# Homework 3: Introduction to Bayesian Methods

Harvard CS 109B, Spring 2017

## Problem 1: Authorship Attribution

In this problem, the task is to build a model that can predict if a given news article was written by one of two authors. We will explore two different probabilistic models for this binary classification task. Your model will be evaluated based on its classification accuracy.

#### Pre-Processing Details

The data is a subset of the Reuter's $50 \times 50$ data set in the UCI repository. We provide you with pre-processed versions of these data sets `dataset1_train_processed_subset.txt` and `dataset1_test_processed_subset.txt`.

The text articles have been converted into numerical feature representations. We use the *bag-of-words* feature encoding, where each article is represented by a vector of word counts. More specifically, we construct a dictionary of $K$ frequent words in the corpus, and represent each article using a $K$-length vector: the $i$-th entry in this vector contains the number of times the dictionary word $i$ occurs in the article.

We then further preprocessed the data to include the **100 words** that are distinctive to each author to improve our predictions. The dictionary of words used for this representation have been provided in the file `words_preprocessed.txt`.

*Hint*: Make use of the `header` argument in either `read.csv` or `read.table`.

We begin with a simple Naive Bayes classifier for this task, and will then explore a fully Bayesian modeling approach.

## Part 1a: Naive Bayes Classifier

Fit a Naive Bayes classification model to the training set and report its classification accuracy on the test set. The input to this model is the word count encoding for an article, and the output is a prediction of the author identity.

**Questions**:

- Using 0-1 loss what is the overall accuracy?
- Explain to a layperson in clear language the problem in using a Naive Bayes probabilistic model for this task.

*Hint:* You may use the `naiveBayes` function in the `e1071` library for fitting a Naive Bayes classifier.

## Part 1b: Dirichlet-Multinomial Model

Let us consider an alternate Bayesian approach for authorship inference. We recommend a Dirichlet-Multinomial probabilistic model for articles by each author. The author identity for an article can be predicted by computing the posterior predictive probability under this model. This is similar to the approach

described in class for the Beatles musical authorship inference example, except that we shall use word features in place of transition couplets.

**Probability model:** Let $(y_1^A, y_2^A, \ldots, y_K^A)$ denote the total counts of the $K$ dictionary words across the articles by author $A$, and $(y_1^B, y_2^B, \ldots, y_K^B)$ denote the total counts of the $K$ dictionary words across the articles by author $B$. We assume the following *multinomial model*:

$$p(y_1^A, y_2^A, \ldots, y_K^A) \propto (\theta_1^A)^{y_1^A} (\theta_2^A)^{y_2^A} \ldots (\theta_K^A)^{y_K^A}$$

$$p(y_1^B, y_2^B, \ldots, y_K^B) \propto (\theta_1^B)^{y_1^B} (\theta_2^B)^{y_2^B} \ldots (\theta_K^B)^{y_K^B}.$$

The model parameters $(\theta_1^A, \ldots, \theta_K^A)$ and $(\theta_1^B, \ldots, \theta_K^B)$ are assumed to follow a *Dirichlet* prior with parameter $\alpha$.

We provide you with an `R` function (`posterior_pA`) to calculate the posterior predictive probability under the above model, i.e. the posterior probability that a given test article was written by author $A$, based on the training data. The input to this function is

- the Dirichlet parameter $\alpha$,
- the total word counts $(y_1^A, y_2^A, \ldots, y_K^A)$ from all articles by author $A$ in the training set,
- the total word counts $(y_1^B, y_2^B, \ldots, y_K^B)$ from all articles by author $B$ in the training set, and
- the word counts $(\tilde{y}_1, \ldots, \tilde{y}_K)$ for a new test article.

The output is the posterior probability $P(A \,|\, data)$ that the test article was written by author $A$.

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #
# function: calculates the probability author is Aaron Pressman
#   See lecture notes for formula
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #

posterior_pA = function(alpha, yA = NULL, yB = NULL, y_til = NULL){
    # number of features
    K = length(yA)
    # total word counts
    n = sum(y_til)
    nA = sum(yA)
    nB = sum(yB)
    # posterior predictive distribution of being class A
    A1 = lfactorial(n) + lfactorial(nA) - lfactorial(n + nA)
    A2 = sum(lfactorial(y_til + yA)) - sum(lfactorial(y_til)) - sum(lfactorial(yA))
    A3 = lfactorial(n + nA) + lgamma(K*alpha) - lgamma(n + nA + K*alpha)
    A4 = sum(lgamma(y_til + yA + alpha) - lfactorial(y_til + yA) - lgamma(alpha))
    A5 = lfactorial(nB) + lgamma(K*alpha) - lgamma(nB + K*alpha)
    A6 = sum(lgamma(yB + alpha) - lfactorial(yB) - lgamma(alpha))
    R_A = exp(A1 + A2 + A3 + A4 + A5 + A6)
    # posterior predictive distribution of being class B
    B1 = lfactorial(n) + lfactorial(nB) - lfactorial(n + nB)
    B2 = sum(lfactorial(y_til + yB)) - sum(lfactorial(y_til)) - sum(lfactorial(yB))
    B3 = lfactorial(n + nB) + lgamma(K*alpha) - lgamma(n + nB + K*alpha)
    B4 = sum(lgamma(y_til + yB + alpha) - lfactorial(y_til + yB) - lgamma(alpha))
    B5 = lfactorial(nA) + lgamma(K*alpha) - lgamma(nA + K*alpha)
    B6 = sum(lgamma(yA + alpha) - lfactorial(yA) - lgamma(alpha))
    R_B = exp(B1 + B2 + B3 + B4 + B5 + B6)
    # probability of being class A
    pA = R_A/(R_A + R_B)
```

```
    return(pA)
}
```

One can use the above function to infer authorship for a given test article by predicting author $A$ if $p_i = p(A \,|\, y_i, data) > 0.5$ and author $B$ otherwise.

**Loss function**

Evaluate the classification accuracy that you get on the test set using the log-loss function

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i),$$

where $y_i$ is the binary author identity in the test set for article $i$ and $p_i$ is the posterior mean probability that article $i$ is written by author $A$. Along with the following choices of the Dirichlet parameter:

- $\alpha$ is set to 1
- $\alpha$ is tuned by cross-validation on the training set under log-loss – you can use the cross-validation function provided in HW1 as a skeleton.

**Questions**:

- What does setting $\alpha = 1$ imply?
- Do the above optimization. What is the optimal value of $\alpha$? What is your final log-loss prediction error?
- For the optimal value of $\alpha$, how do the accuracies (based on $0-1$ loss) obtained compare with the previous Naive Bayes classifier?

# Part 1c: Monte-Carlo Posterior Predictive Inference

In the above `R` function, the posterior predictive distribution was computed using a closed-form numerical expression. We can compare the analytic results to those resulting from Monte Carlo simulation.

We next provide you with an alternate `R` function (`approx_posterior_pA`) that calculates posterior predictive probabilities of authorship using Monte-Carlo simulation. The code takes the number of simulation trials as an additional input.

```
# This function is an approximation of the above exact calculation of p(A|Data):
#
# 1. Make sure to install the MCMCpack and MGLM packages to use this funciton
#
# 2. It isn't written very efficiently, notice that a new simulation from posterior
#   is drawn each time. A more efficient implementation would be to instead
#   simulate the posteriors (post_thetaA, etc.) once and hand them to
#   approx_posterior_pA to calculate the probability.

if(FALSE){
    library('MCMCpack')
    library('MGLM')
```

```
}

approx_posterior_pA = function(alpha = 1, yA = NULL, yB = NULL, y_til = NULL, n.sim = NULL){
    # number of features
    K = length(yA)
    alpha0 = rep(alpha, K)
    # simulate parameters from the posterior of the Dirichlet-Multinomial model
    post_thetaA = MCmultinomdirichlet(yA, alpha0, mc = n.sim)
    post_thetaB = MCmultinomdirichlet(yB, alpha0, mc = n.sim)
    # calculate the likelihood of the observation y_til under simulated posteriors
    # note: ddirm calculates by-row likelihoods for (data, parameter) pairs
    y_til_mat = matrix(rep(y_til, n.sim), nrow = n.sim, byrow = TRUE)
    likeA = exp(ddirm(y_til_mat, post_thetaA))
    likeB = exp(ddirm(y_til_mat, post_thetaB))
    # integrate over simulated parameters
    marginal_pA = sum(likeA)
    marginal_pB = sum(likeB)
    # calculate probability of A
    pA = marginal_pA/(marginal_pA + marginal_pB)

    return(pA)
}
```

Consider the situation in Part 1b, using the Monte-Carlo approximation in `approx_posterior_pA` numbers of simulation trials (you may set $\alpha$ to the value chosen by cross-validation in part 1b).

**Questions**:

- At what point does doing more simulations not give more accuracy in terms of prediction error?
- Report on the number of simulations you need to match the test accuracy in part 1b. Does increasing the number of simulations have a noticeable effect on the model's predictive accuracy? Why or why not?

## Part 1d: Author vocabulary analysis

The prescribed Bayesian model can also be used to analyze words that are most useful for inferring authorship. One way to do this is to compute or approximate the posterior distribution of the ratio of multinomial model parameters (relative ratio) for one author relative to the other, and identify the words that receive high values of this ratio. More specifically, we can calculate this ratio of the posterior parameter values

$R_k = \theta_k^A/(\theta_k^A + \theta_k^B), k = 1, ..., K$

and return a Monte-Carlo approximation of $\mathbb{E}[R_k \,|\, data]$. The largest $R_k$ this would indicate high relative usage of a word for author A while the smaller values would indicate the same instead for author B.

We again provide you with the relevant `R` code. The input to the code is the Dirichlet parameter $\alpha$, the number of MC draws `n.sim` for the approximation and the total word counts $(y_1^A, y_2^A, \ldots, y_K^A)$ and $(y_1^B, y_2^B, \ldots, y_K^B)$ from the training set articles written by author $A$. The output is a vector containing the approximate values of $\mathbb{E}[R_k \,|\, data]$.

```
# This function claculates an approximation to E[R_k|data] described above.
posterior_mean_R = function(alpha = 1, yA = NULL, yB = NULL, n.sim = NULL){
    # number of features
```

```
    K = length(yA)
    alpha0 = rep(alpha, K)
    # posterior parameter values
    post_thetaA = MCmultinomdirichlet(yA, alpha0, mc = n.sim)
    post_thetaB = MCmultinomdirichlet(yB, alpha0, mc = n.sim)
    # empirical values of R_k
    R = post_thetaA/(post_thetaA + post_thetaB)
    # calculate approximation to E[R_k|data]
    ER = apply(R, 2, mean)
    return(ER)
}
```

Using the `posterior_mean_R` function and the word dictionary `words.txt`, list the top 25 words that are indicative of each author's writing style (you may set $\alpha$ and `n.sim` the values chosen in part 1b and 1c respectively).

**Questions**:

- Given the above explanation and code, how can we interpret $E[R_k]$?
- Do you find visible differences in the authors' choice of vocabulary?

## Problem 2: Bayesian Logisitic Regression

You are provided with data sets `dataset_2_train.txt` and `dataset_2_test.txt` containing details of contraceptive usage by 1934 Bangladeshi women. There are 4 attributes for each woman, along with a label indicating if she uses contraceptives. The attributes include:

- `district`: identifying code for the district the woman lives in
- `urban`: type of region of residence
- `living.children`: number of living children
- `age_mean`: age of women (in years, centred around mean)

The women are grouped into 60 districts. The task is to build a classification model that can predict if a given woman uses contraceptives.

Use simple visualizations to check if there are differences in contraceptive usage among women across the districts. If so, would we benefit by fitting a separate classification model for each district? To answer this question, you may fit the following classification models to the training set and compare their accuracy on the test set:

- A separate logistic regression model for each district
- A single logistic regression model using the entire training set (ignoring the district information)

Fit a Bayesian hierarchical logistic regression model to the training set using the district as a grouping variable, and evaluate its accuracy on the test set. Does the Bayesian hierarchical model yield better accuracies than the two models fitted previously? Give an explanation for what you observe. Also, explain why the hierarchical logistic regression is a compromise between fitting separate logistic regressions for each district and a single logistic regression ignoring district membership.

*Hint:* You may use the `MCMChlogit` function in the `MCMCpack` package for fitting a Bayesian hierarchical logistic regression model.