ENFrame at a glance

- Users write code in a subset of Python + relational queries.
- User code is oblivious to the probabilistic nature of the data: ENFrame interprets the code and runs it on probabilistic data.
- We already tested ENFrame on several algorithms, e.g., k-medoids clustering and k-nearest neighbour classification applied to probabilistic data representing query results.

Key Technical Features of ENFrame

- Language to express probabilistic events that capture arbitrary correlations in the input data and in the output as induced by program traces.
- Sequential and parallel algorithms for exact and approximate probabilistic computation of user programs.
- Quality metric for probabilistic computation using, e.g., ENFrame, naive and sampling-based methods.

Event Language, Event Programs, and Event Network

- Event program: ENFrame’s probabilistic interpretation of the user program is captured by events.
- Event language:
  - Event = random variables, conditioned values, Boolean formulas over events, arithmetic operations over events.
  - Variables of type \( T \) from user program become discrete random variables with PDFs over values in \( T \).
- c-values: values (numbers/vectors) conditioned on events.
  - For a number \( v \) and variable \( \Phi \):
    - c-value \( \Phi \otimes v \) evaluates to \( v \) if \( \Phi \) is true, or 0 otherwise.
  - c-values can be summed and compared:
    \[ \Phi_1 \otimes v_1 + \ldots + \Phi_n \otimes v_n \leq v_1 \otimes w_1 + \ldots + v_n \otimes w_n \]
    - Application example: sum of distances between objects.
- Event Network: Joint representation of interconnected events.

Example: User Program and Event Program for k-medoids Clustering

```python
#Initialization phase: Select k cluster medoids (centres)
1: (O, n) = loadData() # list and number of objects
2: (k, iter) = loadParams() # number of clusters and iterations
3: M = init() # initialise medoids
4: for it in range(0,iter): # clustering iterations
5: InCl = [None] * k
6: for i in range(0,k):
7: InCl[i] = [None] * n
8: for l in range(0,n):
9: DistSum[i][l] = reduce_sum([O[l] for l in range(0,n) if Centre[i][l]])
10: InCl[i][l] = reduce_and
11: for j in range(0,k):
12: DistSum[i][l] = reduce_sum([dist(O[l],M[i]) for j in range(0,k)])
13: InCl = breakTies2(InCl) # each object is in exactly one cluster
#Assignment phase: assign objects to closest medoid
14: DistSum = [None] * k
15: for i in range(0,k):
16: for p in range(0,n):
17: DistSum[i] = reduce_sum([dist(O[p],M[i]) for p in range(0,n) if InCl[i][p]])
18: Centre[i] = [None] * k
19: for i in range(0,k):
20: Centre[i] = [None] * n
21: for l in range(0,n):
22: Centre[i][l] = reduce_and
23: DistSum[i][l] = reduce_sum([p for p in range(0,n) if Centre[i][l]])
24: Centre = breakTies1(Centre) # enforce one centre per cluster
25: M = [None] * k
26: for i in range(0,k):
27: M[i] = reduce_sum([O[l] for l in range(0,n) if Centre[i]])
```

Challenges Currently under Microscope

- Which additional event language constructs are needed to capture further data analysis tasks?
- Trade-off: functionality (event-based result explanation, sensitivity analysis) vs. performance (coarse events compiled to C++ code)?

[ENFrame = Programs + Queries Probabilistic Data] Dan Olteanu & Sebastiaan J. van Schaik

Big Uncertain Data (BUDA), June 2014