Memory Mapping and Parallelizing Random Forests for Speed and Cache Efficiency

Eduardo Romero
romerogainza.1@osu.edu
The Ohio State University

Angela Li
li.10011@osu.edu
The Ohio State University

Christopher Stewart
cstewart@cse.ohio-state.edu
The Ohio State University

Kyle Hale
khale1@iit.edu
Illinois Institute of Technology

Nathaniel Morris
morris.743@osu.edu
The Ohio State University
Background – Memory Mapping

Memory mapping as a Depth-first alternative to tree traversal.

Table size grows exponentially
Design

Overview:
Phase 1 – Clustering & Compression

- List all paths in the forest.
- Sort features in each path.
- Sort all paths.
Phase 1 – Clustering & Compression

- Assign paths to clusters.
- Create table from each cluster.
- Resulting table size < naïve table
Phase 2: Parameter selection

How are clusters divided?
Phase 2: Parameter selection

- Tunable cluster size
- Assign tables and/or dictionaries to different clusters
Inference Overview

```
input data

front end

inference processing
for each e in Dictionary
    d = data @ e.features.key
    if (d == e.features.key.values)
        addr = data @ e.features
        results.add(lookup(addr))

result aggregation
Response = mean(results)
```

```
main memory storage

Dictionary[]
Struct Entry e
    Features: [<1,1>, <5,7>, ...]
    Key:
        Features: [<1,1>]
    Values: [0...]
    Collision: #FFFA33

memory-mapped tables

Struct Result j
    data: [5,10,8...]
    table: i

```

```
dict. IDs
dict.

(a,0)
(a,1)
(h,1)

3: (h,1)(e, 1)

f.d result

0: [no]
1: [yes]

1

3

1

1

1

1

hash fn

```

```
---features--- result dict. ID
0 0 0 0 ... 3
0 0 0 1 ... ...
0 0 1 0 ... ...
0 0 1 1 ... ...
0 1 0 0 ... ...
0 1 0 1 ...
..."
Phase 3: Improving Path Table selection

Could this model lead to false positives?
Yes. Hence Dictionary ID.

Bloom filter: Fast and Safe.
**Early Results**

MNIST Results:

Orders of magnitude faster than alternative approaches.

Multiple number of cores.

Two settings for max tree height.
Future Work

Search space methods
Optimization modeling
Explore datasets
Test interpretability benchmarks
References