Fast pairwise redundancy calculation

For the purpose of visualization in FEXUM, we require a redundancy score of every feature pair. To avoid running our correlation measure for each pair and thus reduce runtime, we use a heuristic to estimate the score. Given the circumstance that our contrast measure (see [2] in paper) is defined between a set and a feature, we can extrapolate redundancy for each feature from random subsets.

Given a feature set $F = \{f_1, ..., f_d\}$ in a d-dimensional dataset and $k \in \mathbb{N}$ where k is the number of iterations to run, we can define the following algorithm:

```
function FastPairwiseRedundancy(F, k)
    redundancies ← empty dictionary
    for k do
        S ← pickRandomSubset(F)
        f ← pickRandomFeature(F \ S)

        score ← contrast(S, f)
        for i ∈ S do
            redundancies[{f, i}] = min(score, redundancies[{f, i}])
        end for
    end for
    return redundancies
end function
```

We pick a random subset $S \subseteq F$ and a random feature f out of the remaining set $F \setminus S$, and calculate contrast. For each pair i, j with $i \in S$ and $j = f$, we save the minimum of our current score and previous calculations.

For all tuples (S, f) given features i, j with $i \in S$, $j = f$, it is true that $contrast(\{i\}, j) \leq contrast(S, f)$, because contrast is a measure and as such must be monotonic. As a result, our algorithm will converge toward the correct result, slightly overestimating redundancy until convergence. To be more specific, the score for the aforementioned pair i, j will be correct once there is an iteration with a set $S' = S \setminus \{i\}$ where each element of S' is either completely redundant to i or irredundant to j. Therefore, the time to achieve an optimal solution will depend on the individual dataset, although an approximation running for a predetermined amount of iterations k will be sufficient in most cases.

To give an example let us consider $F_1 = \{f_1, f_2, f_3, f_4\}$, where we would like to determine the redundancy of f_1 to f_2. We assume that the iterations of $contrast(S, f_1)$ used 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, while 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.