Decision Tree and Random Forest Implementations for fast Filtering of Sensor Data

Sebastian Buschjäger and Katharina Morik

TU Dortmund University - Computer Science - Artificial Intelligence Group

July 3, 2018
So... Distributed computation hype?

1991 Ubiquitous Computing

1999 Internet of Things

2015 Edge Computing / Fog Computing
Machine Learning for small devices

**Fact** We measure a lot of data

**Thus** We need to transmit and analyze a lot of data
Machine Learning for small devices

**Fact** We measure a lot of data

**Thus** We need to transmit and analyze a lot of data

**Idea** Use Machine Learning locally to decide which data is useful

**Thus** Continuously apply ML model in realtime on small devices
Random Forest

**Fact** Random Forest is one of the best performing ML model

**Often** We design ML models independently from application
Random Forest

**Fact** Random Forest is one of the best performing ML models.

**Often** We design ML models independently from application.

What system is needed for a given tree/forest?

What is the best way to implement a Decision Tree?
Decision Tree

- **Inner nodes** make decision \( x_i < t \)
- **Leaf nodes** make prediction \( \hat{y} \)
Decision Tree

- **Inner nodes** make decision $x_i < t$
- **Leaf nodes** make prediction $\hat{y}$

Observation: Some path in tree have higher frequency than others
Probabilistic Analysis of Decision Trees

**Idea** Each decision is a Bernoulli Experiment with probability $p_{i\rightarrow j}$
Probabilistic Analysis of Decision Trees

Idea Each decision is a Bernoulli Experiment with probability $p_{i \rightarrow j}$

Path probability

$$p(\pi) = p_{\pi_0 \rightarrow \pi_1} \cdot \ldots \cdot p_{\pi_{L-1} \rightarrow \pi_L}$$
Probabilistic Analysis of Decision Trees

**Idea** Each decision is a Bernoulli Experiment with probability $p_{i \rightarrow j}$

**Path probability**

$$p(\pi) = p_{\pi_0 \rightarrow \pi_1} \cdot \ldots \cdot p_{\pi_{L-1} \rightarrow \pi_L}$$

**Expected no. of comparisons**

$$\mathbb{E}[L] = \sum_{\pi} p(\pi) \cdot |\pi|$$
Probabilistic Analysis of Decision Trees

**Idea** Each decision is a Bernoulli Experiment with probability $p_{i \rightarrow j}$

**Path probability**

$$p(\pi) = p_{\pi_0 \rightarrow \pi_1} \cdot \ldots \cdot p_{\pi_{L-1} \rightarrow \pi_L}$$

**Expected no. of comparisons**

$$\mathbb{E}[L] = \sum_{\pi} p(\pi) \cdot |\pi|$$

**Idea** Use expected no. of comparisons to estimate runtime
There are many ways to implement a Decision Tree

For Example: NativeTree

```c
bool predict(short const * x){
    unsigned int i = 0;
    while(!tree[i].isLeaf) {
        if (x[tree[i].f] <= tree[i].split) {
            i = tree[i].left;
        } else {
            i = tree[i].right;
        }
    }
    return tree[i].prediction;
}
```
There are many ways to implement a Decision Tree

**For Example:** If-Else-Tree

```c
bool predict(short const * x){
    if(x[0] <= 8191){
        if(x[1] <= 2048){
            return true;
        } else {
            return false;
        }
    } else {
        if(x[2] <= 512){
            return true;
        } else {
            return false;
        }
    }
}
```
There are many ways to implement a Decision Tree

For Example: Vectorized Tree

```c
bool predict(short const * x){
    unsigned int i = 0;
    unsigned int mask;
    void * tmp;
    while(!tree[i].isLeaf) {
        load_vectorized(tree[i],tmp);
        mask = compare_vectorized(tmp, x);
        i = mask_to_index(mask);
    }
    return tree[i].prediction;
}
```
Results

So which one is the best? And when?

Come visit me at my poster and find out!