Design Overview for basic_k-NN

Name: Chris Dilger
Student ID: 101133703

Summary of Program

The program is an example of a foundational machine learning algorithm, variants of which have been used to classify Gastric Cancer, Wine Quality and Flower species (Li et al. 2012; Lichman 2013). In this implementation, the k-Nearest-Neighbour search will classify flowers to demonstrate practical uses. This implementation is general enough to give users the freedom to a multitude of other datasets provided the input can be parsed by the CSV parser. This algorithm will be tested against the Iris dataset from the UCI Machine Learning Repository.

The complexity of the implementation is completely invisible to the end user. A user supplies data in a form analogous to CSV, which is then parsed by the k-NN search program. A user is then able to input key parameters which allow the k-NN to perform a classification on the object, returning a classification using the input data.

Example of input data

Input (iris.csv)

```
5.1,3.5,1.4,0.2,Iris-setosa
…
7.0,3.2,4.7,1.4,Iris-versicolor
…
6.3,3.3,6.0,2.5,Iris-virginica
…
```

Where the fields represent:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. classification

(Lichman 2013)

Given input data, say:

```
6.0, 2.7, 4.0, 1.3
```

we would expect an output:

```
Iris-versicolour
```

Data Dictionary

Description of the structures and custom data types used in the knn-search program.

Table 1: Classifier_List details

Field Name	Type	Notes
categories	my_string*	Stores the strings corresponding to each distinct category. This is an array type
num_categories	Int	Keep track of the number of categories in this list of categories

Table 2: Point details

Field Name	Type	Notes
dimension	float*	Keeps all of the floating point coordinates for the dimensions in each point. The number of dimensions in this array is dependent on the number of dimensions in the dataset.
category	int	Smallest possible way to represent the class of a point

Table 3: Dataset details

Field Name	Type	Notes
dimensionality	int	The number of dimensions the dataset is tracking. This would be 2 for a simple dataset with 2 variables, and for more complex datasets this could be 4, as for the Iris dataset.
num_points	int	Keep a record of the length of the array of points
Points	Point*	An array containing all of the points in the dataset

Table 5: Point_Neighbour_Relationship details

Field Name	Type	Notes

distance	float	The distance between the Comparision_Point to which this belongs, and the point this relationship is pointing to
neighbour_pointer	Point*	Reference to the neighbour it's pointing to

Table 6: Boolean enumeration details

Type	**Enumerated Values**	**Notes**
bool	false, true	Allows us to use true and false as it would be in other languages, as well as having it's own datatype

Overview of Program Structure

Menu

A basic menu is implemented so that the user can interact with the program. A file can be read in with training data, and points to be classified can be entered by the user.

Euclidean distance

A function that will perform the core operation inside of the k-NN algorithm. From (Anon n.d.), the following algorithm will be implemented:

$$D_{ij}^2 = \sum_{v=1}^{n} \left(x_{vi} - x_{vj}\right)^2$$

This will involve some simple summation, some for looping and accessing members of arrays. This may take some time for very large datasets.

Sorting algorithm

To calculate the mode of the categories a quicksort is performed on the k nearest neighbours. Pointers to the points which are nearest neighbours are used to read this data from memory.

Mode Calculation

The mode of a set of data needs to be calculated as part of the voting process. This doesn't take into account the case in which there are an even number of data points.

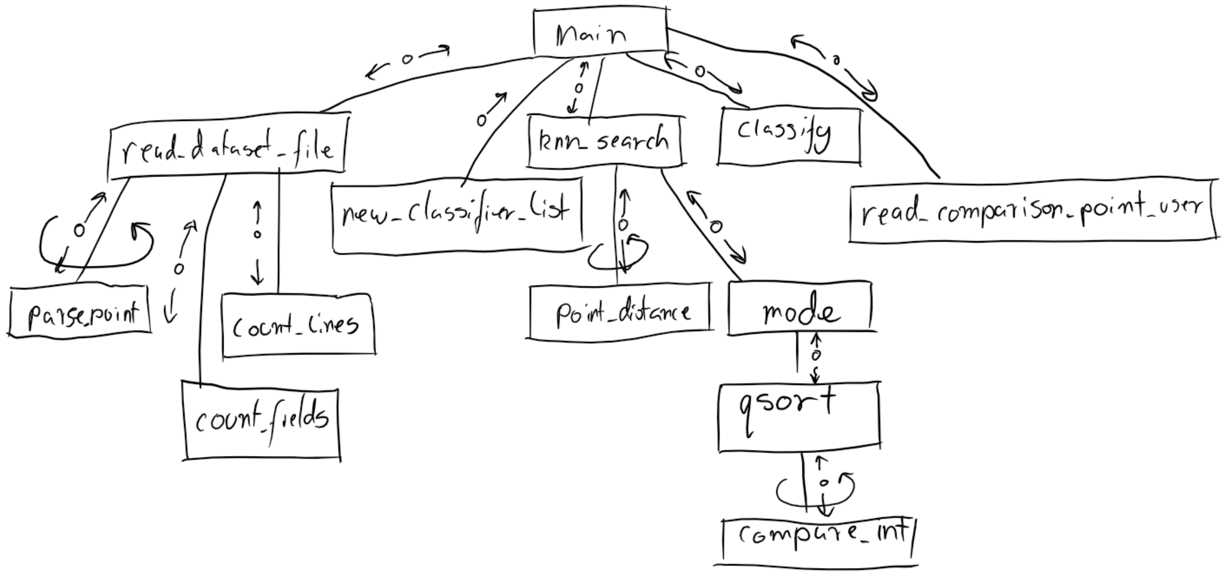
Parsing lines of CSV data

Using strtok to tokenise a buffer input, and then return usable data. File IO makes loading files very easy for the user because all information about setting array sizes happens without user interaction.

Unit tests

A unit testing library greatest.h (Vokes 2017) provides a minimalistic framework with which Test Driven Development will be practiced. This allows unit tests to be run to ensure that the minimum and correct functionality can be achieved at a functional level.

Structure Chart



References

Anon n.d., ‘A Complete Guide to K-Nearest-Neighbors with Applications in Python and R’, viewed 8 May, 2017a, <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>.

Anon n.d., ‘c++ - how to completely disable assertion - Stack Overflow’, viewed 25 May, 2017b, <https://stackoverflow.com/questions/5354314/how-to-completely-disable-assertion>.

Anon n.d., ‘INFO: strtok(): C Function -- Documentation Supplement’, viewed 31 May, 2017c, <https://support.microsoft.com/en-us/help/51327/info-strtok-c-function----documentation-supplement>.

Anon n.d., ‘“n”-Dimensional Euclidean Distance’, viewed 8 May, 2017d, <https://hlab.stanford.edu/brian/euclidean_distance_in.html>.

Anon n.d., ‘“Re: Regression testing in OpenBSD” - MARC’, viewed 25 May, 2017e, <http://marc.info/?l=openbsd-ports&m=139474670315494>.

Datar, M, Immorlica, N, Indyk, P & Mirrokni, VS 2004, 'Locality-sensitive hashing scheme based on p-stable distributions', *Proceedings of the twentieth annual symposium on Computational geometry*, ACM, pp. 253–262, viewed 24 May, 2017, <http://dl.acm.org/citation.cfm?id=997857>.

Dong, W, Moses, C & Li, K 2011, 'Efficient k-nearest neighbor graph construction for generic similarity measures', *Proceedings of the 20th international conference on World wide web*, ACM, pp. 577–586, viewed 19 May, 2017, <http://dl.acm.org/citation.cfm?id=1963487>.

Li, C, Zhang, Shuheng, Zhang, H, Pang, L, Lam, K, Hui, C & Zhang, Su 2012, 'Using the K-Nearest Neighbor Algorithm for the Classification of Lymph Node Metastasis in Gastric Cancer', *Computational and Mathematical Methods in Medicine*, vol. 2012, p. e876545.

Lichman, M 2013, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, viewed <http://archive.ics.uci.edu/ml>.

Vokes, S 2017, *greatest: A C testing library in 1 file. No dependencies, no dynamic allocation. ISC licensed*, C, viewed <https://github.com/silentbicycle/greatest>.