Tackling the Poor Assumptions of Naive Bayes Text Classifiers

DATA MINING [COURSE -MTL782]

PREPARED BY

Harsh Kumar *[2016MT10629]*

Parminder Singh *[2016MT10630]*

Anshuman Shrivastava *[2016MT10620]*

Introduction:

In this paper,we investigate the reasons behind Naive Bayes' poor performance. For each problem, we propose a simple heuristic solution. For example, we look at NaiveBayes as a linear classifier and find ways to improve the learned decision boundary weights. We also bettermatch the distribution of text with the distribution assumed by Naive Bayes. In doing so, we fix many of the classifier's problems without making it slower or significantly more difficult to implement.

Problem 1- [Naive Bayes selects poor weights]:

When one class has more training examples than an-other, Naive Bayes selects poor weights for the decision boundary. This is due to an under-studied bias effect that shrinks weights for classes with few training example.

To balance the amount of training examples used per estimate, we introduce a "complement class"formulation of Naive Bayes.

Problem 2- [Features Independence]:

As a result, even when words are dependent, each word con-tributes evidence individually. Thus the magnitude of the weights for classes with strong word dependencies is larger than for classes with weak word dependencies.To keep classes with more dependencies from dominating, we normalize the classification weights.

Problem 3- [multinomial Naive Bayes]:

In addition to systemic problems, multinomial Naive Bayes does not model text well. We present a simple transform that enables Naive Bayes to instead emulatea power law distribution that matches real term frequency distributions more closely.

Multinomial Naive Bayes:

Assumptions:

Multinomial Naive Bayes models the distribution of words in a document as a multinomial. A document is treated as a sequence of words and it is assumed that each word position is generated independently of every other.

For classification, we assume that there are a fixed number of classes, c∈ {1,2,...,m}, each with a fixed set of multinomial parameters. The parameter vector for a class c is $\theta c=\{\theta_{c1}, \theta_{c2},..., \theta_{cn}\}$, where n is the size of the vocabulary, $\sum_i \theta_{ci}= 1$ and θ_{ci} is the probability that word i occurs in that class. The likelihood of a document is a product of the parameters of the words that appear in the document,

$$p(d|\vec{\theta}_c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod_i (\theta_{ci})^{f_i},$$

By assigning a prior distribution over the set of classes, $p(\theta_c)$, we can arrive at the minimum-error classification rule (Duda & Hart, 1973) which selects the class with the largest posterior probability,

$$l(d) = \text{argmax}_c \left[\log p(\vec{\theta}_c) + \sum_i f_i \log \theta_{ci} \right] \qquad = \text{argmax}_c \left[b_c + \sum_i f_i w_{ci} \right],$$

Where b_c is the threshold term and w_{ci} is the class c weight for wordi. These values are natural parameters for the decision boundary.

Parameter Estimation:

Parameters must be estimated from the training data. We do this by selecting a Dirichlet prior and taking the expectation of the parameter with respect to the posterior. For details, we refer the reader to Section 2 of Heckerman(1995). This gives us a simple form for the estimate of the multinomial parameter,

$$\hat{\theta}_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha},$$

Substituting the true parameters in Equation 2 with our estimates, we get the MNB classifier,

$$l_{\text{MNB}}(d) = \text{argmax}_c \left[\log \hat{p}(\theta_c) + \sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} \right]$$

The prior class probabilities, $p(\theta c)$, could be estimated like the word estimates

Correcting Systemic Errors:

In this section, we show that skewed data—more training examples for one class than another—can cause the decision boundary weights to be biased. This causes the classifier to unwittingly prefer one class over the other. We show the reason for the bias and proposed to alleviate the problem by learning the weights for a class using all training data not in that class.

Class 1 $\theta = 0.25$	Class 2 $\theta = 0.2$	$p(\text{data})$	$\hat{\theta}_1$	$\hat{\theta}_2$	Label for H
T	TT	0.48	0	0	*none*
T	{HT,TH}	0.24	0	$\frac{1}{2}$	Class 2
T	HH	0.03	0	1	Class 2
H	TT	0.16	1	0	Class 1
H	{HT,TH}	0.08	1	$\frac{1}{2}$	Class 1
H	HH	0.01	1	1	*none*

$$p(\hat{\theta}_1 > \hat{\theta}_2) = 0.24$$
$$p(\hat{\theta}_2 > \hat{\theta}_1) = 0.27$$

Table: Shown is a simple classification example with two classes. Each class has a binomial distribution with probability of heads $\theta = 0.25$ and $\theta = 0.2$, respectively. Weare given one training sample for Class 1 and two train-ing samples for Class 2, and want to label a heads (H)occurrence. We find the maximum likelihood parameter settings ($\hat{\theta}$ 1and$\hat{\theta}$ 2) for all possible sets of training data and use these estimates to label the test example with the class that predicts the higher rate of occurrence for heads.Even though Class 1 has the higher rate of heads, the testexample is classified as Class 2 more often.

Table 1 gives a simple example of the bias. In the example, Class 1 has a higher rate of heads than Class 2.However, our classifier labels a heads occurrence asClass 2 more often than Class 1. This is not because Class 2 is more likely by default. Indeed, the classifier also labels a tails example as Class 1 more often,despite Class 1's lower rate of tails. Instead, the effect, which we call "skewed data bias," directly results from imbalanced training data. If we were to use the same number of training examples for each class, we would get the expected result—a heads example would be more often labeled by the class with the higher rate of heads.

Complement Naive Bayes:

In estimating weights for regular MNB we use training data from a single class c. In contrast, CNB estimates parameters using data from all classes except c. CNB's estimate is

$$\hat{\theta}_{\tilde{c}i} = \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha},$$

Effective because - each uses a more even amount of training dataper class, which will lessen the bias in the weight estimates.

NOTATION -

Nci ---> the number of times word i occurred in documents in classes other than c.

N˜c ---> the total number of word occurrences in classes other than c

αi & α ---> smoothing parameters,

The classification rule is

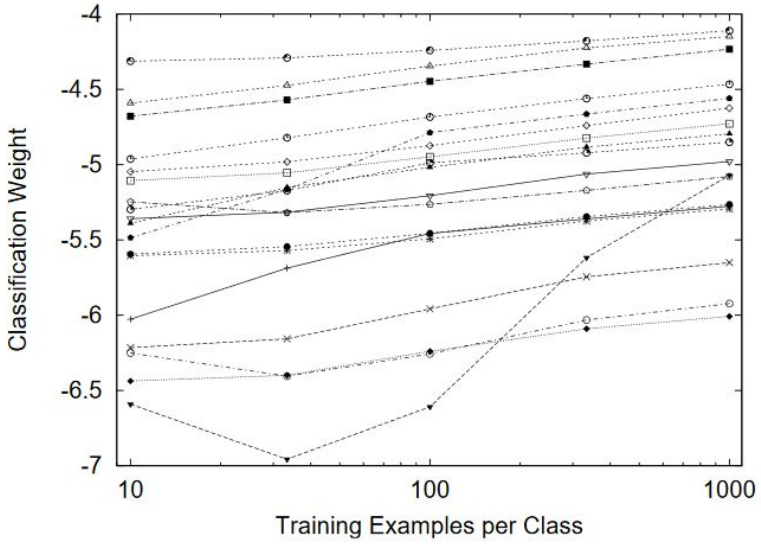
$$l_{\mathrm{CNB}}(d) = \mathrm{argmax}_c \left[\log p(\vec{\theta}_c) - \sum_i f_i \log \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha} \right]$$

The negative sign represents the fact that we want to assign to class c documents that poorly match the complement parameter estimates.

Analysis:

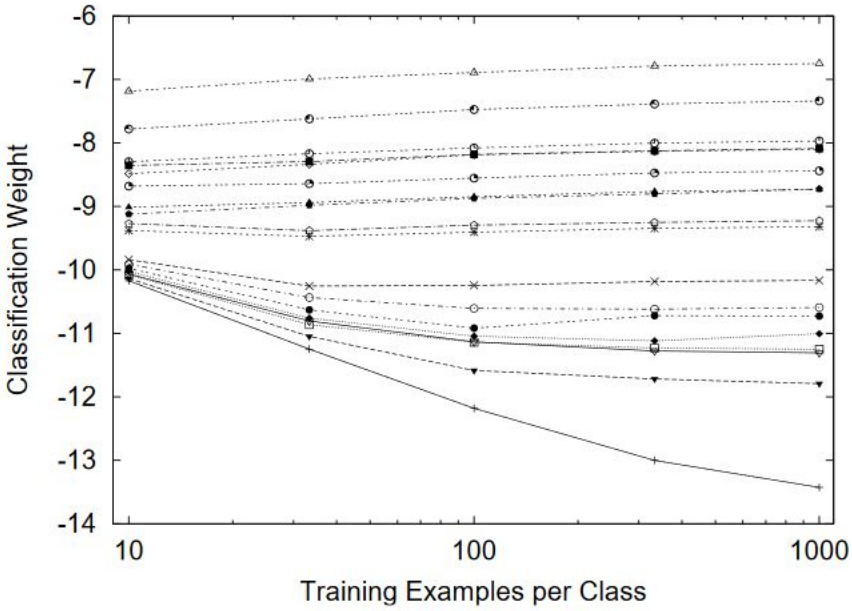
The regular NB weight estimates

In particular, the word that has the smallest weight for 10 through 100 examples moves up.



The complement weight estimates

The complement weights show the effects of smoothing, but do not show such a severe upward bias and retain their relative ordering.



Weight Magnitude Errors:

In this section, we discuss how the independence assumption can erroneously cause Naive Bayes To produce different magnitude classification weights.When the magnitude of Naive Bayes' weight vector~wcis larger in one class than the others, the larger-magnitude class may be preferred. For Naive Bayes,differences in weight magnitudes are not a deliberate attempt to create greater influence for one class. In-stead, the weight differences are partially an artifact of applying the independence assumption to dependent data. Naive Bayes gives more influence to classes that most violate the independence assumption. The following example illustrates this effect.Consider the problem of distinguishing between documents that discuss Boston and ones that discuss San Francisco.Let's assume that "Boston" appears inBoston documents about as often as "San Francisco"appears in San Francisco documents (as one might expect). Let's also assume that it's rare to see the words "San" and "Francisco" apart. Then, each timea test document has an occurrence of "San Francisco,"Multinomial Naive Bayes will double count—it willadd in the weight for "San" and the weight for "Fran-cisco." Since "San Francisco" and "Boston" occurs equally in their respective classes, a single occurrence of "San Francisco" will contribute twice the weight asan occurrence of "Boston." Hence, the summed contributions of the classification weights may be larger for one class than another—this will cause MNB to prefer one class incorrectly. For example, if a document has five occurrences of "Boston" and three of"San Francisco," MNB will label the document as "SanFrancisco" rather than "Boston."

We correct for the fact that some classes have greater dependencies by normalizing the weight vectors. Instead of assigning $w_{ci} = \log \theta_{ci}$, we assign,

$$\hat{w}_{ci} = \frac{\log \hat{\theta}_{ci}}{\sum_k |\log \hat{\theta}_{ck}|}.$$

We call this, combined with CNB, Weight-normalized Complement Naive Bayes (WCNB).

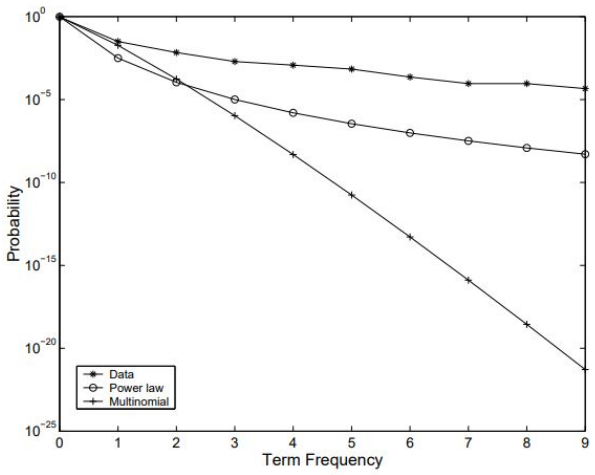
Modeling Text Better:

So far we have discussed systemic issues that arise when using any Naive Bayes classifier. MNB uses a multinomial to model text, which is not very accurate.In this section we look at three transforms to better align the model and the data. One transform affects frequencies—term frequency distributions have a much heavier tail than the multinomial model expects. We also transform based on document frequency, to keep common terms from dominating in classification, and based on length, to keep long documents from dominating during training. By transforming the data to be better suited for use with a multinomial model, we find significant improvement in performance over using MNB without the transforms.

Transforming Term Frequency:

Term frequency distributions have a much heavier tail than the multinomial model expects.

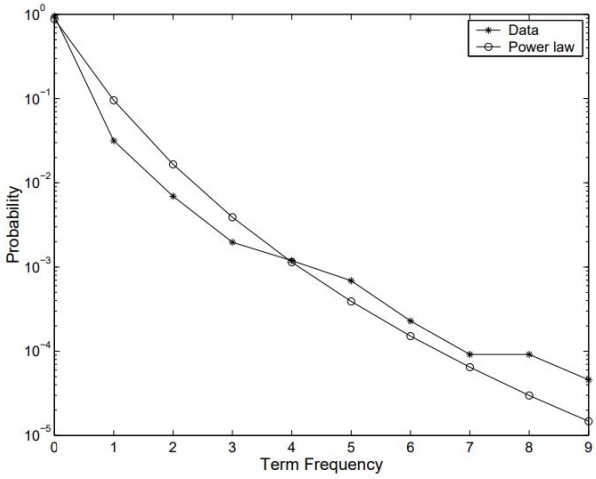
REASON- By transforming the data to be better suited for use with a multinomial model, we find significant improvement in performance over using MNB without the transforms



Thus power law distribution is used instead of direct multinomial distri. ,p(fi) ∝ (d + fi) log θ ,

where d has been chosen to closely match the text distribution.

The probability ∝ θ ^(log(d+fi)). So, we can use the multinomial model to generate probabilities proportional to a class of power law distri. via a simple transform, fi' = log(d + fi).



Transforming Document Frequency:

Common words are unlikely to be related to the class of a document, but random variations can create apparent fictitious correlations. This adds noise to the parameter estimates and hence the classification weights and can sway over a decision. For this reason, it is advantageous to downweight these words. A heuristic transform "inverse document frequency", is used:

$$f_i' = f_i \log \frac{\sum_j 1}{\sum_j \delta_{ij}},$$

where δ ij is 1 if word i occurs in document j, 0 otherwise.

Transforming Based on Length:

Documents have strong word inter-dependencies. Af-ter a word first appears in a document, it is more likely to appear again. Since MNB assumes occurrence in-dependence, long documents can negatively affect parameter estimates. We normalize word counts to avoid this problem.

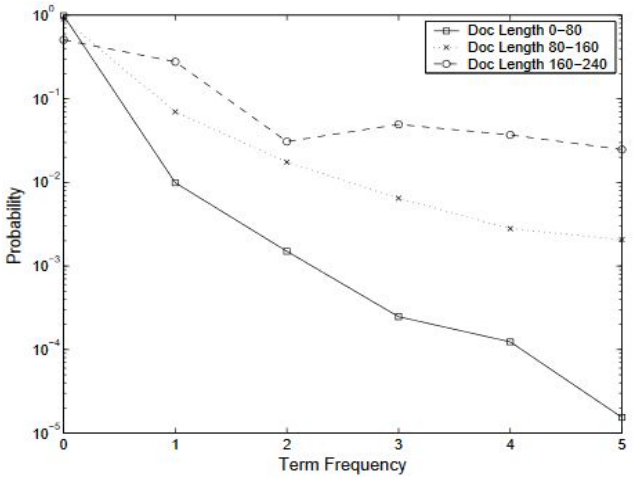


Figure shows empirical term frequency distributions for documents of different lengths. It's Not surprising that longer documents have larger probabilities for larger term frequencies, but the jump for larger term frequencies is disproportionally large. Documents in the 80-160 group are, on average, twice as long as those in the 0-80 group, yet the chance of a word occurring five times in the 80-160 group is larger than a word occurring twice in the 0-80 group. This Would not be the case if text were multinomial.

To deal with this, we again use a common IR transform that is not seen with Naive Bayes. We discount the influence of long documents by transforming the term frequencies according to

$$f_i' = \frac{f_i}{\sqrt{\sum_k (f_k)^2}}.$$

Our new Naive Bayes procedure:

Our new Naive Bayes procedure. Assignments are over all possible index values. Steps 1 through 3 distinguish TWCNB from WCNB.

- Let $\vec{d} = (\vec{d_1}, \ldots, \vec{d_n})$ be a set of documents; d_{ij} is the count of word i in document j.
- Let $\vec{y} = (y_1, \ldots, y_n)$ be the labels.
- TWCNB$(\vec{d}, \vec{y})$

1. $d_{ij} = \log(d_{ij} + 1)$
2. $d_{ij} = d_{ij} \log \frac{\sum_k 1}{\sum_k \delta_{ik}}$
3. $d_{ij} = \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}}$
4. $\hat{\theta}_{ci} = \frac{\sum_{j:y_j \neq c} d_{ij} + \alpha_i}{\sum_{j:y_j \neq c} \sum_k d_{kj} + \alpha}$
5. $w_{ci} = \log \theta_{ci}$
6. $w_{ci} = \frac{w_{ci}}{\sum_i w_{ci}}$

7. Let $t = (t_1, \ldots, t_n)$ be a test document; let t be the count of word i.
8. Label the document according to

$$l(t) = \arg\min_c \sum_i t_i w_{ci}$$

Results:

Dataset Used-20NEWSGROUPDATA (Mentioned in the paper) :

Link:http://archive.ics.uci.edu/ml/datasets/twenty+newsgroups

Analysis - Balanced Classes:
Total Number of Classes - 4

Accuracy = (Correct predictions/Total Examples)*100

Algorithm	Accuracy on Test Data
Naive bayes	93.8%
Complement Naive bayes	93.2%

Analysis - Balanced Classes:
Total Number of Classes - 20

Algorithm	Accuracy on Test Data
Naive bayes	78.71%
Complement Naive bayes	81.73%