Slippery Rock University

Naive Bayes Classifier Backend and Frontend Technical Manual

Trevor Hamilton (tjh1003) & Jonathan Stonebreaker(jds1018)

Data Mining - CPSC 405

Dr. Sam Thangiah

Table of Contents

3.....................................................................Backend Main Class and Standard Program Flow

8............................................................................................Expected Input Data File Formating

9...............................................................................................................Output Data File Format

10.............................................................Functions and Data Structures Explanations and Usage

18....................................................................................................................................................GUI

19..............................................................................................................................Additional Notes

Section 1: Backend Main Class and Standard Program Flow

Main Class Breakdown

* Breakdown of the output and function order of the main class with descriptions of what nonprinting function class do in the next section
  + Strings for the input and output excel files to be used
  + Do the import of the input excel file
  + Generate the training data
  + Notify use training data has been generated and print both the training and testing data
  + Generate the model (Likelihood of all the possible values of each attribute)
  + Inform use model has been generated and print the possible classifications and the model
  + Classify everything in the testing data
  + Inform user the guessed classifications have been generated and print of the guessed and acutal classifications
  + Export results to output excel file
* Output symbols mean the following
  + ::
    - Label was printed and this group of data listed belongs to it
  + :
    - Next data point for this grouping
  + ,
    - New group of data
  + [
    - Start of new dimension of data
  + ]
    - end of dimension of data

Standard Order of Function Execution and tips

* readExcelFile(intputFileName);
  + Should be formatted as specified in section 2
* generateTrainingDataGENERATOR(trainingDataSize);
  + Multiple different ones a brief explanation of each is as follows
    - Random
      * Randomly selects a remaining data point to remove from test and add to training while trainingDataLL.size < trainingDataSize
    - First
      * Removes the first trainingDataSize from testing data to add to training data
    - Stride
      * Takes every dataLL.size/trainingDataSizeth data point from testing data to add to training data. Example end result you have 15 data points and you want a training size of 5 15/5=3 you will get every 3rd data point in your training data
    - FromFile
      * Load an excel file directly into training data. Does not remove duplicates expects this file to be formated the same as original imported file except does not require metadata worksheet as it uses the one from the original file
* generateClassifier();
  + Generates the model that will be used to make guesses on future data’s classification
    - Only a naive bayes classifier is implemented a gaussian bayes classifier was hoped to be implemented but that time was spent implementing laplace smoothing
      * naive bayes algorithm put in simple terms is for each possible classification you for each attribute add up the number of times that value and classification occurred and divide that by the number of times that classification occurred, you then for the classified take the number of times that classification occurred divided by the size of your data set. This generates a likelihood of each value for each classification which will be used in generateClassifications() latter (If smoothing is turned on it will add the laplace smoother value described below) (Good explanation video done by 5 Minutes with Ingo can be found [here](https://www.youtube.com/watch?v=IlVINQDk4o8))
    - Function is entirely ready to have gaussian implemented you would just need to write a gaussian classifier and have a way to select between the two functions
    - Uses a laplace smoother to help with unseen values in an attribute if laplace smoother is greater than 0
      * laplace smoothing in simple terms is adding the constant k/(k\*n) to each distinct attribute values likelihood where k is the smoothing value (normally 1) and n is the number of distinct values for that attribute (also applied to classifier likelihood and n is number of distinct classifications)
      * This prevent multiplying a 0 into your likelihood and have a strongly classified value now equalling 0 because a value showed up that was unknown
* generateClassifications();
  + Uses likelihoods generated before to guess what the classification is for each data point in the test data
  + calls classify(testDataLL(i)) for each data point in the test data. If you wanted to you could call this function manually with hand crafted string arrays to get a classification
    - classify for each classification possible multiplies the likelihood of each value of the string array for the classification together and then multiplies it by the current classification storing it in array of the size of possible classifications. It then goes back through and whatever one is the largest is the guessed classification. Gets what the string value is for that classification number and returns that.
      * Note is smoothing is on there should be no 0s occuring
* outputExcelFile(outputFileName);
  + Outputs the results to the specified location in a nicely formatted excel file
    - Sheet 1 contains all the results with the classifier highlighted in yellow and the guessed classification highlighted in green or red if it was right or not. At the very bottom the percent right is displayed with it being green if it is above 90% and red if it is below 90%
    - Sheet 2 contains the training data in the same format as the original excel file
    - Sheet 3 contains the likelihood value for each of each attributes distinct values and then the likelihood of each classification at the end
    - All sheets are labeled at the top and sheet 3 is labeled on the lefthand side

Section 2: Expected Input Data File Formating

Sheet 1 Breakdown

* Contains a list of all the data going to be used
  + Each column is an attribute
  + Each row is one data point

Sheet 2 Breakdown

* Must be listed in the same attribute order as the first sheet
  + Row 1
    - Contains a label for each attribute that you would like it the be referred to
  + Row 2
    - Contains if the data is discrete or continuous, unused but must contain a value
  + Row 3
    - Contains the word classify if this row is the classifier, can be any value but all other cells must be blank

Section 3: Output Data File Format

Sheet 1 Breakdown

* Contains the testing data in the same format as input file
* Contains the label of each attribute along the top
* Contains the guessed classification on the right end
* Contains the percent right at the bottom right
  + Actual classification in yellow
  + Guessed classification in green if it matched the actual red if it didn't
  + Percent is green if above 90% else it is red

Sheet 2 Breakdown

* Contains the training data in the same format as input file
* Contains the label of each attribute along the top

Sheet 3 Breakdown

* Contains the likelihood of each distinct occurrence in the training data for all known classifications
* Contains label of each attribute along the top
* Contains each distinct possible classification along the left
* Contains the likelihood of each possible classification on the bottom

Section 4: Functions and Data Structures Explanations and Usage

Naive Bayes Data Structures in order of appearance

* metadataLL
  + A LinkedList of string arrays where
    - Each link is an attribute in the excel file
    - Array location 0 is the label of that attribute
    - Array location 1 is the datatype of that attribute (unused)
    - Array location 2 says “attribute” if it is an attribute and “classifier” if it is the classifier
  + Generated during excel file read
* testDataLL, trainingDataLL, dataLL
  + Contains a LinkedList of string arrays where
    - each link is one node of the data stored in that linked list
    - Each array location is one attribute of that node
  + dataLL contains every node in training and test data and is generated on load of excel data and is updated if you load another excel file for training data
  + test and training data are normally populated when a training generator is called
* knownClassifications, guessedClassifications, actualClassifications, classificationsLL
  + Is a LinkedList of Strings where
    - Each link is a data node
    - Each string is that nodes classification
  + classificationsLL contains all the classifications and is generated during the excel import
  + actualClassifications contains the classifications of the test data set and knownClassifications contains the classifications of the training data set they are generated during training data generation
  + guessedClassifications contains the classifications the classify guessed to be correct generated when generateClassifications is called
* Classifier
  + Is a LinkedList of LinkedLists of arrays of strings where
    - This first link is the attribute
    - The second link is the distinct values that occur in that attribute in the training set
    - The first location in the array was the text of that distinct value
    - The rest of the locations in the array is the frequency of that value for each classification. Shortly after it becomes the likelihood of that value
    - Array size is number of possible classifier + 1
    - Generated during generateClassifier
* Less important class variables
  + classificationLocation
    - The location of the classification in the excel sheet is used and generated in excel file import and is saved for later in the event that you import an excel file for your training set so that the training set does not need a metadata sheet
  + classificationTypes
    - All the possible values of the classifier seen in the trainingData populated when training data is generated
  + classificationLikelihood
    - Is a LinkedList where each link is a classification and each double is the likelihood of that classification
    - was originally used to settle ties but since those dont happen with smoothing turned on those dont normally happen so it is mostly just used for when we output sheet 3 of the results
  + attributeTotalFrequency
    - A LinkedList of and array of ints where
      * Each link is an attribute
      * Each array location is a possible classification in the training data
      * Each int is the sum of the frequency of all the possible attribute values for that classification
    - Populated during generateClassifier
    - Used if an unseen value of an attribute comes up to build its likelihood with laplace smoothing
  + unseenDataFlag
* laplaceSmoother
  + Constant value to be used to smooth likelihoods so no likelihood is zero
  + Adds to the numerator of each likelihood
  + Add laplaceSmoother\*(the parent classes total occurences) to the denominator

Naive Bayes Functions in order of appearance

* readExcelFile(String fileLocation)
  + Reads excel file from fileLocation expected to be formatted as described in expected input formatting
    - Loads second sheet then first sheet
    - populates dataLL and classificationsLL, metadataLL is populated, also where classificationLocation is set
* writeExcelFile(String fileLocation)
  + Writes excel file to fileLocation overwriting any existing file unless it has no write access. Format is described output data format.
  + Should only be called once done with standard flow or else sections may be empty
* generateTrainingData(int locationIndex)
  + Function called by all generateTrainingData\*(int) (Note that generateTrainingDataFromFile(String) does not)
  + Called with location in the testDataLL you want to add to the trainingDataLL and remove from testDataLL. Also does the remove from actualClassifications and adds into knownClassifications
* generateTrainingDataRandom(int trainingDataSize)
  + Populates testDataLL and actualClassifications with the values of dataLL and classificationsLL
  + Calls generateTrainingData trainingDataSize number of times with a random value between 0 and the size of the testDataLL
* generateTrainingDataFirst(int trainingDataSize)
  + Populates testDataLL and actualClassifications with the values of dataLL and classificationsLL
  + Calls generateTrainingData trainingDataSize number of times with a 0 effectively adding the first trainingDataSize data points to the training data and removing them from the test data
  + While doing this it looks at the node added to the training sets classification and if it is a new classification add it to classificationTypes
* generateTrainingDataStride(int trainingDataSize)
  + Populates testDataLL and actualClassifications with the values of dataLL and classificationsLL
  + Calls generateTrainingData trainingDataSize number of times with the value of the size of (dataLL/trainingDataSize) - the current number data points removed
    - Effectively the same as dataLL.size()/trainingDataSize but since test data shrinks every time the location needs to be offset for the number of items removed
    - Example: if I have a data size of 15 and I called this function with 5 (15/5) it would add every 3rd value to the training data
* generateTrainingDataFromFile(String fileLocation)
  + Populates trainingDataLL and actualClassifications with the values of dataLL and classificationsLL
  + Populates testDataLL and actualClassifications from the file imported. File is expect to follow the same format as the original
* generateClassifierNaiveBayes(int attribute)
  + Possible function called by generateClassifier
  + Computes the frequency of each distinct value of the attribute passed for each possible classifier
    - While doing this it keeps track of the total number of distinct values it encounters for later
  + Computes the likelihood of each of each distinct value of the attribute passed for each possible classifier
    - Does this by dividing the frequency/by the total done earlier in the function
    - If laplace smoothing is turned on k is added to the numerator and k \* the parent (Current classifier or attribute) properties number of occurrences is added to the denominator
  + All this is stored in the classifier for the attribute passed
* generateClassifier()
  + Calls generateClassifierNaiveBayes for each attribute and sets the size of classifier to the number of nodes in the training set
  + Populates classificationLikelihood which was originally used to settle ties but since those dont happen with smoothing turned on those dont normally happen so it is mostly just used for when we output sheet 3 of the results
* classify(String[] node)
  + Normally called by generateClassifications but can be called manually with an array of size of the number of attributes where each location is an attribute in the order they appear in the excel file it will then pick a classification location and looks up the name of that classification and returns that name
  + Classifies by going through each possible classification multiplying all the likelihoods of each attribute for the classification and then multiplies it by the current classification’s likelihood
    - If value of an attribute of the node for a classification does not occur in the training data it builds a score using the laplace smoothing technique described above but with numerator just being the value of your laplace smoother
    - Likely would need to be changed to implement gaussian bayes classification
* generateClassifications()
  + Calls classify for each node adds the returned result to guessedClassifications
* printLinkedListData(LinkedList<String[]> data)
  + Nicely prints out to the console the linkedlist passed (Intended for any of the data linked lists (trainingDataLL, dataLL, etc))
* printLinkedListDataWithClassification(LinkedList<String[]> data, LinkedList<String> classification)
  + Nicely prints out to the console the same thing as above but does not have the classifications removed
* printLinkedListClassifier(LinkedList<LinkedList<String[]>> classifier)
  + Nicely prints out to the console in a 2d fashion the classifier generated. Each line is an attribute each section between the commas is a distinct value that occured in the training data followed by its likelihood for each possible classification in the training data
* printClassification(LinkedList<String> classification)
  + Nicely prints out to the console the passed classificationLL (knownClassifications, guessedClassifications, etc)
* printClassificationTypes()
  + Nicely prints out to the console each distinct classifier that shows up in the training data, is in the same order that they show up for the classifier print so helpful to print just before printing that
* main

GUI

* main
  + Constructs the actual window at runtime
* gui
  + The layout of the gui and the custom event fuctions
    - Browse buttons have custom events that open a file browser and change the text box next to it to say the returned value
      * input browse button also will change the output text box
    - Import button imports the excel file defined in its text box
    - Export button generates the training data, classifier, and then classifies all the values in the test data and then output the results to the excel file defined in its text box
      * generating the training data, classifier, and classifications was originally going to be under the classify button in the metadata tab but that was scrapped due to time constraints

Section 5: GUI

Breakdown

* GUI was not fully fleshed out but as it currently stands is as follows
  + Input File Location Tab
    - A tab to choose a data file to import
    - Browse opens a file browser
  + Metadata Tab (Disabled)
    - Unimplemented tab that would be used to display each attribute in the data and allow user to rename choose data type and select if it is the classifier
  + Output File Location Tab
    - Tab to export data, currently does the classifications on export to prevent exporting file before finished
  + Intended workflow
    - Select a file to import, click import and the selected file is imported, once finished tab changes to metadata tab
      * If copy to data is selected also store a copy in data
    - See if meta data imported correctly and be able to change the labels, data types, and which attribute is the classifier, click classify and start classification updating user how far along it is, once finished tab changes to export tab
    - Chose a location to export the results to, click export and the file is exported, once done open the file
      * If copy to results is selected also store a copy in results

Section 6: Additional Notes

Common Vocab

* Metadata
  + Labels, data type, and isClassificaiton for each attribute
* Training Data
  + Data to be used to generate the classifier
* Testing Data
  + Data with unknown classifications that the classifier is used to guess the classification
* Known Classifications
  + Classifications that belong to the training data, always known
* Guessed Classifications
  + Classifications that the classifier generated without seeing the actual classifications
* Actual Classifications
  + Classifications that belong to the test data, unknown until compring if classifications are right
* Laplace Smoothing
  + A constant added to the likelihood to prevent likelihood from being 0, mathematical explanation above
* Gaussian Bayes Classifier
  + Bayes classifier used for continuous values, unimplemented, but the program is setup to allow it if implemented
* Naive Bayes Classifier
  + Classifier used for discrete values, used for a values in this program since gaussian is unimplemented, mathematical explanation above
* Classification
  + What class a data point belongs to
* Frequency/Occurrences
  + Count of the number of times that value occurred for each possible classification
* Likelihood
  + Frequency divided by the number of times that frequency’s classification occurred (May be uncaught times that likelihood is called frequency is in the code due to originally on the frequency was stored but I think I got them all)
* Model/Classifier
  + What the testing data is passed through to guess the classification mathematically defined above
* Data Point/Node
  + One occurrence of the data in your excel file (Each row would be a data point in a correct excel file)
* Distinct/Seen/Known/Possible value(s)
  + Go through attribute count the number of different values listed, each of those is a distinct/seen/known/possible value. Used each of these based on what made more sense for the data being used at the time
    - Note: these counts only come from the training data unseen values end up just getting the value of smoothing constant