5/19/2020

Quang Le

**Objective**: Naming schemes for machine learning results.

**Ideas**:

* Words and numbers (What we have right now)
* Numbers only
* First letters and numbers

**Brainstorm:**

Characteristics of a naming scheme we would want:

* The rules are clear and consistent
* It needs to create a unique name for every case
* The naming scheme must contain some information about the machine learning results whether it’s the name itself or its rules

The naming scheme we have right now - dataset-algorithm-feature-tune - contains just the right amount of information to make it unique for every case. However, I think we can change the way it is represented.

**Words and numbers (What we have right now)**

Example: ESOL-rf-0-tuned

Pros

* Most intuitive -> Mostly don’t need any naming scheme info to understand.

Cons

* Slowest and Least memory efficient
* Too long -> Won’t fit very well into our graph

**Numbers Only:**

1. Random Number Generator (RNG)

Pros:

* No rules needed
* Easiest to create and apply

Cons:

* Contains no information about the machine learning results -> Every file is a mystery box that needs to be opened to know its component
* **Only apply when all else fails**

1. Apply current naming scheme:

**Rule**: Every component in *dataset-algorithm-feature-tune* has a number that corresponds to its lexicographic order in its own category.

For example:

Currently, we have 5 elements in the dataset category: Lipophilicity, ESOL, water-energy, jak2\_pic50 and logP14k. Applying the rule, we get

|  |  |
| --- | --- |
| Alphabetical Order | Number |
| ESOL | 0 |
| jak2\_pic50 | 1 |
| Lipophilicity\* | 2 |
| logP14k | 3 |
| water-energy | 4 |

\* In lexicographic order, upper case letters come before lower case ones (<https://chortle.ccsu.edu/java5/Notes/chap92/ch92_2.html>)

By applying the rule, water-energy-rf-0-tuned, will become 0\_3\_0\_0 (I chose the second number arbitrarily since it depends on the feature combination we want to use later on). Without the underscore in the middle, it will be impossible to tell which number belong to which component when the components in *dataset-algorithm-feature-tune* have more than 9 things in them. I also chose the underscore instead of (–) because (-) is also the subtraction operator.

\*\* But right now, it seems like only *dataset* and *feature* have the potential to surpass 9. We can then alter *dataset-algorithm-feature-tune* into *algorithm \_dataset\_ feature\_tune* (Fortunately, it is in lexicographic order) and remove the underscores that are separating each number. Let’s say the run number is 311180. Since we know algorithm and tune will never surpass 9, we can deduce that the two numbers 3 and 0 in each end must be *algorithm* and *tune*, so the run uses the 11th dataset and the 18th feature combination in their respective list. However, it gets hairy once the order of magnitude surpasses 2. For the run number 3111180, the number in between is 1111. There is no way to tell which part of the number belongs to *dataset* and which belongs to *feature*.

***Pros****:*

* Most efficient in terms of speed and memory since integers are mutable.
* The rule is simple enough to easily apply and automate.

***Cons****:*

* We have to make a choice between adding a separator between the integers or not. Depends on how big this project is, \*\* might not be applicable.

**Numbers and Letters:**