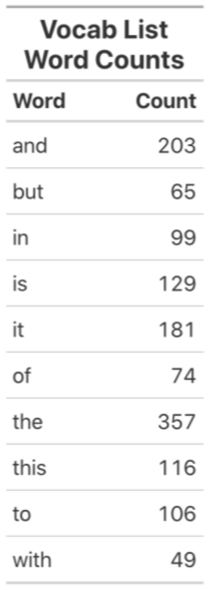
Ross Brancati
October 6, 2021
CS 589 – Machine Learning
Homework 2 Report

1. Naïve Bayes and Bag of Words

1.

Table 1: Vocabulary List Generate from Bag of Words Model with Word Counts



Vocab List Word Counts

Word	Count
and	203
but	65
in	99
is	129
it	181
of	74
the	357
this	116
to	106
with	49

2.

$X = (X_1, X_2, X_3, \dots, X_D)$, where D is the number of words in the vocabulary list

N = number of data entries $\rightarrow N = [1, \dots, i]$

$Y = \{0, 1\}$

For one particular example in the data entries:

$$P(Y=1|X) = \frac{P(X|Y=1)P(Y=1)}{P(X)}$$

$$P(Y=1|X_1, X_2, \dots, X_D) = \prod_{i=1}^{D} P(Y=1|X_i)$$

$$= \prod_{i=1}^{D} \frac{P(X_i|Y=1)P(Y=1)}{P(X)}$$

$$= P(Y=1) \prod_{i=1}^{D} \frac{P(X_i|Y=1)}{P(X)}$$

↳ however, we may have a word in our vocabulary list, $X = (X_1, \dots X_D)$ that does not exist in the data entry, which would cause $P(X|Y=1) = 0$. When we go to multiply over $i = 1, \dots, D$, we will get 0 for $P(Y=1|X)$. We also must adjust the prior

$$\hat{P}(X_d = x_d^v | Y=c) = \frac{\#\{x_d = x_d^v \wedge Y=c\} + (\beta_c - 1)}{\#\{Y=c\} + \sum_{v \in \text{unique}(x_d)} (\beta_c - 1)}$$

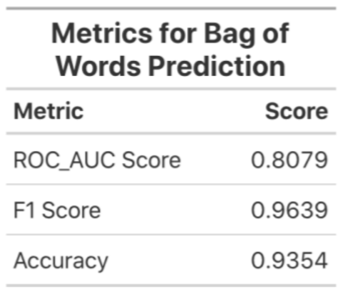
$$P(X_i|Y=1) = \frac{\#\{X_i = x_i^v \wedge Y=c\} + (\beta_c - 1)}{\#\{Y=c\} + \sum_{v \in \text{unique}(x_d)} (\beta_c - 1)}$$

↳ If you know that the prior cannot have $P(Y=c) = 0$ because there are no samples in the training data where a class is not represented, $P(Y=1|X)$ can be written as:

$$P(Y=1|X) = \left(\frac{\#\{Y=c\}}{\text{total } \#}\right) \prod_{i=1}^{D} \left(\frac{\#\{X_i = x_i^v \wedge Y=c\} + (\beta_c - 1)}{\#\{Y=c\} + \sum_{v \in \text{unique}(x_d)} (\beta_c - 1)}\right)$$

3.

Table 2: Evaluation metrics for Naïve Bayes bag of words prediction model with hyperparameter beta = 1



Metrics for Bag of Words Prediction

Metric	Score
ROC_AUC Score	0.8079
F1 Score	0.9639
Accuracy	0.9354

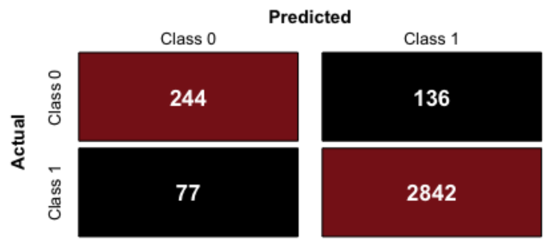


Figure 1: Confusion matrix for Naïve Bayes bag of words prediction model with hyperparameter beta = 1.

4.

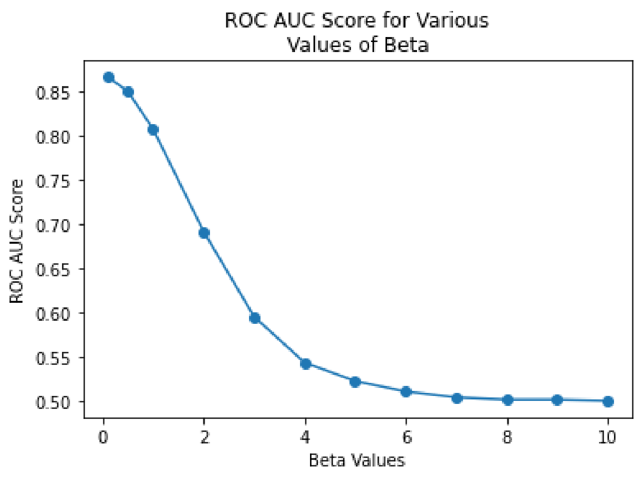


Figure 2: Plot of the ROC_AUC score for different values of the hyperparameter beta.

The ROC_AUC score is an important metric used in machine learning to evaluate the performance of a model. The receiver operating characteristics curve (ROC) plots the true positive rate (or recall) on the y-axis and the false positive rate on the x-axis. As a reminder, recall represents the number of positive class predictions made from all positive examples. So, the ROC curve gives you a sense of how many positive examples in the class were classified correctly and incorrectly. In the plot above, we see that the ROC_AUC score decreases as the values of beta increase. The beta hyperparameter is included in the probability calculations to prevent a probability of a certain feature being zero, which would cause the product of marginal class conditional distributions in *P(Y=1|X)* to be zero, which we want to avoid because missing one or a few features should not push this probability to be zero. It should be noted that higher ROC_AUC scores represent the model's ability to predict each class correctly. Optimal values of beta for this model are low (i.e., 0.5 or 1) because multiplying the denominator by a larger value of beta will decrease the probability of a word being in a class. As that denominator increases, the probability decreases and will inherently not classify as accurately. Thus, it is important to ensure that beta is not too large to avoid this from happening but is important because we do not want to have a probability of one word being equal to 0 when the data is sparse.

2. Probabilistic Classification

1.

$$Bayes\ Rule: P(Y = c|\boldsymbol{X} = \boldsymbol{x}) = \frac{\phi_c(\boldsymbol{x})\pi_c}{\sum_{c' \in \gamma} \phi_{c'}(\boldsymbol{x})\pi_{c'}}$$
$$Bayes\ Optimal\ Classifier: f_B(\boldsymbol{x}) = argmax_{c \in \gamma} P(Y = c|\boldsymbol{X} = \boldsymbol{x})$$

Assuming we have conditional independence between features, the Bayes Optimal Classifier can be written as:

$$Y^{new} = argmax_{c \in \gamma} P(Y = c) \Pi_{d=1}^{D} P(X_i^{new}|Y = c)$$

And we know that the probability of a datapoint being of class Y is:

$$P(X_i, X_{i+1}|Y) = P(X_i|X_{i+1}, Y)P(X_{i+1}|Y)$$

In this case, we have conditional independence between every other feature (i.e. feature 1 and 2 are correlated, but features 1 and 3 are conditionally independent). Substituting this into the above Bayes Optimal Classifier:

$$Y^{new} = argmax_{c \in \gamma} P(Y = c) \Pi_{d=1}^{D} P(X_i|X_{i+1}, Y)P(X_{i+1}|Y)$$

Which can also be written as:

$$f_{NB}(\boldsymbol{x}) = argmax_{c \in \gamma} \pi_c \Pi_{d=1}^{D} P(X_i|X_{i+1}, Y)P(X_{i+1}|Y)$$

2. Given that each pair of correlated features is a bivariate Gaussian distribution, each pair of correlated (X_i, X_{i+1}) features requires estimates of

$$\mu = [\mu_0, \mu_1] \text{ and a covariance matrix } \Sigma = \begin{pmatrix} var(x_1) & cov(x_1, x_2) \\ cov(x_1, x_2) & var(x_2) \end{pmatrix}$$

If we have D total features in the dataset, there will be D/2 pairs of features, and each pair of features will have 5 total parameters associated with it. Plus, we have an additional parameter for the prior, so there will be a total of $\frac{D*5}{2} + 1$ features.

3.

$$\log P(X|Y=c) = \Sigma_i \log P(X^{(i)}|Y=c)$$

$$\log P(X^{(i)}|Y=c) = \log P(X_1^{(i)}, X_2^{(i)}|Y=c) + \log P(X_3^{(i)}, X_4^{(i)}|Y=c)$$

$$+ \dots + D$$

↳ where D in the above expression is equal to the number of pairs of features (X_d, X_{d+1})

- Take the derivative with respect to the hyper-parameters $[\mu_i, \mu_{i+1}]$

$$\frac{\partial}{\partial[\mu_1, \mu_2]} = \Sigma_i \log P(X_1^{(i)}, X_2^{(i)}|Y=c) = 0$$

- We could take the derivative with respect to $[\mu_1, \mu_2]$, or we could use sample estimates:
  - Sample mean = $\bar{X} = \frac{1}{N}\sum_{n=1}^{N} X_i$
  - Sample covariance = $Q = \frac{1}{N}(F-\bar{X})(F-\bar{X})^T$

    where $F = [X_1 \; X_2 \; \dots \; X_N]$ and each X_N is a feature vector of values for each sample

$$\log P(X_1^{(i)}, X_2^{(i)}|Y=c) = f(x)$$

$$f(x) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)}$$

$$\log f(x) = \frac{-k}{2}\log(2\pi|\Sigma|) - \frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)$$

$$\log f(x) = -k\log(2\pi) - \frac{k}{2}\log(|\Sigma|) - \frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)$$

$$\log f(x) = -k\log(2\pi) - \frac{k}{2}\log(|\Sigma|) - \frac{1}{2}(XX^T\Sigma^{-1} - 2\mu\Sigma^{-1}X^T + \mu\Sigma^{-1}\mu^T)$$

$$\frac{\partial}{\partial[\mu_1, \mu_2]}\log f(x) = \left[\frac{\partial \log f(x)}{\partial \mu_1} \quad \frac{\partial \log f(x)}{\partial \mu_2}\right]$$

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{d=1}^{D} X_d$$

$$\boxed{\hat{\mu}_{MLE} = \frac{1}{N}\begin{bmatrix} \sum_{d=1}^{D} X_{d,i} & \sum_{d=1}^{D} X_{d,i+1} \end{bmatrix}}$$

↳ which is the sample estimate ~~for~~ for all i, i+1

↳ For the covariance matrix, it is also a sample estimate:

$$\boxed{\hat{\Sigma}_{MLE} = \frac{1}{n-1}(X_i - \hat{\mu})(X_i - \hat{\mu})^T}$$

4. When data is sparse, there is a risk for the probability of an item being in a class of being equal to zero, especially when we have a lot of features as we did in the bag of words model. This is an issue because when the product of the posterior estimates is calculated, and even just one of these features has a posterior estimate of zero, our whole entire MLE will be equivalent to zero, and the optimal classifier would not be able to classify samples with MLE or zero. To mitigate these risks, we can use prior information and tune our algorithms with hyperparameters, such as the beta parameter we used in question 1.

5.

Question 5:

Multivariate normal posterior hyperparameters:

$$\left(\Sigma_0^{-1} + n\Sigma^{-1}\right)^{-1}\left(\Sigma_0^{-1}\mu_0 + n\Sigma^{-1}\bar{x}\right)$$

↳ where $\bar{x}$ is the sample mean.

The MAP estimate would be

$$\hat{\mu}_{MAP} = \frac{1}{N}\left[\sum_{d=1}^{D} x_{d,i} + \left(\Sigma_0^{-1} + n\Sigma^{-1}\right)^{-1}\left(\Sigma_0^{-1}\mu_i + n\Sigma^{-1}\bar{x}\right),\right.$$

$$\left.\sum_{d=1}^{D} x_{d,i+1} + \left(\Sigma_0^{-1} + n\Sigma^{-1}\right)^{-1}\left(\Sigma_0^{-1}\mu_{i+1} + n\Sigma^{-1}\bar{x}\right)\right]$$

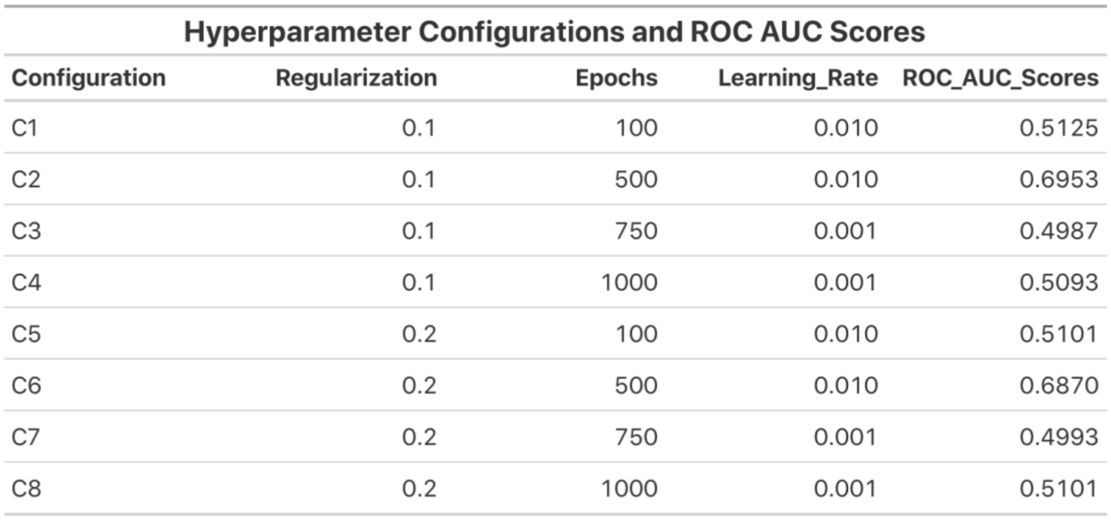
$$\hat{\Sigma}_{MAP} = \frac{1}{n-1}\left(X_i - \hat{\mu}\right)\left(X_i - \hat{\mu}\right)^T$$

$$+\left(\Sigma_0^{-1} + n\Sigma^{-1}\right)^{-1}\left(\Sigma_0^{-1}\hat{\mu} + n\Sigma^{-1}\bar{x}\right)$$

3. Logistic Regression

1. Given that these hyperparameters can span a wide range of combinations, I by choosing random hyperparameters which were a regularization of 0.1, 50 epochs, and a learning rate of 0.01. From there, I began changing one hyperparameter each time to see how that hyperparameter affected both the loss and the ROC_AUC score. Since we are using ROC_AUC score, I plotted each combination of features and its ROC_AUC score and found out a couple of important factors in tuning the hyperparameters. First, a learning rate of 0.001 seems to be too small to converge on the optimal minimum of the loss function with a smaller number of epochs, so I will start with a learning rate of 0.01. Secondly, the number of epochs did not change my ROC_AUC score when I tried 150 and 300 epochs with a learning rate of 0.01 and a regularization of 0.1, so I chose to use four numbers of epochs: 100, 500, 750, and 1000 to test if drastically increasing the number of epochs improves the ROC_AUC score. Given that my first iteration of tests did not yield a high ROC_AUC score when both the number of epochs and learning rate were small, I paired a larger learning rate of 0.01 with the smaller number of epochs, and a learning rate of 0.001 for the larger number of epochs. It is also important to note that I tested a learning rate of 1 and the loss bounced back and forth between positive and negative values, which tells me that that is too large of a learning rate to converge on the minimum of the loss function. For the regularization hyperparameter, I tried both 0.1 and 0.2. Overall, I tried the following combinations of hyperparameters:

Table 3: Randomly selected hyperparameter configurations and respective ROC_AUC Scores, where C1…CN represent the label of the configuration, and Regularization, Epochs, and Learning_Rate represent the specified values of each hyperparameter.



Hyperparameter Configurations and ROC AUC Scores

Configuration	Regularization	Epochs	Learning_Rate	ROC_AUC_Scores
C1	0.1	100	0.010	0.5125
C2	0.1	500	0.010	0.6953
C3	0.1	750	0.001	0.4987
C4	0.1	1000	0.001	0.5093
C5	0.2	100	0.010	0.5101
C6	0.2	500	0.010	0.6870
C7	0.2	750	0.001	0.4993
C8	0.2	1000	0.001	0.5101

Increasing the number of epochs to a large number (i.e., 750 and 1000) with a smaller learning rate did not improve the ROC_AUC score of my model. Next, I tried increasing the number of epochs while testing a learning rate of 0.01 because the learning rate above for epoch counts of 750 and 1000 did not converge on the minimum.

Table 4: Hyperparameter configurations for larger epoch counts. Increasing the number of epochs improved the ROC_AUC score.

Hyperparameter Configurations and ROC AUC Scores

Configuration	Regularization	Epochs	Learning_Rate	ROC_AUC_Scores
C9	0.1	1000	0.01	0.7736
C10	0.1	1500	0.01	0.8133

From the above data, it is evident that increasing the number of epochs improves the ROC_AUC score. So, for the final set of hyperparameters, I decided to keep the number of epochs at 1500 and adjust the regularization and learning rate to see if that changed the model's performance.

Table 5: Final hyperparameter configurations for 1500 Epochs. A regularization of 0.01 and a Learning Rate of 1 yielded the best ROC AUC Score. Interestingly, changing the regularization term from 0.01 to 0.15 did not seem to impact the ROC_AUC score, but I would think that is due to the large number of epochs.

Hyperparameter Configurations and ROC AUC Scores

Configuration	Regularization	Epochs	Learning_Rate	ROC_AUC_Scores
C11	0.01	1500	0.01	0.9273
C12	0.01	1500	1.00	0.9400
C13	0.01	1500	10.00	0.9291
C14	0.15	1500	0.01	0.9273
C15	0.15	1500	1.00	0.9400
C16	0.15	1500	10.00	0.9388

Table 6: Best selected Hyperparameters and ROC_AUC scores for both training and validation sets.

Final Hyperparameter Configurations and ROC AUC Scores

Regularization	Epochs	Learning_Rate	Validation_ROC_AUC_Score	Test_ROC_AUC_Score
0.01	1500	1	0.94	0.9806

3.

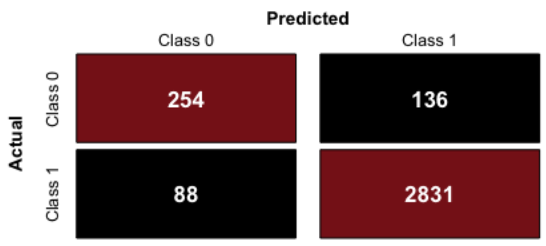


Figure 3: Confusion Matrix for the Logistic Regression Bag of Words Classifier. The hyperparameters for this model were Regularization = 0.01, Epochs = 1500, and Learning Rate = 1.0.