Cs 170: Introduction to Artificial Intelligence

(This file may get updated)

Project 2 (Feature Selection with Nearest Neighbor)

Note: We have divided this project into three parts (with separate submission deadlines)

Project Objectives

In this project you are going to better learn about:

1. Nearest Neighbor Classifier and its sensitivity to irrelevant features
2. How to do a feature search

In particular, you are going to implement:

a. Greedy search
b. The nearest neighbor classifier[1] and Evaluation using leave-one-out validation
c. Feature search using nearest neighbor classifier and the real evaluation function (leave-one-out)

1. Introduction

As discussed in the lecture, the nearest neighbor algorithm is a very simple, yet very competitive classification algorithm. It does have one major drawback however: it is **very sensitive to irrelevant features**. Therefore, we have to **choose the features very carefully**. In other words, we need to do a feature search.

In feature search, given a set of possible features $\{f_1, f_2, f_3, \ldots, f_n\}$, we try different combination of features (i.e. all possible feature subsets $\{\{\}, \{f_1\}, \{f_2\}, \ldots, \{f_1, f_2\}, \{f_1, f_3\}, \ldots\{f_1, f_2, f_3\}, \{f_1, f_3, f_4\}, \ldots\}$) and **choose the combination (subset) that yields the highest accuracy**. Remember that with a large set of features, we cannot do an exhaustive search and instead, we do a greedy (hill-climbing) search.

In this hill-climbing search (see Fig 1), each node represents a subset of features (e.g., $\{f_1, f_3, f_6\}$) and the **evaluation function** is the **accuracy** of the classifier when a particular subset of features is used (e.g., the accuracy of the nearest neighbor classifier when only features $\{f_1, f_3, f_6\}$ are used).

Greedy Forward Section

Initial state: Empty Set: No features
Operators: Add a feature.
Evaluation Function: Leave-one-out

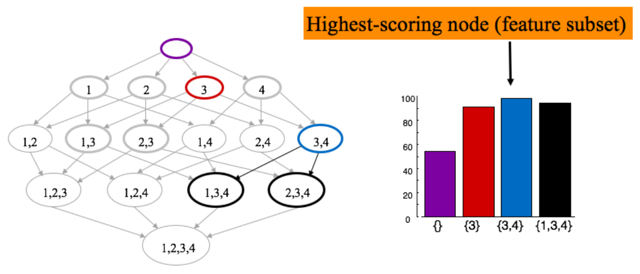


Figure 1: Greedy forward-selection feature search

[1] Very simple code, no training needed for this classifier! Just load all the training instances in memory. To classify a new unseen instance I, just find the nearest training example and report the class of that nearest neighbor as the class of I!

To compute the accuracy, you will need to use the leave-one-out validation algorithm (simpler than k-fold). However, to make debugging simpler, we are asking you to implement the evaluation function (leave-one-out validation algorithm) **later**. Instead, you will **first use a stub evaluation function (that just returns a random value)** and once your search algorithm is complete and tested, implement the actual evaluation function (leave-one-out validation).

We have divided this project into three parts (with separate submission deadlines):

Part I: Implementing the greedy search algorithms only[2] (No need to implement the classifier or the leave-one-out validator). As input, you only need total number of features (not the data file)

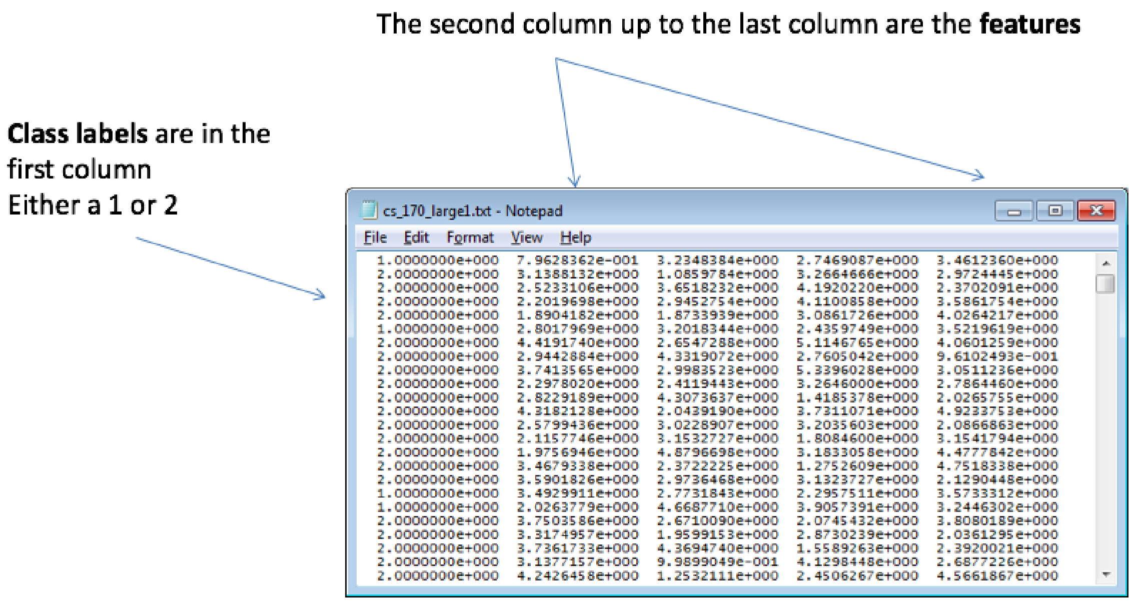
Part II: Implementing the actual evaluation function (leave-one-out validation) and the NN classifier[3] and testing it on dataset#1. (No feature search yet)

Part III: Run your greedy search algorithms on real data with the actual evaluation function, testing your completed system on the initial small and large dataset to make sure it works correctly. Then testing it on your personal small and large datasets and finishing the report.

2. Input and Output of the final System

The **input** to your system is a text file that contains a dataset with the following format (Fig 2):

- Each row is one data point (instance)
- First column is the class and all the other columns are features (f_1, f_2, f_3, …, f_n).


The second column up to the last column are the **features**
Class labels are in the first column Either a 1 or 2
cs_170_large1.txt - Notepad
File Edit Format View Help

```
  1.0000000e+000  7.9628362e-001  3.2348384e+000  2.7469087e+000  3.4612360e+000
  2.0000000e+000  3.1388132e+000  1.0859784e+000  3.2664666e+000  2.9724445e+000
  2.0000000e+000  2.5233106e+000  3.6518232e+000  4.1920220e+000  2.3702091e+000
  2.0000000e+000  2.2019698e+000  2.9452754e+000  4.1100858e+000  3.5861754e+000
  2.0000000e+000  1.8904182e+000  1.8733939e+000  3.0861726e+000  4.0264217e+000
  1.0000000e+000  2.8017969e+000  3.2018344e+000  2.4359749e+000  3.5219619e+000
  2.0000000e+000  4.4191740e+000  2.6547288e+000  5.1146765e+000  4.0601259e+000
  2.0000000e+000  2.9442884e+000  4.3319072e+000  2.7605042e+000  9.6102493e-001
  2.0000000e+000  3.7413565e+000  2.9983523e+000  5.3396028e+000  3.0511236e+000
  2.0000000e+000  2.2978020e+000  2.4119443e+000  3.2646000e+000  2.7864460e+000
  2.0000000e+000  2.8229189e+000  4.3073637e+000  1.4185378e+000  2.0265755e+000
  2.0000000e+000  4.3182128e+000  2.0439190e+000  3.7311071e+000  4.9233753e+000
  2.0000000e+000  2.5799436e+000  3.0228907e+000  3.2035603e+000  2.0866863e+000
  2.0000000e+000  2.1157746e+000  3.1532727e+000  1.8084600e+000  3.1541794e+000
  2.0000000e+000  1.9756946e+000  4.8796698e+000  3.1833058e+000  4.4777842e+000
  2.0000000e+000  3.4679338e+000  2.3722225e+000  1.2752609e+000  4.7518338e+000
  2.0000000e+000  3.5901826e+000  2.9736468e+000  3.1323727e+000  2.1290448e+000
  1.0000000e+000  3.4929911e+000  2.7731843e+000  2.2957511e+000  3.5733312e+000
  1.0000000e+000  2.0263779e+000  4.6687710e+000  3.9057391e+000  3.2446302e+000
  2.0000000e+000  3.7503586e+000  2.6710090e+000  2.0745432e+000  3.8080189e+000
  2.0000000e+000  3.3174957e+000  1.9599153e+000  2.8730239e+000  2.0361295e+000
  2.0000000e+000  3.7361733e+000  4.3694740e+000  1.5589263e+000  2.3920021e+000
  2.0000000e+000  3.1377157e+000  9.9899049e-001  4.1298448e+000  2.6877226e+000
  2.0000000e+000  4.2426458e+000  1.2532111e+000  2.4506267e+000  4.5661867e+000
```

Figure 2: Input dataset format

[2] This can be done in less than 20 lines of Matlab code for all 2 (or 3) search algorithms. A little more in c++ or Java
[3] About 8 lines of Matlab code. A little more in other languages.

The **output** of your final system is the best subset of features and its resulting accuracy (e.g., {f_1, f_4, f_{10}}, acc=0.98).

Notes

- **Think carefully before you start coding** this. Students in the past seem to have made this more complicated than it need be. In particular, in Matlab one should be able to write the nearest neighbor algorithm in 8 lines of code, and the 3 search algorithms in another 17 lines of code. C++ and Java programs tend to be longer, but even so, I would be surprised if this took more than 100 lines of code (although you won't be penalized for this).
- **Please make sure your code is as modular as possible so that one can plug-in different classifiers, validators and search algorithms.**
- You may use some **predefined utility routines, for example sorting routines**. However, I expect **all the major code to be original**. You must **document any book, webpage, person or other resources you consult** in doing this project (see the first day's handout).