Nearest Neighbor, Bayes and Naive Bayes with PYTHON

Objective: In exercise 5 and 6 you learned how to use decision trees and logistic regression for classification. The objective of today's exercise is to understand how k-nearest neighbor, naïve Bayes can be used to solve classification problems.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* C10, C11 as well as the files in the exercise 7 folder available from Campusnet.

Preparation: Exercises 1-6

Part 1: Group discussion (max 15 min)

For the group discussion, each group should have selected a *discussion leader* at the previous exercise session. The purpose of the discussion leader is to ensure all team members understands the answers to the following two questions:

Multiple-Choice question: Solve and discuss **problem 11.1** from chapter 11 of the lecture notes. Ensure all group members understand the reason why one of the options is true and why the other options can be ruled out.

Discussion question: Discuss the following question in the group

- Consider the height/weight example from the slides. Suppose we only considers a classifier which uses height to classify the sex, but that we consider datasets where the height-attribute has been copied from 1 to 100 times. write up the Naïve-Bayes approximation in this setting and explain how the copied attributes affects the predictions made by the Naïve-Bayes classifier.

Part 2: Programming exercises

Piazza discussion forum: You can get help by asking questions on Piazza: `https://piazza.com/dtu.dk/fall2016/02450`

Software installation: Extract the Python toolbox from Campusnet. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/`

Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.matrix	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.matrix	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.matrix	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`*_train`	—	—	Training data.
	`*_test`	—	—	Test data.

Todays exercise is split in two sections. An exercise on k-nearest neighbor classification and an exercise on naïve Bayes.

7.1 K-nearest neighbor classification

In this exercise we will use the k-nearest neighbors (KNN) method for classification. First, we will consider 4 different synthetic datasets, that can be loaded into Python using the `loadmat` function. The data is stored in files `Data/synth1`, ..., `Data/synth4`.

7.1.1 Consider the script `ex7_1_1.py`. For each of the four synthetic datasets, do the following. Load the dataset into Python and examine it by making a scatter plot. Classify the test data `X_test` using a k-nearest neighbor classifier. Choose a distance measure (consider the following distance measures: `euclidean`, `cityblock`). Choose a suitable number of neighbors. Examine the accuracy and error rate.

Script details:

- *The Python class* `KNeighborsClassifier` *from* `sklearn\neighbors` *module can be used to perform k-nearest neighbors classification.*
- *To generate a confusion matrix, you can use the function* `confusion_matrix()` *function from module* `sklearn.metrics` *in the course toolbox. You can use* `imshow()` *function to plot the confusion matrix.*

Which distance measures worked best for the four problems? Can you explain why? How many neighbors were needed for the four problems? Can you give an example of when it would be good to use a large/small number of neighbors? Consider e.g. when clusters are well separated versus when they are overlapping.

In general we can use cross-validation to select the optimal distance metric and number of nearest neighbors k although this can be computationally expensive. We will return to the Iris data we have considered in previous exercises, and attempt to classify the Iris flowers using KNN.

7.1.2 Consider the script `ex7_1_2.py`. The script loads the Iris data into Python. Explain how the script uses leave-one-out crossvalidation to estimate the number of neighbors, k, for the k-nearest neighbors classifier and plots the crossvalidated average classification error as a function of k for $k = 1, \ldots, 40$.

Script details:

- *To load the Iris data, you can run your solution to exercise 4.1.1.*
- *Use* `LeaveOneOut` *crossvalidation from module* `sklearn.cross_validation`.
- *As before, use the* `KNeighborsClassifier` *class for k-nearest neighbors classification.*

7.1.3 Discussion: What are the benefits and drawbacks of K-nearest neighbor classification and regression compared to logistic regression, decision trees and linear regression? (Hint: There are two important aspects of classification and regression methods, how well the methods can *predict* unlabeled data and how well the method *describe* what aspects in the data causes the data to be classified a certain way .)

7.2 Naïve Bayes

In this part of the exercise we will classify names as female or male names `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/`, for further details see also the readme_male_female.txt file in the Data folder. We have a database with 2943 male names and 5001 female names. We will only consider names that contain at least four letters resulting in a total of 2785 male and 4866 female names. As feature for the classification we will use the first and second letter as well as the second last and last letter of the names denoted respectively x_1, x_2, x_3 and x_4. Thus the name "Richard" will have $x_1 = r$, $x_2 = i$, $x_3 = r$, $x_4 = d$. Therefore, $x_i \in \{a, b, c, d, \ldots, z\}$ such that each feature can take the value of any of the 26 letters of the alphabet (from a to z) hence the attributes used are discrete/categorical. In Python, we code each letter as a numbers between 1 and 26 where 1 corresponds to a and 26 to z.

According to Bayes rule we have

$$P(Y|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|Y)P(Y)}{P(\boldsymbol{X})}, \text{ where } P(\boldsymbol{X}) = \sum_{c=1}^{C} P(\boldsymbol{X}|y_c)P(y_c).$$

where C is the total number of classes, i.e. for the names data $C = 2$, i.e. $y_1 =$ Female and $y_2 =$ Male. For our classification of names as female or male names we find

$$\begin{aligned} P(\text{Female}|x_1, x_2, x_3, x_4) &= \frac{P(x_1,x_2,x_3,x_4|\text{Female})P(\text{Female})}{P(x_1,x_2,x_3,x_4)} \\ P(\text{Male}|x_1, x_2, x_3, x_4) &= \frac{P(x_1,x_2,x_3,x_4|\text{Male})P(\text{Male})}{P(x_1,x_2,x_3,x_4)}, \end{aligned}$$

where

$$P(x_1, x_2, x_3, x_4) = P(x_1, x_2, x_3, x_4|\text{Female})P(\text{Female}) + P(x_1, x_2, x_3, x_4|\text{Male})P(\text{Male}).$$

We will classify a name as a female name if $P(\text{Female}|x_1, x_2, x_3, x_4) > P(\text{Male}|x_1, x_2, x_3, x_4)$ and as a male name otherwise. We will split the data into a training and a test set and build our classifier using the training data, i.e.

$$\begin{aligned} P(x_1, x_2, x_3, x_4|\text{Female}) &= \frac{N_{\text{train}}^{\text{Female}}(x_1,x_2,x_3,x_4)}{N_{\text{train}}^{\text{Female}}} \\ P(x_1, x_2, x_3, x_4|\text{Male}) &= \frac{N_{\text{train}}^{\text{Male}}(x_1,x_2,x_3,x_4)}{N_{\text{train}}^{\text{Male}}}, \end{aligned}$$

where $N_{\text{train}}^{\text{Female}}(x_1, x_2, x_3, x_4)$ and $N_{\text{train}}^{\text{Male}}(x_1, x_2, x_3, x_4)$ denotes respectively the number of female and male names with first letter x_1, second letter x_2, second last letter x_3 and last letter x_4 in the training data while $N_{\text{train}}^{\text{Female}}$ and $N_{\text{train}}^{\text{Male}}$ is the total number of female and male names in the training data. $P(\text{Female})$ and $P(\text{Male})$ is the so-called prior beliefs that a name is a male or a female name. Often this is either selected as a uniform prior, i.e. $P(\text{Female}) = P(\text{Male}) = 0.5$ or estimated from the empirical distribution of the training data given by

$$\begin{aligned} P(\text{Female}) &= \frac{N_{\text{train}}^{\text{Female}}}{N_{\text{train}}^{\text{Male}} + N_{\text{train}}^{\text{Female}}}, \\ P(\text{Male}) &= \frac{N_{\text{train}}^{\text{Male}}}{N_{\text{train}}^{\text{Male}} + N_{\text{train}}^{\text{Female}}} = 1 - P(\text{Female}). \end{aligned}$$

7.2.1 In order to classify names as male or female according to the above we need to count the number of times given letter combinations occurred in the male and female names in the training data, i.e. how many times each of the letter combinations $(x_1, x_2, x_3, x_4) = (a, a, a, a), (x_1, x_2, x_3, x_4) = (a, a, a, b), \ldots, (x_1, x_2, x_3, x_4) = (z, z, z, z)$ occurred. How many different letter combinations do we have to evaluate?

7.2.2 How well do you think we can identify the probabilities $P(x_1, x_2, x_3, x_4|\text{Male})$ and $P(x_1, x_2, x_3, x_4|\text{Female})$ from the data at hand? Consider how likely it is that a given dataset will contain enough training samples to allow us to accurately estimate these probabilities.

Rather than finding the full joint distribution $P(\boldsymbol{X}|Y)$ we will use a Naïve Bayes classifier instead that assumes that each attribute (letter in a name) is independent, thus

$$P(Y|\boldsymbol{X}) = \frac{P(Y)\prod_i P(X_i|Y)}{P(\boldsymbol{X})}, \text{ where } P(\boldsymbol{X}) = \sum_{c=1}^{C} P(y_c)\prod_i P(X_i|y_c).$$

From the training data we can obtain

$$\begin{aligned} P(X_i = x_t|\text{Male}) &= \frac{N_{\text{train}}^{\text{Male}}(X_i = x_t)}{N_{\text{train}}^{\text{Male}}} \\ P(X_i = x_t|\text{Female}) &= \frac{N_{\text{train}}^{\text{Female}}(X_i = x_t)}{N_{\text{train}}^{\text{Female}}} \end{aligned}$$

where $N^{\text{Male}}(X_i = x_t)$ is the number of times the letter x_t occurred in the i^{th} considered letter of the name. Using Naïve Bayes we only need to estimate the probability of observing each of the 26 letters for each of the four considered position in the spelling of the name separately. From more than one thousand male and female names we can quite accurately estimate these probabilities.

7.2.3 Inspect and run the script `ex7_2_3.py`. The script loads the male and female names into Python, filters the text data and extracts the first two and last two letters of each name as feature.

7.2.4 Inspect and run the script `ex7_2_4.py`. The script is used to classify the names using a naïve Bayes classifier. Use a uniform prior, assuming male and female names are equally likely (i.e. $P(\text{Male}) = P(\text{Female}) = 0.5$). Compute the classification error using 10-fold crossvalidation.

Script details:

- *Type* `help(sklearn.naive_bayes.MultinomialNB)` *to learn how to use naive bayes framework in Python.*
- *As usual, use* `sklearn.cross_validation` *module to set up the crossvalidation partitions.*

Try classifying names based on only one of the letters in the name. You can do this by writing, e.g., `X=X[:,0]` to choose the first letter. Show that the last letter is most useful for classifying names. Can you explain why?

7.2.5 Change the prior to `empirical` such that $P(\text{Female}) = \frac{N_{\text{train}}^{\text{Female}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}}$, $P(\text{Male}) = \frac{N_{\text{train}}^{\text{Male}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}} = 1 - P(\text{Female})$. Does this setting improve the classification and if so why?

7.3 Tasks for the report

Classify your data by KNN and Naive Bayes. For KNN you can for instance use cross-validation to select the number of nearest neighbors. (notice, KNN analyze nominal variables, i.e. categorical variables, using one-out-of-K coding).

References