Generalized Naive Bayes Classifiers

Kim Larsen
Concord, California
kim.larsen@sbcglobal.net

ABSTRACT

This paper presents a generalization of the Naive Bayes Classifier. The method is specifically designed for binary classification problems commonly found in credit scoring and marketing applications. The Generalized Naive Bayes Classifier turns out to be a powerful tool for both exploratory and predictive analysis. It can generate accurate predictions through a flexible, non-parametric fitting procedure, while being able to uncover hidden patterns in the data. In this paper, the Generalized Naive Bayes Classifier and the original Bayes Classifier will be demonstrated. Also, important ties to logistic regression, the Generalized Additive Model (GAM), and Weight Of Evidence will be discussed.

1. INTRODUCTION

The problem considered is that of predicting the class probability of a binary event (or target) given a set of independent variables (predictors). This problem arises frequently in various fields such as credit scoring, marketing and medical studies. One of the oldest and most simple techniques for binary classification is the Naive Bayes Classifier. Although simple in structure and based on unrealistic assumptions, Naive Bayes Classifiers (NBC) often outperform far more sophisticated techniques. In fact, due to its simple structure the Naive Bayes Classifier is an especially appealing choice when the set of independent variables is large. This is related to the classic bias-variance tradeoff; the NBC tends to produce biased estimated class probabilities, but it is able to save considerable variance around these estimates.

Despite these appealing features, the NBC is rarely used today in credit scoring or marketing applications. In fact, most popular statistical software packages do not have an NBC module. There are at least two reasons for this: First, the biased class probabilities can be a genuine problem for modeling applications where the sole focus is not classification or ranking. Second, the NBC is estimated under the assumption that predictors are conditionally independent given the target variable. As a result relationships between the dependent variable and the predictors are estimated in isolation without paying attention to covariance between the predictors. The NBC is therefore not able to approximate the multivariate regression function, and as a data exploration tool it adds no more information than a univariate analysis.

This paper presents a generalization of the Naive Bayes Classifier. The method is called the Generalized Naive Bayes Classifier (GNBC) and extends the NBC by relaxing the assumption of conditional independence between the predictors. This reduces the bias in the estimated class probabilities and improves the fit, while greatly enhancing the descriptive value of the model. The generalization is done in a fully non-parametric fashion without putting any restrictions on the relationship between the independent variables and the dependent variable. This makes the GNBC much more flexible than traditional linear and non-linear regression, and allows it to uncover hidden patterns in the data. Moreover, the GNBC retains the additive structure of the NBC which means that it does not suffer from the problems of dimensionality that are common with other non-parametric techniques.

Previous research has successfully tackled the severe assumptions of the Naive Bayes Classifier [4] [3]. However, most of these efforts deal with improving the Naive Bayes Classifier for text classification, i.e. the classification of some document into one of a set pre-defined categories given a vector of words and characters. The generalization presented in this paper is specifically designed for the type of binary classification problems that arise in credit scoring and marketing applications. In this type of setting, the vector of attributes is usually comprised of multiple continuous and discrete variables, and hence most text mining techniques do not apply. In fact, the method presented here is closely related to the Generalized additive model and various non-parametric regression techniques.

2. NAIVE BAYES CLASSIFIERS

Let Y be a binary random variable where

$$Y_i = \begin{cases} 1 & \text{if an event occured} \\ 0 & \text{otherwise} \end{cases} ,$$

and $X_1, \ldots, X_p$ be a set of predictor variables. If the predictors are conditionally independent given Y, the conditional joint probabilities can be written as

$$P(X_1, \ldots, X_p \mid Y) = \prod_{j=1}^{p} P(X_j \mid Y) .$$

Combining this with Bayes Theorem leads to the *Naive*

Bayes Classifier

$$\log \frac{P(Y=1 \mid X_1,\ldots,X_p)}{P(Y=0 \mid X_1,\ldots,X_p)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_{j=1}^{p} \log \frac{f(x_j \mid Y=1)}{f(x_j \mid Y=0)},$$

where $f(x_j \mid Y)$ is the conditional density of X_j.

The conditional densities can be estimated separately using non-parametric univariate density estimation. Joint density estimation is therefore avoided, which is especially desirable when the model contains a large number of predictors. More advanced non-parametric techniques of the form $y = f(x_1,\ldots,x_p)$, such as thin-plate splines and loess, perform poorly under such conditions. Rapidly increasing variance around the estimates as a result of sparseness of data is a typical problem when estimating the multidimensional function $f(x_1,\ldots,x_p)$, and CPU time is a problem with large data sets. Furthermore, given that densities can be estimated non-parametrically, the NBC can model the relationship between X_j and Y in a flexible and unrestricted fashion. In this aspect, the NBC is much more flexible than traditional linear and non-linear regression.

Since the conditional independence assumption is rarely true, the savings in variance and CPU time is not free. The estimated class probabilities tend to be biased and therefore the probabilities are typically not used directly but rather as rank-scores. This is a problem for certain modeling applications where ranking or classification is not the only purpose of the analysis. Furthermore, the descriptive value of the model is limited: Consider the ratio,

$$g_j(x_j) = \log \frac{f(x_j \mid Y=1)}{f(x_j \mid Y=0)},$$

which shows how the log-odds changes with X_j. This function is called the *Naive Effect* since covariance between X_j and other predictors is not taken into consideration. Obviously, as a data exploration tool, each Naive function adds nothing more than a univariate analysis.

3. GENERALIZED NAIVE BAYES CLASSIFIERS

The idea behind the Generalized Naive Bayes Classifier is to relax the conditional independence assumption by adjusting the Naive Effects. This is done by adding p functions, $b_1(x_1),\ldots,b_p(x_p)$, to the Naive Effects, where $b_j(x_j)$ accounts for the *marginal bias* attributed to $g_j(x_j)$. Hence the GNBC can be written as

$$\log \frac{P(Y=1 \mid X_1,\ldots,X_p)}{P(Y=0 \mid X_1,\ldots,X_p)} = \alpha + \sum_{j=1}^{p} \left(g_j(x_j) + b_j(x_j)\right),$$

where

$$\alpha = \log \frac{P(Y=1)}{P(Y=0)} ,$$

and $b_1(x_1),\ldots,b_p(x_p)$ are unspecified smooth functions. Obviously if the predictors are conditionally independent, $b_j(x_j) = 0$ for all j, and the model reduces to an NBC.

In short, the GNBC decomposes the total bias into p terms where the j'th term is a function of X_j. Treating the bias like this not only allows us to minimize the bias in the estimated probabilities, but also enhances the descriptive value of the model. The expression

$$\sum_{j=1}^{p} (g_j(x_j) + b_j(x_j))$$

is actually an additive approximation to the multivariate version

$$\log \frac{f(x_1,\ldots,x_p \mid Y=1)}{f(x_1,\ldots,x_p \mid Y=0)}$$

and thus $g_j(x_j) + b_j(x_j)$ can be interpreted as the marginal effect of X_j, fully adjusted for the effect of the other variables. The function $b_j(x_j)$ reflects the marginal bias, i.e. how the effect of X_j changes in the presence of the other predictors.

The decomposition of the bias also allows the GNBC to preserve the additive property of the NBC and hence avoid joint density estimation. Sparseness of data and inflated variance is therefore not an issue which makes the GNBC a practical choice for most binary modeling applications.

4. ESTIMATING THE GENERALIZED NAIVE BAYES CLASSIFIER

Fitting a Generalized Naive Bayes Classifier is a two-stage process. In the first stage, the Naive Effects are estimated using univariate density estimation. In the second stage the adjustment functions are estimated by iteratively smoothing the partial residuals against the predictors.

4.1 Estimation of Naive Effects

The Naive Effects can be estimated separately using univariate kernel density estimates. Let $N_{ij}(x_{ij})$ denote the symmetric nearest neighborhood around x_{ij}, which contains the k observations to the left of x_{ij} and the k observations to the right of x_{ij}. The kernel density estimate for X_{ij} has the form

$$\hat{f}_{ij}(x_{ij}) = \frac{1}{nh} \sum_{k=1}^{n} I(x_{ij}, x_{kl}) K_\lambda \left(\frac{|x_{ij} - x_{kj}|}{h} \right)$$

where h is the width of the neighborhood and $I(x_{ij}, x_{kl})$ is an indicator function

$$I(x_{ij}, x_{kl}) = \begin{cases} 1 & \text{if } x_{kl} \in N_k(x_{ij}) \\ 0 & \text{otherwise} \end{cases} .$$

The purpose of the weight function K_λ is to reduce the weight given to observations within $N_k(x_{ij})$ that are far away from x_{ij}. This reduces jaggedness in the curves and makes estimates more robust when dealing with large numbers of ties in the predictor variable. Popular choices for K_λ include the minimum variance kernel

$$K_\lambda(t) = \begin{cases} \frac{3}{8}(3 - 5t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases} ,$$

and the Epanechnikov kernel

$$K_\lambda(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases} .$$

The estimated conditional densities can then be computed as

$$\hat{f}(x_{ij} \mid Y=1) = \frac{\sum_{k=1}^{n} I(x_{ij}, x_{kl}) K_\lambda\left(\frac{|x_{ij}-x_{kj}|}{h}\right) y_k}{h\sum_{i=1}^{n} y_k}$$
$$\hat{f}(x_{ij} \mid Y=0) = \frac{\sum_{k=1}^{n} I(x_{ij}, x_{kl}) K_\lambda\left(\frac{|x_{ij}-x_{kj}|}{h}\right) (1-y_k)}{h\sum_{i=1}^{n} (1-y_k)},$$

which leads to the estimator of for the Naive Effects

$$\hat{g}_{ij}(x_{ij}) = \log\frac{\hat{f}(x_{ij} \mid Y=1)}{\hat{f}(x_{ij} \mid Y=0)}.$$

If X_j is discrete, simple histogram estimators can be used

$$\hat{g}_{ij}(x_{ij}) = \log\frac{P(X_{ij}=x_{ij} \mid Y=1)}{P(X_{ij}=x_{ij} \mid Y=0)},$$

which provides a seamless way to mix discrete and continuous variables. In the discrete case, the Naive Effects are popularly known as the *weight-of-evidence*.

The smoothness of the estimated densities is controlled by the smoothing parameter. As with any smoother there is a bias-variance tradeoff when selecting λ. A large λ will produce a smooth curve with low variance and potentially high bias. Conversely, a small λ will produce a flexible curve with potentially high variance and little bias. The former curve might be too smooth and miss important patterns in the data, while the latter curve might be too jagged and not generalize well to testing data. Although there are iterative methods to find the optimal smoothing parameter, a good choice can usually be found by combining some simple rules with cross-validation results. If an estimated curve fits the training data well but fits the testing data poorly, increasing λ will most likely improve the validation results. Likewise, If n or the number of events is small, a large λ may be necessary to avoid sparseness of data.

4.2 Estimation of Marginal Bias

The marginal biases are estimated through an iterative backfitting algorithm similar to the procedure used with a logistic Generalized Additive Model [1]. It uses the NBC as a starting point and then, for each predictor in turn, estimates the marginal bias using the most current estimates for the other predictors. This step is repeated until the estimates $\hat{b}_1(x_i), \ldots, \hat{b}_p(x_p)$ stabilize. Note that the estimates $\hat{g}_1(x_1), \ldots, \hat{g}_p(x_p)$ never change throughout this process. This reiterates that the goal of the GNBC is simply to extend the NBC by correcting for the bias caused by the conditional independence assumption.

Let $\hat{\theta}_i$ denote the estimated log odds for observation i, i.e.

$$\hat{\theta}_i = \alpha + \sum_{j=1}^{p}\left(\hat{g}_j(x_j) + \hat{b}_j(x_j)\right),$$

and $\hat{\pi}_i$ denote the estimated target probability

$$\hat{\pi}_i = \frac{e^{\hat{\theta}_i}}{1+e^{\hat{\theta}_i}}.$$

The odds ratio at y_i around the current estimate $\hat{\pi}_i$ can be approximated by the first order Taylor series

$$\log\frac{P(Y_i=1)}{P(Y_i=0)} \approx \hat{\theta}_i + \frac{y_i - \hat{\pi}_{ij}}{\hat{\pi}_{ij}(1-\hat{\pi}_{ij})}.$$

This leads to the *partial residual* for variable X_j

$$e_{ij} = \log\frac{P(Y_i=1)}{P(Y_i=0)} - (\hat{\theta}_i - \hat{b}_j(x_j)) = \hat{b}_j(x_j) + \frac{y_i - \hat{\pi}_{ij}}{\hat{\pi}_{ij}(1-\hat{\pi}_{ij})},$$

which isolates the bias caused by X_j by removing the effect of all other variables. Hence an estimate for $b_j(x_j)$ can be obtained by smoothing e_j against X_j with weights

$$w_{ij} = \mathrm{var}\left(\theta_i + \frac{y_i - \hat{\pi}_i}{\hat{\pi}_i(1-\hat{\pi}_i)}\right) = \frac{1}{\hat{\pi}_i(1-\hat{\pi}_i)},$$

using the most current estimate of π_i. Because π_i and w_i are functions of $\hat{b}_j(x_j)$, the dependent variable changes when $\hat{b}_j(x_j)$ is updated and the smoothing itself must be done iteratively. Hence the marginal bias for x_j, given the most current estimate of π_i, is estimated by repeatedly smoothing e_j against x_j, updating the weights at each iteration.

In order to estimate $b_1(x_i), \ldots, b_p(x_p)$ in a simultaneous fashion, a *backfitting* algorithm is used in conjunction with the iterative smoothing described above. The backfitting algorithm works by cycling through the predictors and fitting smoothing functions to the partial residuals, using the most current estimates for the other smoothing functions. Hence we can fit the GNBC through a nested do-loop: The outer loop controls the cycles of the backfitting algorithm. Each time the backfitting algorithm has cycled through all the predictors, the outer loop evaluates a goodness of fit criterion as well as the convergence status of the smoothing functions. If the smoothing functions cease to change or the goodness of fit criterion ceases to improve, the outer loop terminates. The inner loop is simply the backfitting algorithm itself, and within the backfitting algorithm is a do-until loop that controls the iterative smoothing described above.

This is the same type of maximizer used with GAM, and it is also referred to as the *Local Scoring Algorithm* [1]. Furthermore, it can be shown that this routine is a special case of Iteratively Re-weighted Least Squares (IRLS) where the log-likelihood function

$$L\left(b_1(x_1), \ldots, b_p(x_p)\right) = \sum_{i=1}^{n}\left(y_i \log\hat{\pi}_i + (1-y_i)\log(1-\hat{\pi}_i)\right)$$

is being maximized. The difference between this procedure and the typical IRLS is that the weighted linear regression has been replaced by the backfitting algorithm in the outer loop. The procedure has several attractive properties: First, assuming that $y_1, \ldots, y_n$ are i.i.d., the log-likelihood function is an obvious fitting criterion for the GNBC. Second, the log-likelihood for this problem is concave which means that convergence is almost always guaranteed.

The algorithm is described in more detail in the following: Let $\mathcal{S}_j\left(h(x_j)\right)$ denote the weighted smooth of some function $h(x_j)$ against X_j using the weights given above. The choice of smoother is not critical, but for consistency the smoother should be based on the same neighborhoods used in the estimation of the Naive Effects. This is because the purpose of this stage is to remove bias caused by the conditional independence assumption, not bias caused by smoothing. Given a smoother, we can write the algorithm as:

Initialize: $\hat{b}_j^0(x_j) = 0$, for all j
Outer Loop: $k = 1, 2, \ldots$
Inner Loop: $j = 1, \ldots, p, j = 1, \ldots, p, \ldots$

$$\hat{b}_j^k(x_j) = \mathcal{S}_j \left(\hat{b}_j^{k-1}(x_j) + \frac{y_i - \hat{\pi}_{ij}}{\hat{\pi}_{ij}(1 - \hat{\pi}_{ij})} \right),$$

where $\mathcal{S}_j$ has to be applied iteratively until $\hat{b}_j^k(x_j)$ stabilizes. If X_j is a categorical variable, a simple weighted average of the partial residuals for each class is used instead of $\mathcal{S}_j$. The outer loop is repeated until the ratio

$$\frac{\sum_{j=1}^p \sqrt{\sum_{i=1}^n \left(\hat{b}_j^k(x_j) - \hat{b}_j^{k-1}(x_j) \right)^2}}{\sum_{j=1}^p \sqrt{\sum_{i=1}^n \hat{b}_j^k(x_j)^2}}$$

is less than some specified threshold, or the likelihood function ceases to improve. The initial estimate of the intercept is

$$\hat{\alpha}^0 = \log \frac{\sum_{i=1}^n y_i}{n - \sum_{i=1}^n y_i}.$$

At the end of each iteration, the intercept is adjusted to ensure that the expected number of events equals the actual, i.e.

$$\sum_{i=1}^n \hat{\pi}_i = \sum_{i=1}^n y_i,$$

which is identical to the way the intercept is estimated in a logistic regression model.

4.3 Building a Classifier with the GNBC

4.3.1 Variable Selection Strategy for the GNBC

The GNBC is not a practical algorithm for variable selection when the number of variables is large. Although the GNBC is computationally efficient compared to other non-parametric techniques it still requires much more CPU time than LDA or logistic regression. Therefore, another technique is recommended for large variable selection tasks. One technique that has proven to be very effective for such problems is a combination of the standard Naive Bayes Classifier and logistic regression. First, Naive Effect functions are estimated for all the candidate variables. Second, a logistic regression containing all the Naive Effects is fitted, and a stepwise selection is performed to pick the most significant Naive Effects. Once a manageable set of variables have been determined, the GNBC is used to fit the final model. This technique should be preferred over variable selection from linear effects, as the Naive Effects ensure that variables with strong non-linear effects will be picked up during the selection process. Once the marginal bias functions have been estimated by the GNBC, we are often able to reduce the set of variables even further by identifying variables that are highly affected by multicollinearity. In such cases, the marginal bias functions can neutralize the Naive Effects, which is usually a sign that the variables are abundant.

The %GNBC SAS macro allows for estimation of both Naive and Generalized Naive Bayes Classifiers, and has a function to output Naive Effects and Marginal bias functions. Hence it can be used in conjunction with PROC LOGISTIC for large variable selection problems.

4.3.2 Implementation

In most cases the ultimate goal is to build a classification system that can score all prospects or existing customers, in order to evaluate their credit risk or propensity to purchase a certain product. This requires the ability to score records that are not in the training data. Implementing the GNBC as a classification system is easy, although more complicated than implementing a simple LDA model or logistic regression model. For every variable, a look-up table linking X_j to $b_j(x_j) + g_j(x_j)$ must be created. Furthermore, we expect to observe values that are not present in the training data which means that the system should be able to extrapolate from the loop-up tables. If we let $x_{(1)j}, \ldots, x_{(n)j}$ denote the ordered values of X_j in the training data, the system should map values in the range $(x_{(i)j}, x_{(i+1)j}]$ to $b_j(x_{(i+1)j}) + g_j(x_{(i+1)j})$. Values that exceed boundaries of the training data are mapped to $b_j(x_{(1)j}) + g_j(x_{(1)j})$ or $b_j(x_{(n)j}) + g_j(x_{(n)j})$.

This implementation algorithm can be performed with the %SCOREGNBC SAS macro.

4.4 Spam data example

The data set used in this example is the same data used to demonstrate GAM in the book "Elements of Statistical Learning" by Hastie and Tibshirani [2]. The data was donated by George Forman of Hewlett-Packard laboratories, Palo Alto, California, and is publicly available at the site `ftp.ics.uci.edu`. It contains various information on 4601 email messages, where 1813 are categorized as spam and 2788 are legitimate emails. The goal of this analysis is to build a filter that prevents spam from entering the inbox. One strategy is to estimate the probability of spam for every inbound email using a binary regression model. If the estimated probability exceeds a certain level, the email is classified as spam and sent to a junk mail folder.

The data set contains 57 continuous ordinal predictors as described below:

- `wordFreq[WORD]` 48 continuous attributes that contain the percentage of words in the e-mail that match a given word. These attributes indicate whether a specific word occurs frequently in the email. Examples include `business`, `address`, and `free`
- `charFreq[CHARACTER]` 6 continuous attributes that contain the percentage of characters in the e-mail that match a given character. These attributes indicate whether a specific character occurs frequently in the email. Examples include `!`, `#` and `$`
- `capLengthAVE` A continuous attribute that contains the average length of uninterrupted sequences of capital letters
- `capLengthMAX` An integer attribute that contains the longest uninterrupted sequences of capital letters
- `capLengthTOT` An integer attribute that contains the sum of uninterrupted sequences of capital letters

1536 records were randomly selected for testing, and 3065 were allocated to the training data. The model was then built in two steps: In the first step, a simple NBC containing all 57 predictors was fitted, followed by a stepwise selection of Naive Effects. The following variables were selected:

```
wordFreqGEORGE
wordFreqOUR
wordFreqOVER
wordFreqREMOVE
wordFReqINTERNET
wordFReqREPORT
wordFReqFREE
wordFReqBUSINESS
wordFReqCREDIT
wordFReqMONEY
wordFReq1999
wordFReqEDU
wordFReqEXCL
wordFReqDOLLAR
wordFreqHP
wordFreqPROJECT
capLengthMAX
capLengthAVE
charFreqDOLLAR
charFreqEXCLAMATION
```

In the second step, the final model was obtained by fitting a GNBC containing the selected predictors. A smoothing parameter of $\lambda = 0.6$ was chosen for all the variables, and the minimum variance kernel was chosen for density estimation. With the convergence criterion set at 0.001, the %GNBC macro converged in 6 iterations, which took roughly 1 minute of real computing time. The GNBC requires roughly $O(n)$ operations, so even with a large dataset computation time is manageable. The model fit statistics are summarized below

Stage	-2LogL	MSE	C
NBC	1303.78	573.137	0.976
GNBC	1082.36	5.10	0.979

It appears that the GNBC does remove a substantial amount of bias. Note that the GNBC is primarily a bias reduction technique which explains why the improvement is mostly reflected in the log-likelihood and MSE, and to a lesser extent in the C-statistic.

The spam cut-off can be derived using Bayes rule. Let L_1 denote the loss associated with classifying legitimate emails as spam, and L_2 denote the loss associated with classifying spam as legitimate email. With an equal loss assumption, the cut-off is then given by

$$\frac{L_1}{L_1 + L_2} = 0.5.$$

In reality, it is more severe to classify legitimate email as spam and hence L_1 should be greater than L_2, but for this example we assume equal loss. Using the %SCOREGNBC macro to score the testing data the following confusion matrix was generated

	spam	email
Predicted spam	59.58%	3.62%
Predicted email	1.65%	35.15%

which shows an overall misclassification rate of 5.3%.

The spam data illustrates how much information and predictive strength can be lost when assuming linear effects. The spam data is typical in the sense that the correlation between the dependent variable and the independent variables is not linear. Two patterns that are common in the spam data, and in most other real-life modeling problems, is the V-shaped and the "hockeystick-shaped" correlation. The hockeystick occurs when the log odds increase or decrease rapidly as the variable increases from the minimum, but then does not change much after that. Both patterns are hard to model with parametric functions, and poorly captured with a linear effect. Moreover, in order to attempt approximating these patterns with parametric functions, one would have to know that these patterns exist prior to fitting the model. With the GNBC, no knowledge about the correlation between X_j and Y is required prior to fitting the model, and because of the non-parametric density estimation, it can adapt to correlation of any shape. The table below demonstrates how much fit would have been lost if we had used a standard linear technique.

Technique	C	Testing Error
NBC	0.976	6.1%
GNBC	0.979	5.3%
Logistic regression	0.967	8.1%
Fisher LDA	0.961	9.3%

Examples of hockeystick shaped correlation and V-shaped correlation are given in figures 1 & 2. The graphs show the bias adjustment function, the Naive Effect, as well as the final estimate $b_j(x_j) + g_j(x_j)$. Figure 1 reveals that `capLengthMax` has the hockeystick effect, whereas the variable `wordFreqOur`, shown in figure 2, has more of a U-shaped effect. Both graphs indicate that most of the bias adjustment is needed when the naive function peaks. This suggests that, had we just fitted an NBC, we would have overestimated the target probability in certain ranges.

5. CONCLUSION

In this paper, we have discussed the Generalized Naive Bayes Classifier. The GNBC is a flexible algorithm for predicting probabilities of a binary target given a set of independent variables. It can fit a large model without any prior knowledge about the independent variables, which enables us to uncover hidden patterns in the data and spend more time selecting variables and designing the model. This makes the GNBC a powerful tool for prediction as well as data exploration.

Finally, it is clear that there is a strong link between the logistic GAM and the GNBC. In fact, the GNBC can be viewed as a two-stage method that combines GAM and NBC; the first stage fits and NBC to the data, and the second stage uses GAM to minimize the bias. This turns out to be a powerful combination. The NBC is fast regardless of the size of the model, and we typically only need a couple of additional GAM iterations to adjust the bias and find the optimum. In addition, the two-stage process allows us to measure the severity of the conditional independence assumption, as well as the predictive strength gained from removing the bias.

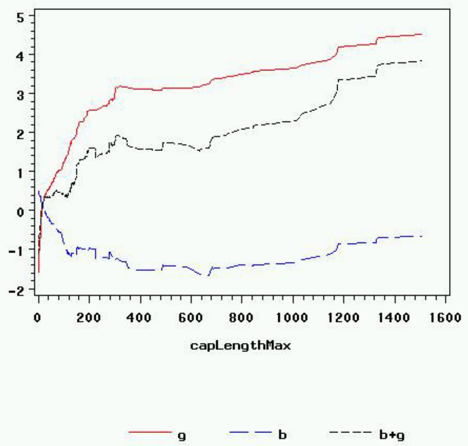

Figure 1: Estimated effects for `capLengthMax`

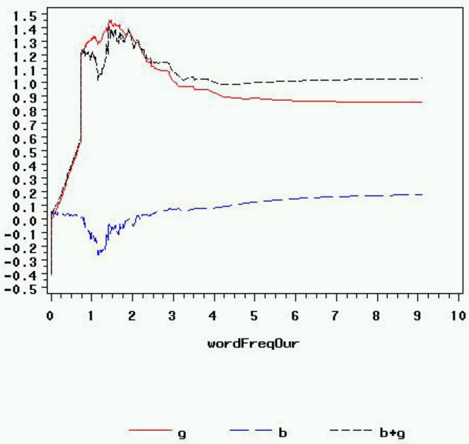

Figure 2: Estimated effects for `wordFreqOur`

6. REFERENCES

[1] T. Hastie and R. Tibshirani. *Generalized Additive Models.* Monographs on Statistics and Applied Probability 43. Chapmann and Hall, 1990.

[2] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning.* Springer, 2001.

[3] F. Peng, D. Schuurmans, and S. Wang. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval, 7, 317-345, 2004.*

[4] J. D. M. Rennie, L. Shih, J. Teewan, and D. R. Karger. Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.*