Lab Exercise 4

Naïve Bayes classifier with WEKA

Naïve Bayes classifier is a statistical classifier. It assumes that the values of attributes in the classes are independent. This assumption is called class conditional independence. Naïve Bayes classifier is based on Bayes' theorem, which reads as follows:

$$\mathbf{P(C|X) = (P(X|C) * P(C))/P(X)}$$

where:

- P(C) is the a priori probability of a class C occurrence (i.e. the probability that any sample belongs to class C),
- P(X|C) indicates the posteriori probability that X belongs to class C,
- P(X) is the a priori probability of a sample X occurrence

According to the Bayes' rule, sample X is classified as coming from the class C_i for which the value of $P(C_i|X)$, i = 1, 2, ..., m, is the **highest.**

Because the P(X) probability is constant for all classes, therefore the C_i class, for which the value of $P(C_i |X)$ is the highest, is the C_i class, for which the value of **$P(X| C_i) * P(C_i)$ is the highest.**

$$\mathbf{max[P(X| C_i) * P(C_i)]}$$

How to estimate the a priori probability $P(C_i)$?

- The values of P(Ci) are replaced by the relative frequency of C_i class in the training set:

$$\mathbf{P(Ci) = s_i/n}$$

  where:
  - s_i denotes the number of C_i class samples in the training set
  - n is the number of samples in the training set

  OR

- Assume that all classes have the same probability:

$$\mathbf{P(C_1) = P(C_2) = ... = P(C_m)}$$

How to calculate $P(X|C_i)$?

Probabilities of $P(X_1|C_i)$, $P(X_2|C_i)$, ..., $P(X_n|C_i)$ can be estimated based on the training set as follows:

- if the j-th attribute is a categorical attribute, then $P(X_j|C_i)$ estimate the relative frequency of occurrence of examples from class C_i with the value x_j for the j-th attribute:

$$\mathbf{P(X_j|C_i) = s_{ij}/s_i.}$$

- if the j-th attribute is a continuous attribute, then $P(x_j|C_i)$ is estimated using the density function of Gauss (assuming a normal distribution of the values of the attribute):

$$f(x)\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\cdot\sigma^2}}$$

Exercise 1: Basic manual classification using Naïve Bayes

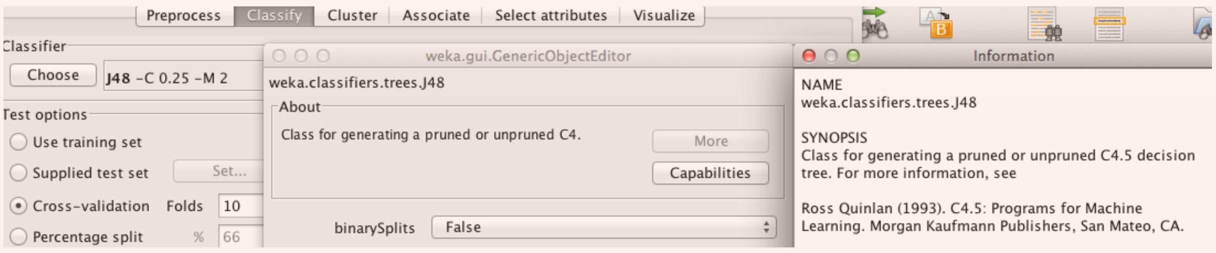
Using Naïve Bayes classifier make a prediction of the class to which the below cases belongs to:

ID	Age	Income	Student	Status	Buy_computer
1	<=30	high	no	single	no
2	<=30	high	no	married	no
3	31..40	high	no	single	yes
4	>40	medium	no	single	yes
5	>40	low	yes	single	yes
6	>40	low	yes	married	no
7	31..40	low	yes	married	yes
8	<=30	medium	no	single	no
9	<=30	low	yes	single	yes
10	>40	medium	yes	single	yes
11	<=30	medium	yes	married	yes
12	31..40	medium	no	married	yes
13	31..40	high	yes	single	yes

a) X1 = (age='31..40', income='high', student = 'yes', status='single')
b) X2 = (age='<=30', income='high', student = 'yes', status='married')
c) X3 = (age='>40, income='medium', student = 'no', status='married')

WEKA tip - Docs

In the last tutorial you have used J48 algorithm to implement a decision tree model following the C4.5 algorithm. C4.5 is an algorithm used to generate a decision tree, which was developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The ID3 algorithm uses "Information Gain" measure. The C4.5 uses "Gain Ratio" measure. If you look at the picture below, you can see that Weka cites the reference of the implemented classifier.



J48 -C 0.25 -M 2

(●) Cross-validation Folds 10

Percentage split % 66

weka.gui.GenericObjectEditor

weka.classifiers.trees.J48

About

Class for generating a pruned or unpruned C4.

binarySplits False

Information

NAME
weka.classifiers.trees.J48

SYNOPSIS
Class for generating a pruned or unpruned C4.5 decision tree. For more information, see

Ross Quinlan (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA.

J48 can deal with both nominal and numeric attributes. However, please remember that this is not always the case, since some classifiers do not have this flexibility, for example linear classifiers. In the following exercise you will explore the behavior of Weka's Naïve Bayes implementations. NB is neither a linear classifier, nor a "divide and conquer" classifier, is a probabilistic classifier. How does NB behave with linguistic datasets? Let's carry out this exploration today...

Exercise 2: Spam filtering with WEKA and Naïve Bayes classifier

We will use Weka to train a Naïve Bayes classifier for the purposes of spam detection.

The data set we will consider is the Spambase set, consisting of tagged emails from a single email account. You can download this from the website.

Read through the description available for this data to get a feel for what you're dealing with. The data has been converted to ARFF format for you:

Preprocessing

Some simple preprocessing of the data will be required before it is ready for use. We can do this in Weka:

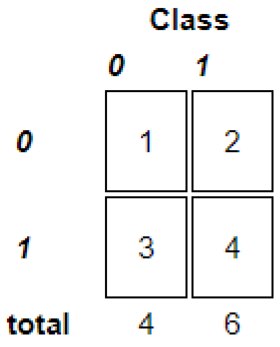
1. Familiarize yourself with the ARFF format
2. From the Preprocess (default) tab in Weka, hit **Open file...** and select the *spambase.arff* file that you downloaded above.
3. A full list of the attributes in this data set will appear in the "Attributes" frame.
4. Delete the *capital_run_length_average, capital_run_length_longest and capital_run_length_total* attributes by checking the box to their left and hitting the **Remove** button.
5. The remaining attributes represent relative frequencies of various important words and characters in emails. We wish to convert these to Boolean values instead: 1 if the word or character is present in the email, 0 if not. To do this, select the **Choose** button in the **Filter frame** at the top of the window, and pick **filters > unsupervised > attribute > NumericToBinary**. Now hit the **Apply** button. All the numeric frequency attributes are now converted to Booleans. Each e-mail is now represented by a 55 dimensional vector representing whether or not a particular word exists in an e-mail. This is the so called “bag of words’ representation (this is clearly a very crude assumption since it does not take into account the order of the words). For more details on “bag of words” see (https://en.wikipedia.org/wiki/Bag-of-words_model)
6. Save this preprocessed data set for future use using the **Save...** button. You will need this for lab 2.

Classification

Given the data set we've just loaded, we wish to train a Naïve Bayes classifier to distinguish spam from regular email by fitting a distribution of the number of occurrences of each word for all the spam and non-spam e-mails. Under the Classify tab:

1. Select **Choose** in the **Classifier frame** at the top and select **classifiers > bayes > NaiveBayes**.
2. Leave the default settings and hit **Start** to build the classifier. Study the output produced, most importantly the percentages of correctly and incorrectly classified instances. You probably will notice that your classifier does rather well despite making a very strong assumption on the form of the data.
   - Can you come up with a reason for the good performance? What would be the main practical problems we would face if we were not to make this assumption for this particular dataset?
   - How long did your classifier take to train and classify? Given this, how scalable do you think the Naïve Bayes classifier is to large datasets? Can you come up with a good reason for this?
3. Examine the classifier models produced by Weka (printed above the performance summary). Find the prior probabilities for each class.
   - How does Naïve Bayes compute the probability of an e-mail belonging to a class (spam/not spam)?
   - Compute the conditional probability of observing the word "3d" given that an e-mail is spam *P(3d|spam)* and that it is non-spam *P(3d|non-spam).* To do this, we need to use the counts of the built model that are produced within the **Classifier output screen** under the **Classify tab**. The general format of the Weka count output is (Note: this is a toy example.

You will need to examine your Weka output to find the true counts for the word "3d".):



	Class	
	0	1
0	1	2
1	3	4
total	4	6

This means that 4 instances (e.g. e-mails) contain that particular attribute value (e.g. the word "3d") in Class 1 (e.g. Is Spam). 2 instances didn't contain that value of the attribute in Class 1. 3 instances of Class 0 contained that attribute value, whilst 1 instance of Class 0 (e.g. Not Spam) didn't contain that attribute value. The totals reflect the number of instances belonging to both classes e.g. the number of e-mails that are Spam and not Spam.

4. For the final part of this section we will now pretend we are spammers wishing to fool a spam checking system based on Naïve Bayes into classifying a spam e-mail as ham (i.e. a valid e-mail – see: https://blog.barracuda.com/2013/10/03/ham-v-spam-whats-the-difference/). We will now use all of the training data to train our classifier and apply the learnt classifier to a dedicated test set.

    a. Load the test set in Weka.

    b. Under the Classify tab, select supplied test **set > set > open file** and set the test file to the supplied *spambase_test.arff*. This ARFF file contains the binary vector representing one spam e-mail.

    c. **Run** the Naïve Bayes classifier on this test set. Does the classifier classify the spam e-mail correctly?

5. Open the test file spambase_test.arff in text editor. Identify good non-spam words and add these to the e-mail. Important: Leave the class label (last attribute value) in the test data file untouched. During testing, Weka will ignore this attribute and will instead use our previously trained classifier to predict the class label of this e-mail. Re-run the classifier on the modified test set. Has the class label (spam/non-spam) for this e-mail changed?

    You've now managed to switch the predicted class label for that e-mail. Adding more “hammy” words to this e-mail has sufficiently increased the probability that this e-mail is ham so the classifier now outputs "ham" as the e-mail's class label (by changing the word content of the e-mail you have added extra evidence or "votes" towards this e-mail being classified as ham). This is the "stuffing" example given in the lectures and is directly caused by the independence assumption that is made by Naive Bayes. Each word contributes independently of each other to the final score. This is a reason that a lot of spam e-mails include random excerpts from the passages of books so as to effectively add “ hammy” words in the hope that the spam e-mail will bypass the spam filters. For this reason, in practice, many commercial e-mail systems (consider Gmail) likely use a lot more sophisticated spam detection models.