Part I: Implementing the **search** algorithm only

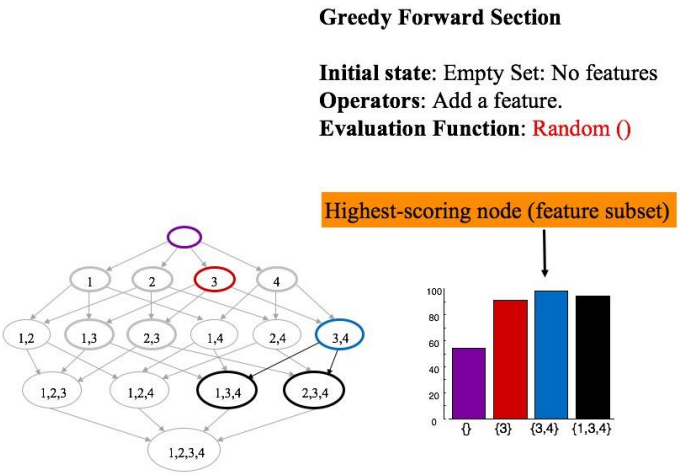

Figure 1: Greedy forward selection feature search with a dummy evaluation function that returns a random number

Code

Since feature search is an optimization problem, we are going to use a **greedy** algorithm. In this particular problem, we are trying to find the node (i.e., subset of features) that has the **maximum score**. How do we find the score of each node? Using an **evaluation function**.

The evaluation function takes a node as input and **calculates** a score for that node as the output. However, for this part of the project, you don't need to implement the actual evaluation function. Instead, you will use a stub evaluation function that returns a random value! You will implement the actual function later in part II.

Now for the search algorithm: For feature search, you are going to implement the following search methods:

1) Forward Selection
2) Backward Elimination

Don't be scared by the phrase "search algorithm". We discussed forward-selection in class (see the slides for MachineLearning2_featureSelection). **Backward-elimination is very similar: it starts with the full set of features and removes one feature at a time. Both forward-selection and backward-elimination are greedy**.

Note that at this point you don't need to read data from the file, since you are not going to do anything with data (no classification and validation yet). You only need the total number of features to do the forward–selection and backward-elimination searches. So you will have a trace like the following for forward-selection (submit the trace for backward-elimination as well). Trace example of **forward selection**:

Welcome to First Last name (change this to your name) Feature Selection Algorithm.

Please enter total number of features: **4**

Type the number of the algorithm you want to run.

1 Forward Selection
2 Backward Elimination

1

Using no features and "random" evaluation, I get an accuracy of

55.4% Beginning search.

Using feature(s) {1} accuracy is 35.4%
Using feature(s) {2} accuracy is 56.7%
Using feature(s) {3} accuracy is 41.4%
Using feature(s) {4} accuracy is 28.5%

Feature set {2} was best, accuracy is 56.7%

Using feature(s) {1,2} accuracy is 58.9%
Using feature(s) {3,2} accuracy is 40.4%
Using feature(s) {4,2} accuracy is 58.1%

Feature set {1,2} was best, accuracy is 58.9%

Using feature(s) {3,1,2} accuracy is 60.1%
Using feature(s) {4,1,2} accuracy is 76.4%

Feature set {4,1,2} was best, accuracy is 76.4%

Using feature(s) {1,2,4,3} accuracy is 73.1%

(Warning: Decreased accuracy!)
Search finished! The best subset of features is {4,1,2}, which has an accuracy of 76.4%

2
…