Project A1
Data Mining for Ubiquitous System Software
Prof. Dr. Katharina Morik, Prof. Dr. Jian-Jia Chen

Phase 1 & 2

Central objectives
- Minimum prediction error
- Fast inference
- Small models
- Minimum resources assignment
- Minimum energy consumption
- Real-time guarantee

Limited to a single device

Phase 3

Extend analysis to
- Network of heterogenous devices
- Emerging new memory models
- Dynamic and adaptive execution of learning

Additional aspects
- Communication costs
- Synchronization costs
- Exploitation of heterogenous hardware
- Mapping models onto hardware

Hardware/Software Co-Design

Goal Reduece the gap between hardware design and machine learning
- Analyse model application
- Analyse model learning
- Explore applications in

Preliminary Results
- Architecture-specific implementation accelerates Random Forest application exploiting
- Model-dependent execution graph
- Data-dependent code synthesis
- Architecture-specific caching behaviour

Distributed Machine Learning

Goal How to learn utilizing the edge?

Open questions
- How to post-process or prune f?
- Regularisation instead of post-processing?
- What are the statistical guarantees?
- What are the real-time guarantees?

Approaches
- Constrained model families \( H \subset G \subset F \)
- Model learning via constrained optimization \( g = \arg\min_{f \in F} L(f, D) \)
- Model application via regularisation \( g = \arg\min_{f \in F} L(f, D) + \lambda R(f) \)

Machine Learning & Emerging Memory

Goal Identify resource saving potentials of non-volatile memories to enable architecture-aware learning algorithms

Non-volatile memories (NVM)
- Slow write, but fast read
- Only infrequent / no refresh required
- Potential drop-in replacement for DDR

Potential benefits for ML
- Apply ML in heavily resource restricted environments, e.g. smart bins
- Faster and more efficient model learning

Central question How to utilize NVM? For example, when to use NVM and DDR?

Data Aggregation and Sampling

Goal Extract representatives from stream
\( S^* = \arg\max_{S \in \Pi(V)} f(S) \)

where f is a sub-modular function.

Approach Apply Sieve-Streaming
- Add element if gain exceeds threshold
- Each Sieve has its own threshold
- Guarantees \( 1/2 - \varepsilon \) approximation by using \( O(\log k / \varepsilon) \) sieves

For example
\( f(S) = \log \det(\Sigma_S) \)

So far Bounding \( \log \det(\Sigma_S) \) leads to fewer sieves

Open questions
- How can we use summaries, e.g. for concept drift detection?
- Can we merge/delete elements from a summary?
- What is the relationship with coresets?

Representation, Execution, and Dependency of Learning

Goal Derive scheduling strategies for classes of ML models
- Probabilistic guarantees for both timing behaviour and statistical performance
- Respect precedence constraints using Dependency Graphs (DGs)
- Flexible DG construction and scheduling

Two orthogonal approaches during schedule design
- Start from DG with the minimum required learning output, and add constraints

Preliminary results Probability of deadline misses for multi-mode tasks with independent probability

Open questions
- Probabilistic timing guarantees for dependent random variables
- Dependent execution times in probabilistic graphical models
- Flexible precedence-constraints in scheduling and ML