Fast Pairwise Redundancy

May 11, 2017

Given a feature set $F = \{f_1, ..., f_d\}$ and $k \in \mathbb{N}$ where k is the number of iterations to run.

```
In [10]: for k times:
             S = pick_random_subset(F)
             f = pick_random_feature(F \ S)

             # Contrast between S and f is proportional to its redundancy
             score = contrast(S, f)

             for i in S:
                 # For all contrast(S, f) with i in S, j equal to f,
                 # contrast({i}, j) <= contrast(S, f) <->
                 # "Adding an element to the subset can only increase redundancy"
                 redundancy[{f, i}] = min(score, redundancy[{f, i}])
```

To give an example let us consider $F = \{f_1, f_2, f_3, f_4\}$, where we would like to determine the redundancy of f_1 to f_2.

Assuming that the configurations (S, f_1) contained the subsets $S_1 = \{f_2, f_3\}, S_2 = \{f_2, f_4\}$, and that f_3 is redundant to f_1 but irredundant to f_2, f_4 is irredundant to f_1.

Scoring (S_1, f_1) will overestimate redundancy of $\{f_1, f_2\}$, as f_3 supplies additional information about f_1 compared to f_2 alone.
Scoring (S_2, f_1) will exactly equal the redundancy of $\{f_1, f_2\}$, as f_4 does not contain any information pertaining to f_1

As a result, our algorithm will slightly overestimate redundancy, but will arrive at the correct result without fail given a certain amount of iterations. To be more specific, the score will be correct once an iteration has a set $S' = S \setminus \{f_2\}$ where each element of S' is either completely redundant to f_2 or irredundant to f_1. Therefore, runtime will largely depend on the individual dataset, although an approximation running for a predetermined amount of iterations k will be sufficient in most cases.