**CS448 - Project 3**

**K-Nearest Neighbor Classification of Cloud Data**

Credit: 5 points

Due date: 04/23/2014

**Project Description:**

In this project, you will use a simple data mining algorithm (k-nearest neighbor classifier) for categorization of iris plants into three classes. The project has two goals:

1. Using the Amazon Relational Database Service (RDS) to obtain data from a public cloud service
2. Building a simple k-nearest neighbor (KNN) classifier to categorize instances in a given dataset.
3. Evaluating the effect of neighbor set size and training set size on the accuracy of the KNN classifier built.

**Preliminaries:**

1. **KNN Classification:**

A k-nearest neighbor classifier is a simple data classifier, where the category (class label) of a data item is determined by using a majority vote of its neighbors, assigning the data item to the category most common among its *k* nearest neighbors. The input to the algorithm is a set of tuples, where one of the attributes is a class label and the other attributes are the features of the given data. While some of the tuples have known class labels, others have unknown class labels, and the task is to label those tuples with unknown class labels. The class label for any tuple *t* is determined with the following procedure:

1. Measure the distance of the tuple *t* to each of the labeled tuples in the dataset.

2. Find the set *S* of the k-nearest neighbors of *t* (i.e. *k* tuples with the smallest distance to *t*).

3. Find the majority class label in the set *S*. If two or more classes have the majority (i.e. the same number of occurrences in the set), decrease *k* by 1 until you find the majority class label (Note that you decrease k only for the classification of this instance, not the whole dataset).

Although various distance metrics could be used to measure the distance between two data items in KNN, the metric you will use in this project is the Euclidean distance. Given two tuples *T1* and *T2* with *n* features (where T1x represents the xth feature value of T1), the Euclidean distance between T1 and T2 is calculated as follows:

$\sqrt{\left(T1\_{1}-T2\_{1}\right)^{2}+\left(T1\_{2}-T2\_{2}\right)^{2}+\left(T1\_{3}-T2\_{3}\right)^{2}+…+ \left(T1\_{n}-T2\_{n}\right)^{2} }$

1. **Dataset:**

The dataset you will use in this project consists of data for 3 different types of iris plants. Each tuple in the dataset consists of the following 7 attributes:

* item\_no: integer (primary key, used to identify tuples, has no effect on the classification)
* sepal\_length: real
* sepal\_width: real
* petal\_length: real
* petal\_width: real
* class\_label: string (one of the three classes ‘Iris-setosa’, ‘Iris-versicolor’ or ‘Iris-virginica’)
* label\_known: integer (the value of this attribute is 0 if the class label of the tuple is not known, and 1 if it is known)

When running your KNN classifier, you should consider the items with a *label\_known* attribute value of 1 as your training set (i.e. items with a known label), and try to find the class labels of the tuples which have a *label\_known* attribute value of 0. Note that the class label for each tuple has to be one of ‘Iris-setosa’, ‘Iris-versicolor’ and ‘Iris-virginica’, as specified above. You should only use the values of the attributes sepal\_length, sepal\_width, petal\_length and petal\_width in calculating the Euclidean distance between any two tuples during classification.

1. **Connecting to Amazon RDS:**

The dataset you will use in this project resides in an Oracle database on Amazon’s Relational Data Services (RDS) cloud. Your KNN classification program should connect to this database using a database connection library (JDBC) to retrieve the tuples in the dataset. Note that you will use this database in a read-only manner, i.e. you should not try to modify any tuples or table in the database. The data is stored in a table named “IRIS”, with the attributes specified in the above section.

To get started using JDBC, you can go through the tutorial at <http://www.tutorialspoint.com/jdbc/jdbc-quick-guide.htm>. This level of knowledge will be sufficient for this project, as the only thing you need to do is to connect to the database at RDS and retrieve the data. You can also see other sample code at <https://www.cs.purdue.edu/homes/bb/cs448/StudentDB.pdf>. Note that when registering the database driver in your code, you should pass the argument “oracle.jdbc.driver.OracleDriver” to the Class.forName() method, as you will be connecting to an Oracle database. Also, you will need to include the JDBC driver [**ojdbc6.jar**](http://cs.purdue.edu/homes/bb/cs448/ojdbc6.jar) in your classpath at runtime to be able to use the database connectivity functions.

You should use the following information to connect to the database on RDS:

Connection string (on a single line, with no spaces):

jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=cs448db.cdo5tavfgia0.us-east-1.rds.amazonaws.com)(PORT=1521))(CONNECT\_DATA=(SID=irisdb)))

username: your career account username

password: your career account **username**

You should use the above parameter values with the *getConnection* method of the *DriverManager* class when connecting to the database. Note that when accessing the IRIS table in the database, you should access it as *cs448admin.IRIS*, as this is a table owned by the cs448admin account. For example, if you want to list all records in the table, you should use the following query:

*SELECT \* FROM cs448admin.IRIS;*

**Program Execution:**

You should name your program file KNN.java. Your program should accept two command line arguments: *k* and *num\_training*, where *k* is the neighbor set size in KNN and *num\_training* is the number of training examples to use for classification. Because the item\_no attribute will assume values starting at 1 and increasing by 1 for each subsequent tuple, if *num\_training* is set to 100, you should consider the tuples with item\_no <=100 as the training set. This means that the labeled set of items you consider when classifying items with unknown label is based on *num\_training*, but the items to be classified will be a fixed set (i.e. you should disregard all items with item\_no > num\_training, if their *label\_known* attribute value is 1).

We should be able to run your program using the following command:

*java –cp ojdbc6.jar:. KNN k num\_training*

where k and num\_training will be positive integers.

Your program should output the predicted class label for each item with label\_known = 0, one line per item in the following format:

101 Iris-setosa

102 Iris-virginica

103 Iris-setosa

…

 where the first value on the line is item\_no followed by a tab character and the value after the tab is the class label predicted for that item. Note that the items should be **SORTED** in increasing order on item\_no. The last line of the output should be a single decimal stating the accuracy of your classifier calculated as follows:

*accuracy = number of correct predictions / total number of predictions*

In order to find the number of correct predictions, for each tuple classified by the algorithm, you should compare the class label predicted by the algorithm with the true class label of the item as specified by the class\_label attribute of that item.

**Algorithm Evaluation:**

You should run your program with the following parameter values and observe the effects of *k* and *num\_training* on the accuracy of the classifier:

k = 1, num\_training = 111

k = 4, num\_training = 111

k = 5, num\_training = 111

k = 7, num\_training = 111

k = 3, num\_training = 111

k = 3, num\_training = 75

k = 3, num\_training = 60

Write a paragraph about your observations in a file named evaluation.txt.

**Submission Instructions:**

Please create a README file containing identifying information. For example:

CS448 - Project 3

Author: John Doe

Login: jdoe

Email: jdoe@cs.purdue.edu

Include here anything you might want us to know when grading your project.

To turn in your project, ssh to lore.cs.purdue.edu, create a folder named project3 in your home directory and copy KNN.java, evaluation.txt and your README file to that folder.

After copying your files in the folder project3, execute the following command in your home directory:

turnin –c cs448 –p proj3 project3

To verify the contents of your submission, execute the following command right after submission:

turnin –c cs448 –p proj3 -v