CSE 176 / EECS 276
Machine Learning
Specification for Programming Assignment #2

David C. Noelle

Due: 11:45 P.M. on Sunday, March 3, 2013

Overview

The primary learning goal of this programming assignment is to provide you with a more intimate understanding of the functional requirements of instance-based machine learning algorithms, in general, and of the k-nearest-neighbor algorithm, in particular. Your efforts to implement the specified program are intended to expose you to issues surrounding the storage and manipulation of feature vectors, the use of a set of such vectors to generate responses to novel inputs, and the patterns of generalization produced by nearest neighbor approaches.

In order to meet this goal, you will write a program that implements the k-nearest-neighbor algorithm in a fairly general way. Your program will support real valued input feature vectors with arbitrary dimensionality, real valued output feature vectors with arbitrary dimensionality, and flexible values for the "k" parameter. Your program will be able to use both Euclidean distance and angular distance as similarity measures. It will be able to produce output vectors that are the unweighted mean of the current input's "k" neighbors, and also the mean weighted by similarity. Once your program is complete, you will demonstrate its application on a provided set of training and testing patterns, and you will identify the optimal value of "k" for these pattern sets.

Submission for Evaluation

In order to submit your solution to this assignment for evaluation, you must generate a GNU Zipped TAR archive file, as described below, containing all of your source code and an assignment report. This archive file must then be delivered to the teaching team through the UCMCROPS system prior to the assignment due date.

The submitted archive file must unpack into a directory (i.e., "folder") called "`P2`". This directory must contain an assignment report file in PDF format named "`report.pdf`". The "`P2`" directory should also contain a subdirectory called "`code`", which should contain all of the source code necessary to compile and execute your program. (The primary exception to this requirement

is that the "`code`" subdirectory need *not* contain standard libraries, including the *GNU Scientific Library*.) Program building and configuration files, such as "makefiles", are appreciated but are not required. The assignment report file should contain:

- a brief description of the program that was generated, including the programming language that was used, any interesting libraries or other tools that were used, instructions for compilation, and instructions for execution
- a trace of the program running, performing the operations described below on a training set and testing set that will be provided to you
- a graph of the testing set error as a function of the "k" parameter for the training set and the testing set that will be provided to you
- a brief summary of what was learned from this programming assignment
- a listing of acknowledgments and references, appropriately citing *all* assistance that was received on this project

This report should be clear and easy to read, and it must be provided as a PDF file.

In order to generate the required archive file on a Unix or Linux computer, such as those available in the class laboratory, simply construct a directory called "`P2`" with the contents described above and then execute the following command from the directory containing "`P2`":

```
$ tar -czvf P2.tgz P2
```

This will generate a GNU Zipped Tape Archive (TAR) file called "`P2.tgz`". This file is to be submitted for evaluation using the UCMCROPS system.

To submit this archive file, log onto the course `UCMCROPS` website, and navigate to the "Assignments" section. Then, locate "Programming Assignment #2" and select the option to submit your solution for this assignment. Provide your archive file as an attachment to this submission. Comments to the teaching team should appear in your report document and as inline comments in your source code.

Submissions must arrive by 11:45 P.M. on Sunday, March 3rd. Please allow time for potential system slowness immediately prior to this deadline. You may submit assignment solutions multiple times, and only the most recently submitted version will be evaluated. (If there are difficulties with performing multiple submissions, make use of the "Drop Box" feature on UCMCROPS to submit subsequent versions *and* send a note to the instructor indicating that you have done so.) As discussed in the course syllabus, late assignments will not be evaluated.

If your last submission for this assignment arrives by 11:45 P.M. on Friday, March 1st, you will receive a 10% bonus to the score that you receive for this assignment. This bonus is intended to encourage you to try to complete this assignment early.

Activities

You are to provide a computer program that implements the k-nearest-neighbor algorithm. Specifically, your program is to read a training set of patterns from a file and a testing set of patterns from an additional file. Using only the training patterns, the program should produce an output for each of the testing set patterns, and it should report this output vector for each pattern, along with a measure of output error.

Your program should take a single input, either on the command line or entered at the terminal upon execution. This input should be the name of a plain text "configuration" file that contains a description of the current run. Your program should proceed to open this configuration file and read its contents. It should then use the read information to prepare a k-nearest-neighbor system with the specified properties. Each configuration file is expected to contain the following fields, in this order, in plain text format, with one field per line:

1. k — a whole number indicating the number of neighbors to consider
2. d_{in} — the dimensionality of the input vectors, as a whole number
3. d_{out} — the dimensionality of the output vectors, as a whole number
4. distance metric — a string starting in "E" indicates that Euclidean distance is to be used, while a string starting with "A" indicates that angular distance is to be used
5. output method — a string starting with "U" or "M" indicates that the unweighted mean of the output vectors of the k neighbors is to be used as the output vector, while a string starting with "W" indicates the distance weighted mean of the output vectors is to be used
6. N_{train} — the number of patterns in the training set
7. training set file name — the name of a file containing the set of training patterns to be used
8. N_{test} — the number of patterns in the testing set
9. testing set file name — the name of a file containing the set of testing patterns to be used
10. output file name — the name of the output file, which should be created and filled with plain text output

Given this configuration information, your program should read the set of training patterns and testing patterns. Using the specified value of k, the specified distance metric, and the specified output method, your program should then produce an output vector for each input vector in the testing set. It should open a new plain text file with the given output file name, and it should write one line to this file for each of the testing set patterns. Each output line should contain a row of real numbers, consisting of:

1. input vector — the current test pattern input vector should be written first on the line

2. output vector — the output vector produced for the current test pattern by the k-nearest-neighbor algorithm should appear next on the line

3. target vector — the "target" output vector for the current test pattern, as specified in the testing set file, should appear next on the line

4. SSE — the sum squared error between the output vector and the target vector for the current test pattern should be provided as a single scalar value at the end of the line

Thus, the output file will contain one line for each testing set pattern. One additional line should be appended to this output file, however, and this last line should contain a single real value: the sum of the SSE values over all of the patterns in the testing set. This last statistic can be used to compare performance on a given training and testing set across, for example, different values of k.

Recall that the sum squared error (SSE) for a given testing set pattern is calculated as follows. If $\vec{y}$ is the output vector produced by the k-nearest-neighbor algorithm and $\vec{t}$ is the target vector for this pattern, as specified in the testing set file, then the SSE value is:

$$SSE \;=\; \sum_{i=1}^{d_{out}} (y_i - t_i)^2$$

The SSE value reported at the end of the output file is simply the sum of these pattern-specific SSE values over all N_{test} testing set patterns.

The pattern files to be read by your program — both the training set file and the testing set file — are to be plain text files that contain one pattern on each line of the file. Thus, each line will contain a sequence of real numbers separated only by whitespace. The first d_{in} numbers on each line form the input vector for that pattern, and the remaining d_{out} numbers form the corresponding output vector. For each pattern in the training set, the output vectors need to be recorded so that they can be used by the k-nearest-neighbor algorithm. For each pattern in the testing set, the output vectors are provided to allow you to assess how well your learning algorithm generalizes.

When the configuration file indicates that the distance weighted mean of the output vectors is to be used in order to construct outputs for novel input vectors, the strategy outlined in Equation 8.3 and Equation 8.4 of Mitchell (1997) should be used. Specifically, the output vectors of the neighbors should be averaged, with each output vector weighted by the inverse square of the distance between the corresponding input vector and the query vector. (The same distance measure used to identify the nearest neighbors should be used: either Euclidean distance or angular distance, as specified by the configuration file.) In particular, if $\vec{y}_i$ is the output vector corresponding to the training set neighbor with input vector $\vec{x}_i$, then, for query vector $\vec{x}_q$, the output should be computed as ...

$$\vec{y}_q \;=\; \frac{\sum_{i=1}^{k} w_i \vec{y}_i}{\sum_{i=1}^{k} w_i}$$

...where the weights are defined to be related to the distance between the input vectors in an inverse square fashion:

$$w_i \;=\; \frac{1}{d(\vec{x}_q, \vec{x}_i)^2}$$

As described in Mitchell (1997), a special case arises if there is a zero distance between the query vector and any of the k neighbor input vectors. In this special case, the output should simply be the unweighted average of all of the output vectors with corresponding neighboring input vectors that are exactly equal to the query vector.

The source code that you submit for evaluation should be of your own design. You may use openly available standard libraries in order to construct your program, however. In particular, if you are writing your program in C or C++, you are encouraged to make use the rich collection of tools provided by the *GNU Scientific Library*. This library may be found at:

`www.gnu.org/software/gsl/`

This web site also contains extensive documentation describing the functionality provided by the library.

You will be provided with a training set file and testing set file, called "`glass-train.dat`" and "`glass-test.dat`", respectively. These will be made available to you through the UCM-CROPS system. Once you have implemented your program, you may use these files to test your code. Additionally, you are expected to exercise your program on these pattern sets and briefly describe what you observe in the written report that you are required to submit with your source code. In particular, you should:

- try various configurations, involving different distance measures and the use of weighted and unweighted output means, reporting how these configuration changes affect the performance of your program on the given problem
- for the best distance measure and method of producing an output, construct a graph that relates the value of the k parameter to the resulting testing set error
- given your observations, suggest an optimal value of k for this problem

These analyses should be considered minimal. You are welcome to exercise your program in additional ways and report your observations.

Your implementation will be evaluated primarily for accuracy, with efficiency being a secondary consideration. (In particular, you are *not* expected to implement data structures to efficiently index training vectors at the time that they are read. Training patterns may be stored in a simple matrix format or in a list, allowing for linear complexity of access.) Your source code *will* be examined, however, and the readability and style of your implementation will have a substantial influence on how your assignment is evaluated. As a rough rule of thumb, consider the use of good software writing practices as accounting for approximately 10% of the value of this exercise. Note also that, as discussed in the course syllabus, submissions that fail to successfully compile will not be evaluated and will receive *no credit*.

As for all assignments in this class, submitted solutions should reflect the understanding and effort of the individual student making the submission. Not a single line of computer code should be shared between course participants. If there is ever any doubt concerning the propriety of a given interaction, it is the student's responsibility to approach the instructor and clarify the situation *prior* to the submission of work results. Also, helpful conversations with fellow students, or any other

person (including members of the teaching team), should be explicitly mentioned in submitted assignments (e.g., in comments in the submitted source code files). Failure to appropriately cite sources is called *plagiarism*, and it will not be tolerated!

The members of the teaching team stand ready to help you with the learning process embodied by this assignment. Please do not hesitate to request their assistance.