Assignment 2

Aidan Fischer

2/14/2022

I pledge my honor that I have abided by the Stevens Honor System.

Homework assignments will be done individually: each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. Electronic submission on Canvas is mandatory. **Do not use any package/tool for implementing the algorithms; You can use packages for matrix/vector operations and data processing**.

1. **Linear Discriminant Analysis** (20 points) Please download the "processed.cleveland.data" from Heart-disease data set in the UCI Machine Learning repository and implement a binary Fisher's Linear Discriminant Analysis to distinguish no-heart disease (0) from heart disease(1 – 4) and report your results. Please read "heart-disease.names" for the explanation of features (13 features are used). Split data into training (80%) and test (20%). Write down each step of your solution.

   Notes: names document mentioned that missing data values are denoted by a -9 in the dataset. There were no -9s in the set, but there were ?s. I decided to keep them as -9, since not including those specific attributes would be the same as setting those parts to 0, which isn't much better. An alternative would be to not include those particular samples.

   1) Load full dataset into program.

   2) Split dataset into 80% training and 20% test data by splitting a fixed-seed random permutation (for a deterministic program result) of the dataset into 80% and 20% partitions.

   3) Split training data into two classes, + and -, where + is 1-4, indicating heart disease, and - is 0, indicating no heart disease, using the 14th member of the dataset, the actual result.

   4) Using split training data, compute S_W^{-1}, $\mathbf{m}_+$ and $\mathbf{m}_-$ in order to compute $w_{lda} = S_W^{-1}(\mathbf{m}_+ - \mathbf{m}_-)$. Ignores actual result member of the dataset.

   5) Choose threshold value as mean of $m_+ = \mathbf{w}^T\mathbf{m}_+$ and $m_- = \mathbf{w}^T\mathbf{m}_-$

   6) Evaluate chosen threshold and w on training and test sets. Output training and test accuracy in percentage of correct guesses.

   Results:

   The w resulting from LDA performs with an accuracy of about 80%. I assume the inaccuracy results from outliers of the data that don't fit with the rest of the pattern.

   Exact output:

   Results for problem 1: Output of do_LDA

   Chosen w:

[4.37725902e-03, 6.74351296e-01, 3.88990594e-01, 9.94197607e-03,
5.98772208e-04, -1.48337775e-01, 1.20918685e-01, -7.10796144e-03,
4.95241402e-01, 5.73658028e-02, 2.19786069e-01, 2.15629942e-01,
1.17410686e-01]

Determined threshold: 3.7235726510281584

Results of training: Chosen w gets:

86.61% of C1 correct

81.89% of C2 correct

81.66% of test set correct.

2. **Generative methods vs Discriminative methods** (50 points) Please download the breast cancer data set from UCI Machine Learning repository.

   1. (10 pts) Show that the derivative of the error function in Logistic Regression with respect to $\mathbf{w}$ is:

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n)\mathbf{x}_n$$

The loss function for logistic regression is
$-ln \prod_{n=1}^{N} p(C_1|x_n)^{y_n}(1 - p(C_1|x_n))^{1-y_n}$
We also have $p(C_1|x) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(x)$
where $\sigma(a) = \frac{1}{1+e^{-a}}$
Going from there,
$\frac{\partial}{\partial w}[[-ln \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n}]$
$= \frac{\partial}{\partial w}[-ln \prod_{n=1}^{N} f(\mathbf{x}_n)^{y_n}(1 - f(\mathbf{x}_n))^{1-y_n}]$
$= \frac{\partial}{\partial w}[-\sum_{n=1}^{N} ln(f(\mathbf{x}_n)^{y_n}(1 - f(\mathbf{x}_n))^{1-y_n})]$
$= \frac{\partial}{\partial w}[-\sum_{n=1}^{N} [ln(f(\mathbf{x}_n)^{y_n}) + ln(1 - f(\mathbf{x}_n)^{1-y_n})]]$
$= \frac{\partial}{\partial w}[-\sum_{n=1}^{N} [y_n ln(f(\mathbf{x}_n)) + (1 - y_n)ln(1 - f(\mathbf{x}_n))]]$
$= \frac{\partial}{\partial w}[-\sum_{n=1}^{N} [y_n ln(\sigma(\mathbf{w}^T\mathbf{x}_n + \omega_0)) + (1 - y_n)ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_n + \omega_0))]]$
$= \frac{\partial}{\partial w}[-\sum_{n=1}^{N} [y_n ln(\frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + (1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]]$
$= -\sum_{n=1}^{N} [\frac{\partial}{\partial w} y_n ln(\frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [\frac{\partial}{\partial w}(-y_n ln(1 + e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)})) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(-y_n(\frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})\frac{\partial}{\partial w}(1 + e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)})) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{-y_n \frac{\partial}{\partial w}(1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)})}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{-y_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)} \frac{\partial}{\partial w}(-(\mathbf{w}^T\mathbf{x}_n+\omega_0))}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{y_n\mathbf{x}_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + \frac{\partial}{\partial w}(1 - y_n)ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{y_n\mathbf{x}_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + (1 - y_n)\frac{\partial}{\partial w}ln(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{y_n\mathbf{x}_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + (1 - y_n)\frac{1}{1-\frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}}\frac{\partial}{\partial w}(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{y_n\mathbf{x}_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + (1 - y_n)\frac{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}-1}\frac{\partial}{\partial w}(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$
$= -\sum_{n=1}^{N} [(\frac{y_n\mathbf{x}_n e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}) + (1 - y_n)\frac{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}\frac{\partial}{\partial w}(1 - \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}\frac{\partial}{\partial w}(\frac{-1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}(\frac{\frac{\partial}{\partial w}(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})}{(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})^2})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}(\frac{\frac{\partial}{\partial w}(e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})}{(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})^2})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}(\frac{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}\frac{\partial}{\partial w}(-(\mathbf{w}^T \mathbf{x}_n+\omega_0))}{(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})^2})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}(\frac{-e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}(\mathbf{x}_n)}{(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})^2})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)\frac{1}{e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}(\frac{-e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}(\mathbf{x}_n)}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + (1-y_n)(\frac{-\mathbf{x}_n}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})]$

$= -\sum_{n=1}^{N}[(\frac{y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}) + \frac{\mathbf{x}_n(y_n-1)}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}]$

$= -\sum_{n=1}^{N}\frac{\mathbf{x}_n(y_n-1)+y_n \mathbf{x}_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}$

$= -\sum_{n=1}^{N}\mathbf{x}_n(\frac{(y_n-1)+y_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$-\sum_{n=1}^{N}\mathbf{x}_n(\frac{(y_n-1)+y_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$= -\sum_{n=1}^{N}\mathbf{x}_n((y_n-1)+y_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})(\frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$= -\sum_{n=1}^{N}\mathbf{x}_n(y_n+y_n e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}-1)(\frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$= -\sum_{n=1}^{N}\mathbf{x}_n(y_n(1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)})-1)(\frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$= -\sum_{n=1}^{N}\mathbf{x}_n(y_n-\frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

Recall that $f(\mathbf{x}_n) = \sigma(\mathbf{w}^T \mathbf{x}_n+\omega_0) = \frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}}$

$-\sum_{n=1}^{N}\mathbf{x}_n(y_n-\frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x}_n+\omega_0)}})$

$= -\sum_{n=1}^{N}\mathbf{x}_n(y_n-f(\mathbf{x}_n))$

$= \sum_{n=1}^{N}\mathbf{x}_n(f(\mathbf{x}_n)-y_n)$

$= \sum_{n=1}^{N}(f(\mathbf{x}_n)-y_n)\mathbf{x}_n$

Done

2. (20 pts) Implement a logistic regression classifier with maximum likelihood (ML) estimator using Stochastic gradient descent and Mini-Batch gradient descent algorithms. Divide the data into training and test. Choose a proper learning rate. Use cross-validation on the training data to choose the best model and report the recall, precision, and accuracy on malignant class prediction (class label malignant is positive) on the test data using the best model. Write down each step of your solution.

   Notes: When executing the results, I began thinking that the loss function may not be the best indicator of the accuracy of a model here. A single wrong guess, where say it reports a value very close to 0 when the actual answer is one, can result in loss function skyrocketing. I decided to keep using error to evaluate the models, because I wasn't sure if using recall, precision, accuracy was a better metric. However, I included a P2_2_output.txt in the Code folder that includes RPA and error results for all models tested.

   1) Load data into program

   2) Move class variable to end of each sample, replace with class indicator $y_n \in \{0, 1\}$.

   3) Put a 1 at the start of each sample array for the bias term

   4) Split dataset into training and test sets

   5) Further split trainset set into K = 5 subsets

6) Perform K-Fold Cross validation on those 5 subsets

6.1) Run Stochastic and Minibatch gradient descent for each i = 1, 2, ..., k with the ith training subset as the validation set

6.2) Select best model for each of stochastic and minibatch

7) Get results for test sets for both models, report results.

Results:

Best stochastic model:

w = [1.60259679e-03 1.18276110e-02 1.45042448e-02 6.65165166e-02 2.10457290e-02 9.51523926e-05 -1.07705293e-04 -3.07533068e-04 -1.27972131e-04 1.96314809e-04 8.27400463e-05 6.76767532e-05 7.81694979e-04 -3.95987327e-04 -2.28350773e-02 6.64402373e-06 -2.60843241e-05 -4.49298472e-05 -6.63765629e-06 1.57451088e-05 3.14020845e-08 1.24372991e-02 1.67893157e-02 6.56492733e-02 -3.29560145e-02 1.15808855e-04 -3.87059815e-04 -7.18459003e-04 -1.73052379e-04 2.41305931e-04 6.11358320e-05]

Model Error on training set = 17.44710568214963

Best Minibatch model:

w = [2.05128629e-03 1.91590582e-03 2.05803134e-03 1.87850164e-03 1.78410055e-04 2.01719471e-03 1.42723823e-03 -1.65687244e-03 -1.59789607e-03 2.03267469e-03 2.04778338e-03 2.45735275e-04 1.87529500e-03 -3.91442852e-04 -1.20729595e-03 2.06798239e-03 1.96946414e-03 1.61619290e-03 1.73824078e-03 2.06046845e-03 1.90369962e-03 1.72928346e-03 2.01654117e-03 1.68550525e-03 -8.61173220e-04 1.99125716e-03 5.46426227e-05 -1.22257858e-03 -1.03833621e-03 1.99581551e-03 2.00970193e-03]

Model Error on training set = 55.21085010002854

Evaluating models on test set...

Evaluation complete.

Stochastic Model Error on Test Set: 21.67963031151801

Minibatch Model Error on Test Set: 69.49262132360123

Recall, Precision, Accuracy on test set:

Stochastic Model: (0.8421052631578947, 0.9142857142857143, 0.9203539823008849)

Minibatch Model: (0.9736842105263158, 0.5362318840579711, 0.7079646017699115)

Recall, Precision, Accuracy on models' respective validation sets

Stochastic Model: (0.9166666666666666, 0.868421052631579, 0.9130434782608695)

Minibatch Model: (1.0, 0.65, 0.7692307692307693)

From these results, Minibatch seems to do worse overall than stochastic, but is really good at recall. However, it's accuracy and precision metrics leave much to be desired. On the other hand, stochastic isn't perfect, but is about 90% in all three metrics, and has a far lower error metric.

3. (20 pts) Implement a probabilistic generative model (the one in our lecture) for this problem. Use cross-validation on the training data and report the recall, precision, and accuracy on malignant class prediction (class label malignant is positive) on the test data using the best model. Write down each step of your solution.

   1) Load data into program

   2) Move actual result to end of each x and transform into class identifier yn = 0,1

   3) Split into training and testing set

   4) Further split training set into K = 5 sets for K-fold cross validation

   5) Split all 5 sets into corresponding C1 and C2 sets

   6) Run k-fold cross validation, using generative maximum likelihood estimators to get w and w_0

   7) Since there is no general loss function, I choose to use a metric of recall * precision * accuracy, choosing the model where this is closest to 1, which I believe will show the model that is best overall. I didn't use this approach in part 2 because logistic regression has a defined loss function.

8) Display best model, recall, precision, accuracy, and the product of RPA.
Results
Best model:
w = [7.54181411e+00, -1.00019229e-01, -8.62431379e-01, -1.10675850e-02,
5.86122456e-01, 8.09134951e+01, -1.86691446e+01, -5.84365878e+01,
1.00845617e+00, 2.29849086e+01, -5.09793409e+00, 2.16454403e-01,
-2.32188849e-01, 3.05707101e-02, -8.04199228e+01, -2.56500122e+01,
4.36960567e+01, -1.50838654e+02, -5.92237568e+01, 4.05788966e+02,
-4.13805733e+00, -1.02914034e-01, 5.71387503e-02, 2.13603907e-02,
-3.12900701e+01, 5.40312608e+00, -7.54736978e+00, 7.64410123e-02,
-5.77150523e+00, -1.19933617e+02]
w0 = 42.8210313
Validation Recall: 0.9444444444444444
Validation Precision: 0.9714285714285714
Validation Accuracy: 0.967032967032967
Validation Performance Metric (R*P*A): 0.8872143729286586
Test Set Recall: 0.868421052631579
Test Set Precision: 1.0
Test Set Accuracy: 0.9557522123893806
Test Perf Metric (R*P*A): 0.8299953423381463

3. **Naive Bayes** (20 points) From Project Gutenberg, we downloaded two files: The Adventures of Sherlock Holmes by Arthur Conan Doyle (pg1661.txt) and The Complete Works of Jane Austen (pg31100.txt). Please develop a multinomial Naive Bayes Classifier that will learn to classify the authors from a snippet of text into: Conan Doyle or Jane Austen. A multinomial Naive Bayes uses a feature vector $\mathbf{x} = \{x_1, ..., x_D\}$ as a histogram and model the posterior probability as:

$$p(C_k|\mathbf{x}) \propto p(C_k) \prod_{i=1}^{D} p(x_i|C_k) \tag{1}$$

where $p(x_i|C_k)$ can be estimated by the number of times word i was observed in class C_k plus a smoothing factor divided by the total number of words in C_k

In the test phase, given a new example $\mathbf{x}_t$, you can output the class assignment for this example by comparing $\log p(C_1|\mathbf{x}_t)$ and $\log p(C_2|\mathbf{x}_t)$. If $\log p(C_2|\mathbf{x}_t) > \log p(C_1|\mathbf{x}_t)$, assign C_2 to this example. You need to divide the data into training and test. For the words that appear in the test set but not in the training set, you can either ignore these words in the probability calculation or you can apply smoothing in $p(x_i|C_k)$ (e.g. Laplace smoothing).

You can apply some preprocesssing techniques such as removing stop-words and punctuation. You can remove the unrelated text in the beginning of each file. Make sure the test data has equal number of samples from Conan Doyle and Jane Austen.

Report accuracy on test data using your Naive Bayes classifier.

Step 1) Manually removed unrelated text at start and end of two files.

Step 2) Load files into program

Step 3) Programmatically remove punctuation

Step 4) Split on spaces to get list of words for each class

Step 5) Filter stopwords using NLTK's stopword list

Note: One set contains 800000 words and one contains 100000. Since test set has to be same number of samples from each, and test set removes from training set, take 20000 words in samples from both.

Step 5) Split into test and train data. Because Naive Bayes acts on samples with multiple words, collect 2500 samples from both classes, 250 samples with 16-20 words each (inclusive) (250 with 5 words, 250 with 6...), collected randomly from each dataset. This results in 250(16+17+18+19+20) = 22500 words taken from the dataset for each class for testing.

Step 6) Using training data, estimate $p(x_i) = \frac{count(i,C_k)+1}{count(C_k)}$ for each word i in each class C_k, where 1 is a smoothing term.

Step 7) Evaluation of accuracy on model using testing data. Since we took equal samples from both classes for testing, $p(C_k) = 0.5$, where we calculate $log(p(C_k|\mathbf{x}_i)$, where $p(C_k|\mathbf{x}_t) \propto \prod_{i=1}^{D} p(x_i|C_k)$ (Note, since $p(C_1) = p(C_2) = 0.5$, proportionality absorbs the $p(C_k)$ term, and assign the test sample to the class with higher log probability

Results:

Model accuracy: 0.2412

I'm not sure if I implemented it incorrectly, or if it is just the case that Naive Bayes does not work well for this case, but model accuracy is extremely low.

4. **Linear classification** (10 points) Please prove that 1) the multinomial naive Bayes classifier in log-space essentially translates to a linear classifier. 2) Logistic regression is a linear classifier.

1) The multinomial naive Bayes classifier is represented by $p(C_k|\mathbf{x}) \propto p(C_k) \prod_{i=1}^{D} p(x_i|C_k)$.

Taking the log to get this in log space, we obtain

$log[p(C_k) \prod_{i=1}^{D} p(x_i|C_k)]$

$= log[p(C_k)] + log[\prod_{i=1}^{D} p(x_i|C_k)]$

$= log[p(C_k)] + \sum_{i=1}^{D} log[p(x_i|C_k)]$

Where $p(x_i|C_k)$ is estimated by a ratio of the count of x_i in C_k to the total number of elements in C_k

Thinking of each "feature" in this case as it's own count, with each "w" as 1/(count in class), we have a variable feature and constant ws:

$p(x_i|C_k) \rightarrow w_i x_i$

$log[p(C_k)] + \sum_{i=1}^{D} log[w_i x_i]$

For this to be a linear classifier, it's class boundary needs to be linear. The class boundary of naive bayes is if the result of the function for one class is greater than the result for the other. w_1 represents the w_0 for the other class. $x_k i$ represents the count of the word in class k. The boundary is at $log[p(C_1)] + \sum_{i=1}^{D} log[w_{1i} x_{1i}] = log[p(C_2)] + \sum_{i=1}^{D} log[w_{2i} x_{2i}]$

$log[p(C_1)] - log[p(C_2)] + \sum_{i=1}^{D} log[w_{1i} x_{1i}] = + \sum_{i=1}^{D} log[w_{2i} x_{2i}]$

$log[\frac{p(C_1)}{p(C_2)}] + \sum_{i=1}^{D} log[w_{1i}x_{1i}] = \sum_{i=1}^{D} log[w_{2i}x_{2i}]$

$log[\frac{p(C_1)}{p(C_2)}] = \sum_{i=1}^{D} log[w_{1i}x_{1i}] - \sum_{i=1}^{D} log[w_{2i}x_{2i}]$

$log[\frac{p(C_1)}{p(C_2)}] = \sum_{i=1}^{D} log[w_{1i}] + log[x_{1i}] - \sum_{i=1}^{D} log[w_{2i}] + log[x_{2i}]$

Pulling out the constant w terms

$log[\frac{p(C_1)}{p(C_2)}] = Dlog[w_1] - Dlog[w_2] + \sum_{i=1}^{D} log[x_{1i}] - \sum_{i=1}^{D} + log[x_{2i}]$

$log[\frac{p(C_1)}{p(C_2)}] - Dlog[w_1] + Dlog[w_2] = \sum_{i=1}^{D} log[x_{1i}] - \sum_{i=1}^{D} + log[x_{2i}]$

$log[\frac{p(C_1)w_2^D}{p(C_2)w_1^D}] = \sum_{i=1}^{D} log[x_{1i}] - \sum_{i=1}^{D} log[x_{2i}]$

constant left term to w_0

$w_0 = \sum_{i=1}^{D} log[x_{1i}] - \sum_{i=1}^{D} log[x_{2i}]$

Which is linear in log-space.

2) With logistical regression, we have

$f(\mathbf{x}_n) = \sigma(\mathbf{w}^T\mathbf{x}_n + \omega_0) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}$

We need to show that this is a linear classifier, which implies that the decision boundary is linear.

Thus, we should show that the decision boundary is linear. For logistic regression, the decision boundary is at f(x) = 0.5

$0.5 = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}}$

$0.5(1 + e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)}) = 1$

$0.5e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)} = 0.5$

$e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)} = 1$

$ln(e^{-(\mathbf{w}^T\mathbf{x}_n+\omega_0)} = 1)$

$-(\mathbf{w}^T\mathbf{x}_n + \omega_0) = 0$

$\mathbf{w}^T\mathbf{x}_n + \omega_0 = 0$

Which is a linear equation in x, therefore logistic regression is a linear classifier.

Please follow the below instructions when you submit the assignment.

1. You are allowed to use packages for preprocessing data or cross-validation.
2. You shall submit a zip file named Assignment2_LastName_FirstName.zip which contains:
   - a pdf file contains all your solutions for the written part
   - python files (jupyter notebook or .py files)