Predicting Unroll Factors Using Supervised Classification

Hengjia Zhang, Kaitao Wu, Kejia Yang, Yuren Zhong

Loop Unrolling

- Loop unrolling is a loop transformation in which the loop body is **replicated** a number of times.
- Loop unrolling reduces overhead by decreasing the number of branch operations.

Normal loop	After loop unrolling

`int x;`
`for (x = 0; x < 100; x++)`
`{`
`  delete(x);`
`}`
	`int x;`
	`for (x = 0; x < 100; x += 5 )`
	`{`
	`  delete(x);`
	`  delete(x + 1);`
	`  delete(x + 2);`
	`  delete(x + 3);`
	`  delete(x + 4);`
	`}`

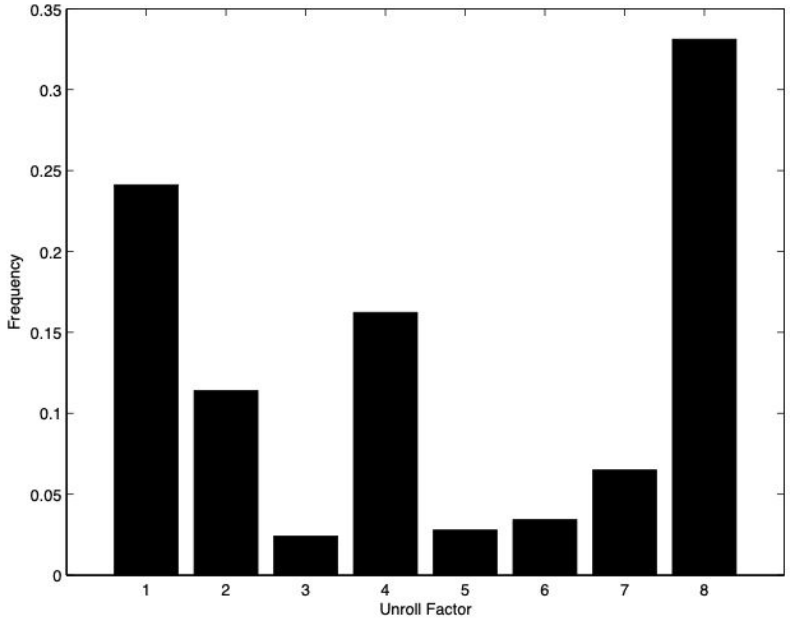
Loop Unrolling

- Advantages
  - Expose instruction level parallelism to the compiler
  - Decrease the branch penalty.
  - Reduce the number of executed instructions such as the same memory access

- Disadvantages:
  - Code expansion can degrade the performance of the instruction cache.
  - Increase the live ranges of variables and result in additional register pressure.

Problem

- It is difficult to decide when loop unrolling is appropriate.



Predicting unroll factor using supervised classification

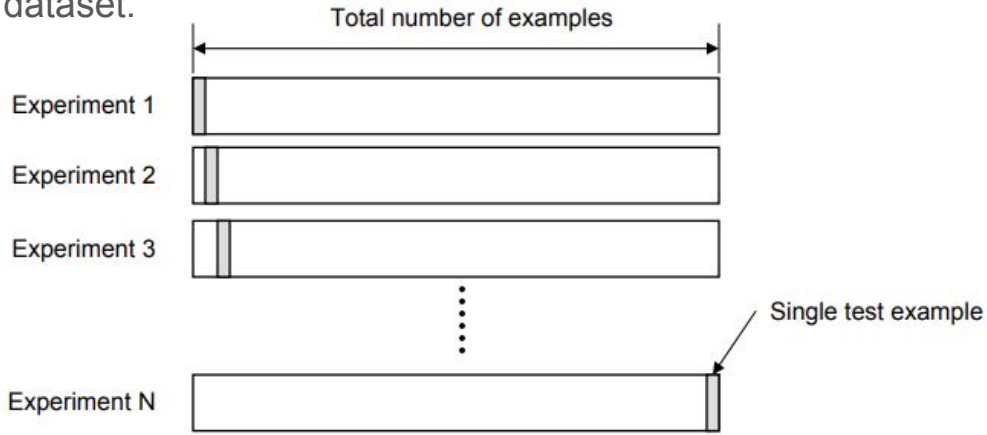
“Using the Open Research Compiler as a testbed, we demonstrate how one can use supervised learning techniques to determine the appropriateness of loop unrolling.”

High-level Idea

- Extract features from datasets.
  - Feature selection
- Extract training label for each unrollable loop in our benchmark.
  - Measure each loops using 8 different unroll factors (1, 2, …, 8)
  - Select the unroll factor for the loop that yields the best performance as label
- Using supervised learning methods to train offline
  - Nearest Neighbor
  - Support Vector Machine

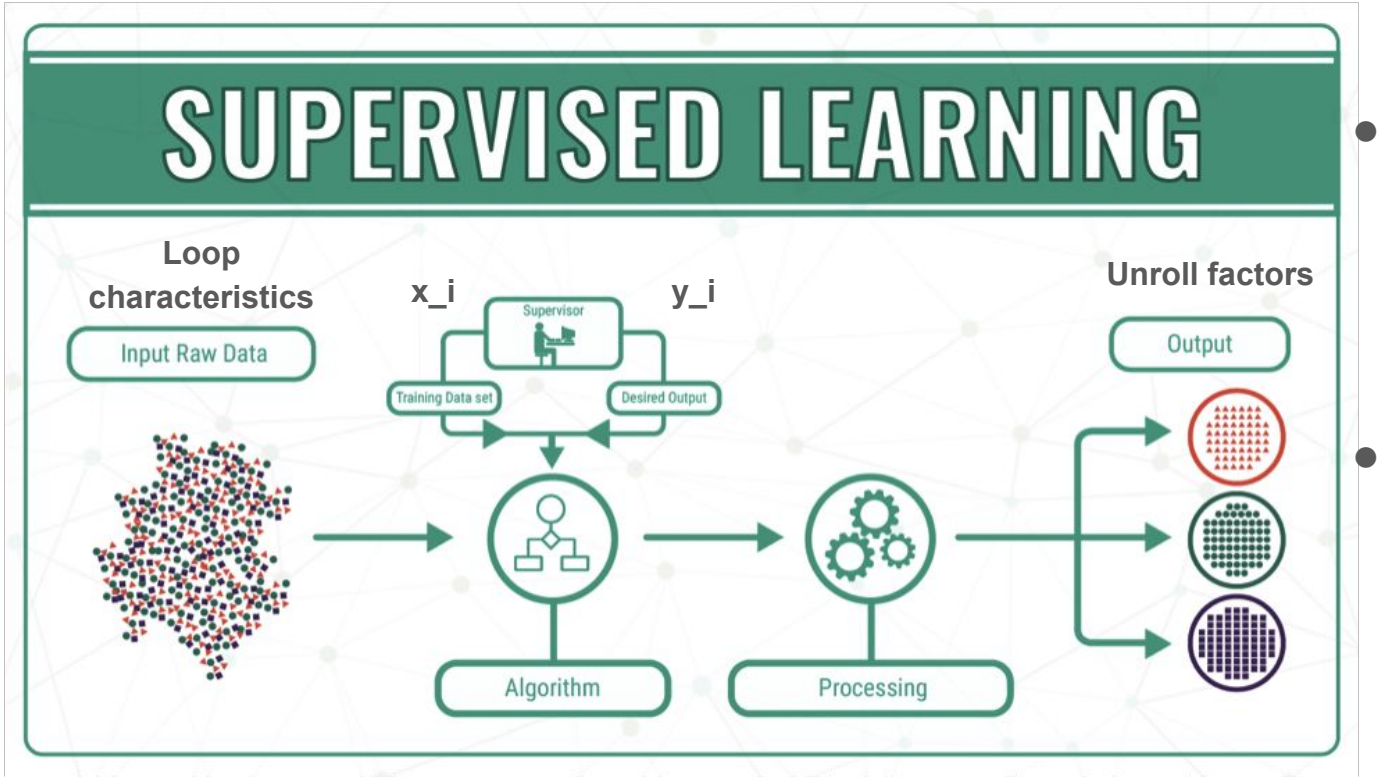
Methodology

- Computing the accuracy
  - Leave-one-out cross-validation (LOOCV): an iterative process that iterates N times, where N is the size of the training dataset.



- Benchmarks used
  - Extracted training examples from 72 benchmarks taken from a variety of benchmark suites (SPEC 2000, SPEC ’95, and SPEC ’92 [21])
  - For each benchmark they only use loops that ORC can unroll and whose optimal unroll factor is measurably better than the average (1.05x) over all unroll factors up to eight

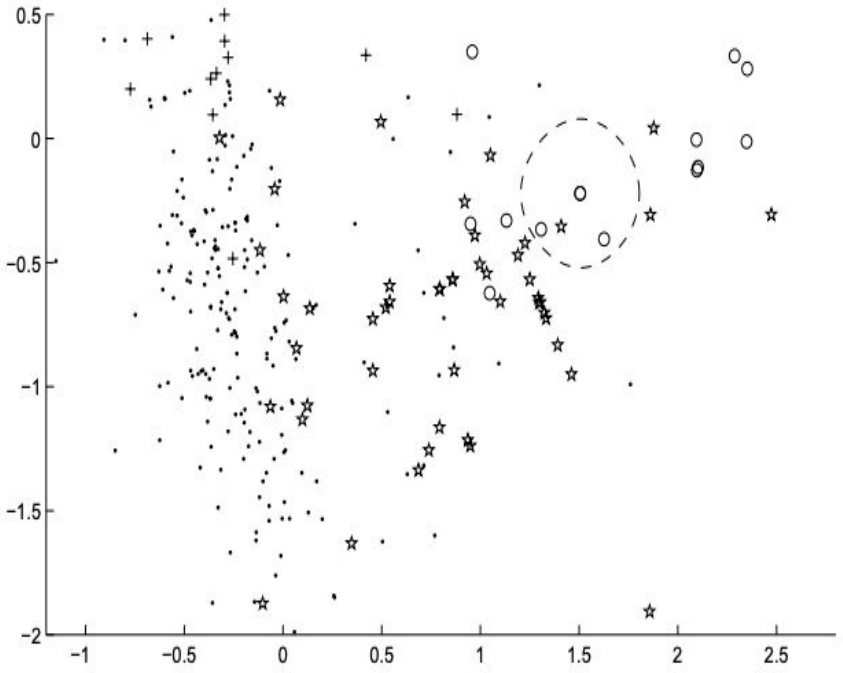
Supervised Learning



- Machine learning task of learning a function
  - Given: example input-output pairs
  - Maps an input to an output
- Loop characteristics: trip count of the loop, the number of operations in the loop body, the programming language, etc.

Multi-Class Classification: Nearest Neighbors

- Constructs a database of all <xi, yi> pairs in the training set.
- Computes the label for a novel example by inspecting the labels of the nearest examples in the database and choose the most common y.
- Use **Euclidean distance** as the similarity metric with $||x_i - x|| < d$

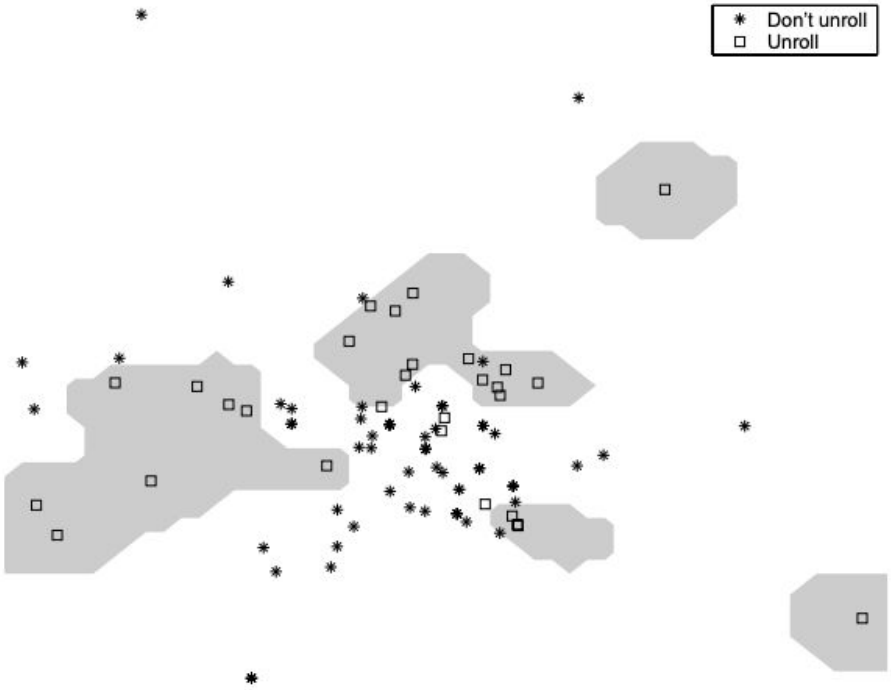


Multi-Class Classification: Support Vector Machine

- Transform multi-class classification problem into many binary classification problems.

class	h_1	h_2	h_3
1	1	0	0
2	0	1	0
3	0	0	1

- The multi-class prediction is the class label corresponding to the closest codeword (in hamming distance) to the query's code.



Multi-Class Classification: Support Vector Machine

Comparing to NN:

- More complex and longer to train than NN classification
- Less chance of overfitting compared to NN classification
  - Results in better accuracy on novel examples after training

Feature Selection

38 features are extracted

Need to determine which features to use

Too many features:

- Long training time
- Overfitting
- Noise

Too few features:

- Loss of information

Feature
The loop nest level.
The number of ops. in loop body.
The number of floating point ops. in loop body.
The number of branches in loop body.
The number of memory ops. in loop body.
The number of operands in loop body.
The number of implicit instructions in loop body.
The number of unique predicates in loop body.
The estimated latency of the critical path of loop.
The estimated cycle length of loop body.
The language (C or Fortran).
The number of parallel “computations” in loop.
The max. dependence height of computations.
The max. height of memory dependencies of computations.
The max. height of control dependencies of computations.
The average dependence height of computations.
The number of indirect references in loop body.
The min. memory-to-memory loop-carried dependence.
The number of memory-to-memory dependencies.
The tripcount of the loop (-1 if unknown).
The number of uses in the loop.
The number of defs. in the loop.

Mutual Information Score

MIS measures the reduction in uncertainty of the label y if one feature f become known, which is defined by

$$I(f;y) := \sum_{v \in \mathcal{V}_f} \sum_{l \in \mathcal{L}} \mathbb{P}[f = v, y = l] \cdot \log\left(\frac{\mathbb{P}[f = v, y = l]}{\mathbb{P}[f = v] \cdot \mathbb{P}[y = l]}\right)$$

Informative features will receive higher scores than uninformative features.

Rank	Feature	MIS
1	# floating point operations	0.19
2	# operands	0.186
3	instruction fan-in in DAG	0.175
4	live range size	0.16
5	# memory operations	0.148

Mutual Information Score

MIS measures the reduction in uncertainty of the label y if one feature f become known, which is defined by

$$I(f;y) := \sum_{v \in \mathcal{V}_f} \sum_{l \in \mathcal{L}} \mathbb{P}[f = v, y = l] \cdot \log\left(\frac{\mathbb{P}[f = v, y = l]}{\mathbb{P}[f = v] \cdot \mathbb{P}[y = l]}\right)$$

Informative features will receive higher scores than uninformative features.

Drawbacks:

- Ignore correlation between features
- No guarantee on actual performance (regardless of classifier)

Greedy Feature Selection

Iteratively choose a sequence of features, given a classifier (e.g., NN, SVM):

- Suppose F is the set of all features and c is the classifier
- Let $A(g; c)$ be the accuracy obtained when training classifier c on feature selection $g \subseteq F$
- Pick $f_1 \in F$ by $f_1 = \arg\min_{f \in F} A(\{f\}; c)$
- Pick $f_2 \in F/\{f_1\}$ by $f_2 = \arg\min_{f \in F/\{f_1\}} A(\{f_1, f\}; c)$
- Pick $f_3 \in F/\{f_1, f_2\}$ by $f_3 = \arg\min_{f \in F/\{f_1, f_2\}} A(\{f_1, f_2, f\}; c)$
- ...

Rank	NN	Error	SVM	Error
1	# operands	0.48	# floating point operations	0.59
2	live range size	0.06	loop nest level	0.49
3	critical path length	0.03	# operands	0.34
4	# operations	0.02	# branches	0.20
5	known tripcount	0.02	# memory operations	0.13

Greedy Feature Selection

Iteratively choose a sequence of features, given a classifier (e.g., NN, SVM):

- Suppose F is the set of all features and c is the classifier
- Let $A(g; c)$ be the accuracy obtained when training classifier c on feature selection $g \subseteq F$
- Pick $f_1 \in F$ by $f_1 = \arg\min_{f \in F} A(\{f\}; c)$
- Pick $f_2 \in F/\{f_1\}$ by $f_2 = \arg\min_{f \in F/\{f_1\}} A(\{f_1, f\}; c)$
- Pick $f_3 \in F/\{f_1, f_2\}$ by $f_3 = \arg\min_{f \in F/\{f_1, f_2\}} A(\{f_1, f_2, f\}; c)$
- ...

Drawbacks:

Need to re-run when use a new classifier

Don’t know correlation of features

May not be the best practical choice

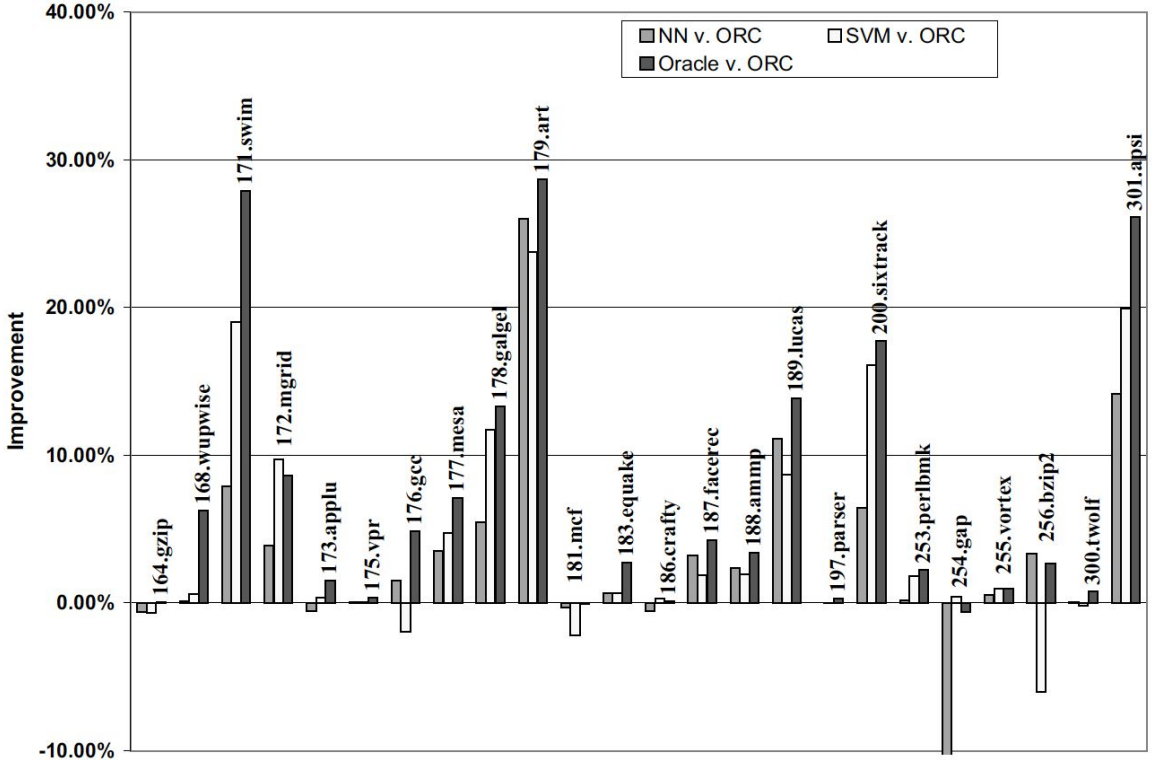
Feature Selection

In this paper, they take the union of the top features in MIS and GFS

Rank	Feature	MIS
1	# floating point operations	0.19
2	# operands	0.186
3	instruction fan-in in DAG	0.175
4	live range size	0.16
5	# memory operations	0.148

Rank	NN	Error	SVM	Error
1	# operands	0.48	# floating point operations	0.59
2	live range size	0.06	loop nest level	0.49
3	critical path length	0.03	# operands	0.34
4	# operations	0.02	# branches	0.20
5	known tripcount	0.02	# memory operations	0.13

Experiment

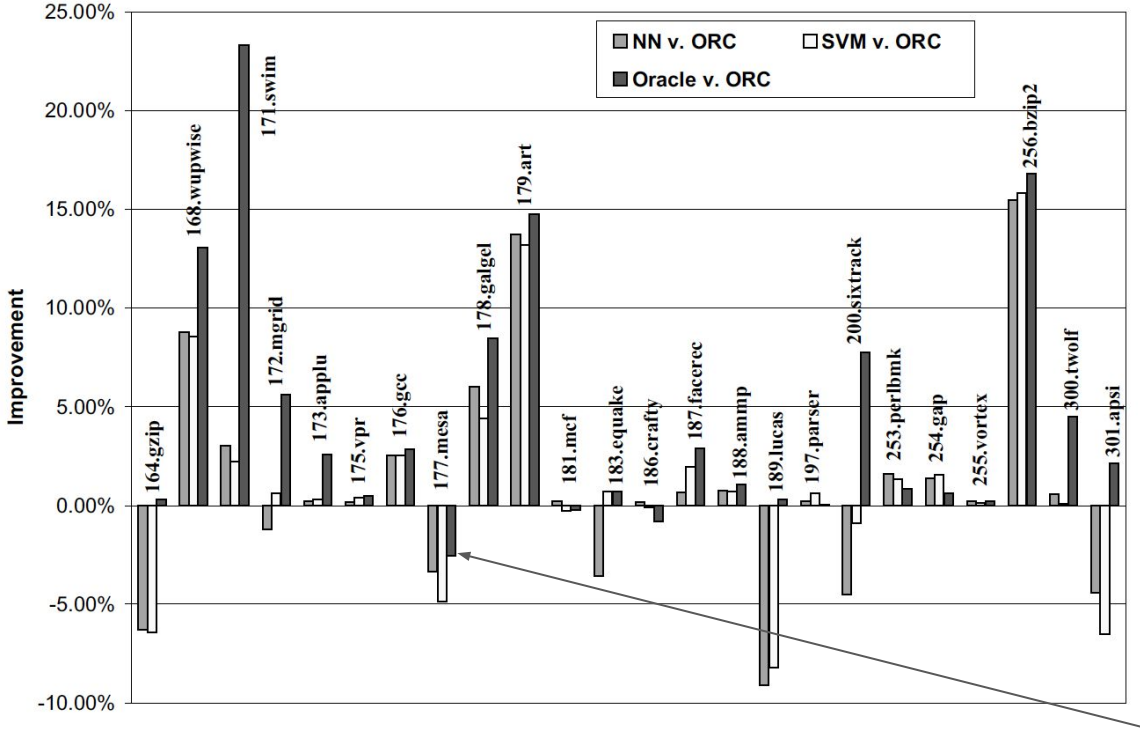


Realized performance on the SPEC 2000 benchmarks with SWP disabled.

*Oracle: the “perfect” classifier

*Baseline: ORC performance

Experiment



Realized performance on the SPEC 2000 benchmarks with SWP enabled.

*Oracle: the “perfect” classifier

*Baseline: ORC performance

Result

SPEC 2000 benchmarks with SWP disabled

- Both NN and SVM achieves a speedup on 19 of the 24 SPEC benchmarks
- The SVM technique attains overall
  - 5% average speedup on the SPECs
  - 9% speedup on the SPECfp only [outperformed]
- The NN technique boosted the performance by 4%
- Oracle attains 7.2% on average

SPEC 2000 benchmarks with SWP enabled

- Both NN and SVM achieves a speedup on 16 of the 24 SPEC benchmarks, leading to slightly over 1% speedup overall.
- Oracle attains 4.4% speedup overall

Result [with SWP disabled]

- Their best SVM classifier
  - Predict with 65% accuracy the optimal unroll factor
  - Predict nearly optimal unroll factor 79% of time, which is within 7% of optimal performance
- Their best NN classifier
  - predict with 62% accuracy the optimal unroll factor

Discussion

- Need support from compiler infra
  - Features extracted from compiler writer (in our case, ORC infra)
  - Time-consuming to collect labels
- Need regression to expand the label space
- Eliminate noises

Any Questions?