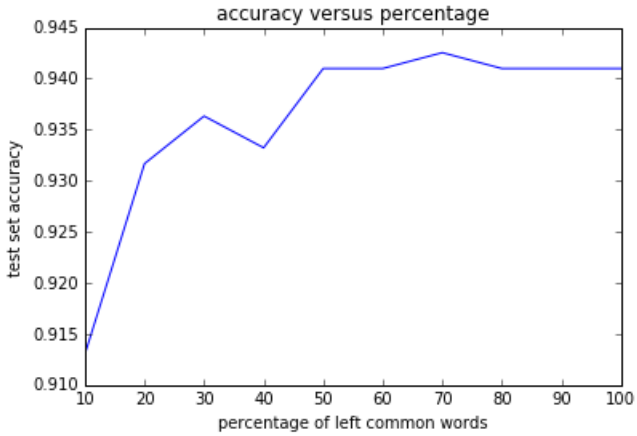
We put all the words that appeared in both Hilary's and Trump's tweets togethers, and there are more than $2,500$ words. We calculate the log ratios of their probabilities in two classes, i.e., $\log \frac{P(x_i|y=1)}{P(x_i|y=0)}$, and we get a distribution of the log ratios.

If the log ratio of a word is close to 0, i.e., the probabilities are quite the same in two classes, this word is really of no discriminative power, and we would like to remove these words.

Our strategy is to exclude certain percentage of words with log ratios close to 1, but not all of them. The following percentages are the percentages of word that we have KEPT in the file:

$$10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%.$$

We compare the testing accuracy (via multinomial model) after removing some words with no discriminative power. The result is shown in the figure below:



A general trend is that as we increase the percentage of such words that have been kept, the testing accuracy first increases and then decreases. This means there is a tradeoff between the percentage of words we keep and testing accuracy.

If we keep all these words, it would add a lot of confusion to our classification model, thus testing accuracy will be low. If we remove all the overlapping words, our vocabulary would become small; and when these words appear again in the test data set, the probability would tend to be smaller than it should be, making poor our prediction performance.

Therefore we should remove some, but not all the words of no discriminative power. From the figure, it seems that keeping 70% of these words would result in the best testing accuracy.