MACHINE LEARNING FOR ROBOTICS I - 86928

Lab 3: kNN classifier

- Task 1: Obtain a data set
- Task 2: Build a kNN classifier
- Task 3: Test the kNN classifiers

Task 1: Obtain a data set

Download the mnist data set. The zipfile contains, in separate files, a training set, corresponding labels, a test set, corresponding labels, two Matlab functions to load data and labels, and one flexible function to load the data named `loadMNIST`.

The first two functions are there just because they are distributed along with the data. For this lab assignment you don't need them; you will just use `loadMNIST`. Read its synthetic documentation by typing:

```
help loadMNIST
```

The data represent handwritten digits in 28x28 greyscale images, and are a standard benchmark for machine learning tasks.

Task 2: Build a kNN classifier

Here you have to create a program (a Matlab function for instance) that takes the following parameters (input arguments):

1. a set of data, as a *n* x *d* matrix, to be used as the training set
2. a corresponding column (a *n* x 1 matrix) of targets, i.e., class labels
3. another set of data, as a *m* x *d* matrix, to be used as the test set
4. an integer *k*
5. OPTIONALLY, another set of data, as a *m* x *1* matrix, to be used as the test set ground truth (class labels)

Remark: Note that the requirement here asks for a *separate* target column for both the training set and (optionally) the test set. This is one of the many possible forms in which data may be organised.

Remark: The optional column is give as the last argument; this simplifies coding. Many programming languages allow for optional arguments or default values for arguments that were not specified when calling the function. Arguments are indicated with a name even when calling the function: for instance in Python a possible syntax is `knn(mydata = dataset)`. In this way they can be positioned in any order.

In Matlab this mechanism does not exist *per se*, but there is a convention which is to use "Name-Value pairs". These are arguments that are interpreted in pairs. The first element, always a string, is the name of the argument; the second one is the corresponding value to set. You may have used this to set some properties of graphs with the plot function. There are functions that make it easier to use this convention.

However, to simplify our lives, we choose to have a *fixed* argument ordering, and to place the optional argument last (since it is just one). In this way, we can simply check the number of argument received (built-in variable `nargin`)to see whether the last one is present or not.

The program should:

1. Check that the number of arguments received (`nargin`) equals at least the number of mandatory arguments
2. Check that the number of columns of the second matrix equals the number of columns of the first matrix
3. Check that *k*>0 and *k*<=cardinality of the training set (number of rows, above referred to as *n*)
4. Classify the test set according to the kNN rule, and return the classification obtained
5. If the test set has the optional additional column (`nargin` == n.mandatory + 1), use this as a target, compute and return the error rate obtained (number of errors / *m*)

Hint: Use the slides to implement the classifier.

Task 3: Test the kNN classifier

Use the MNIST character recognition data.

Compute the accuracy on the test set

- on 10 tasks: each digit *vs* the remaining 9 (i.e., recognize whether the observation is a 1 or not; recognize whether it is a 2 or not; ...; recognize whether it is a 0 or not)
- for several values of *k*, e.g., k=1,2,3,4,5,10,15,20,30,40,50 (you can use these, or a subset of these, or add more numbers, and you can also implement the rule: "*k* should not be divisible by the number of classes," to avoid ties).

Provide data or graphs for any combination of these parameters (e.g. recognize 1 with k=1,2,3,4...; recognize 2 with k=1,2,3,4...; and so on)

Hint: As usual, don't waste your time writing several versions of the program for each experiment; write just one script that calls the knn function in multiple ways and produces the results.

Submission status

Attempt number	This is attempt 1.
Submission status	No attempt
Grading status	Not graded
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet.

◄ Lab 2: Linear regression

Jump to...

Lab 4: neural networks ►