Automatic Generation of Behavioral Hard Disk Drive Access Time Models

Adam Crume[1] Carlos Maltzahn[1] Lee Ward[2]
Thomas Kroeger[2] Matthew Curry[2]

[1]University of California, Santa Cruz
{adamcrume, carlosm}@cs.ucsc.edu

[2]Sandia National Laboratories, Livermore, CA
{lee, tmkroeg, mlcurry}@sandia.gov

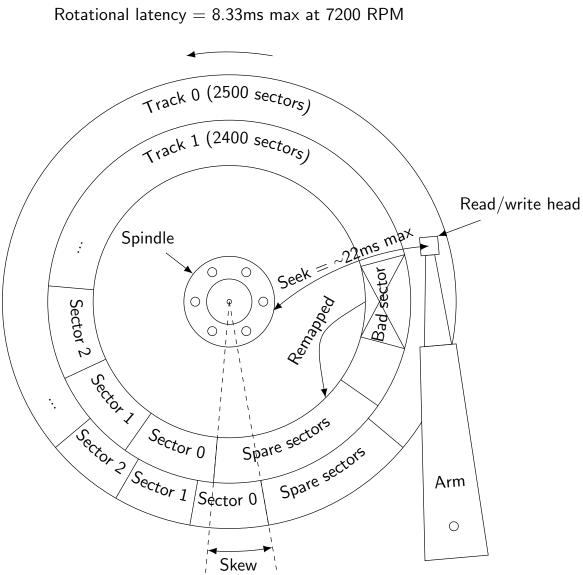
June 6, 2014

Predicting hard drive performance

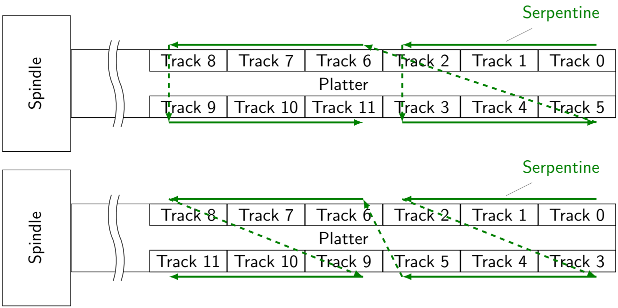
Use cases:

- System simulations
- File system design
- Quality of service / predictable performance

Complexity, part 1



Complexity, part 2



Complexity, part 3

Additionally:

- Queueing
- Scheduling
- Caching
- Readahead
- Write-back

Machine learning of hard drive performance

Short term goals (this presentation):

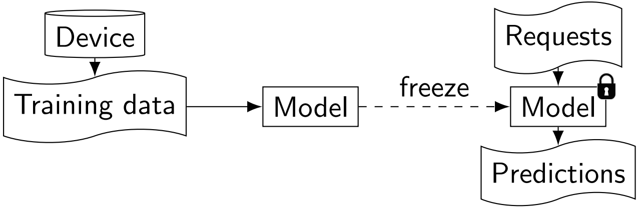
- Automated
- Fast

Long term goals (future work):

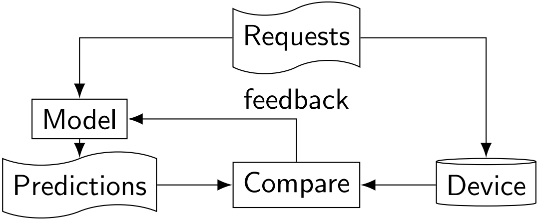
- Future-proof
- Device-independent

Offline vs. online machine learning

Offline: separate train and test phases



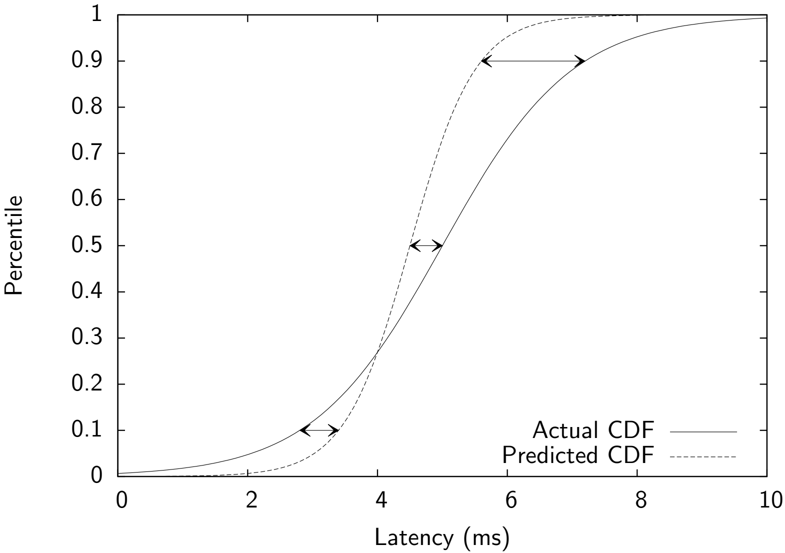
Online: feedback from real device



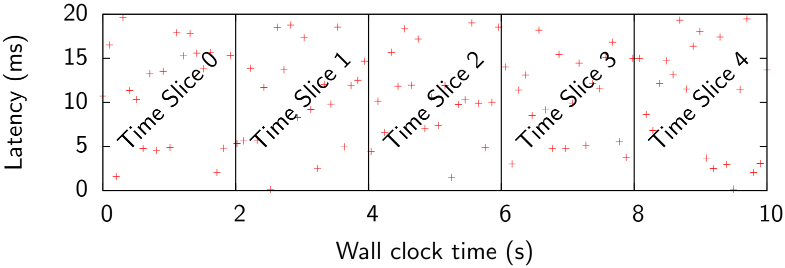
Offline vs. online use cases

- System simulations (offline)
- File system design (offline)
- Quality of service / predictable performance (offline/online)

Existing machine learning approaches: demerit



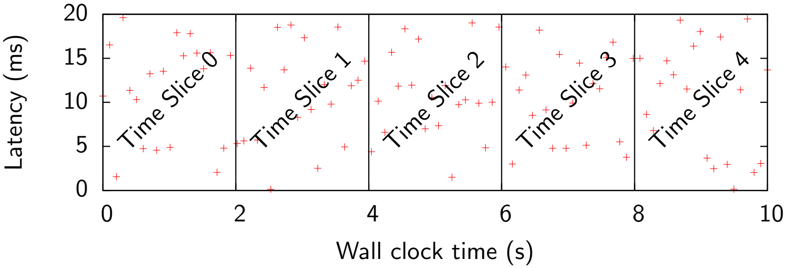
Existing machine learning approaches: average in time slice



For each time slice:

$$\left.\begin{array}{lcl} r_0 & \rightarrow & prediction_0 \\ r_1 & \rightarrow & prediction_1 \\ \vdots & & \vdots \\ r_n & \rightarrow & prediction_n \end{array}\right\} \rightarrow \text{average} \rightarrow \text{compare with real average}$$

Existing machine learning approaches: predict average



For each time slice:

$$\left.\begin{matrix} r_0 \\ r_1 \\ \vdots \\ r_n \end{matrix}\right\} \rightarrow \text{aggregate} \rightarrow \text{predict average} \rightarrow \text{compare with real average}$$

Existing machine learning approaches: limitations

All aggregate. None predict individual latencies with low error.

Hard part? Access times.

Workload

Characteristics:

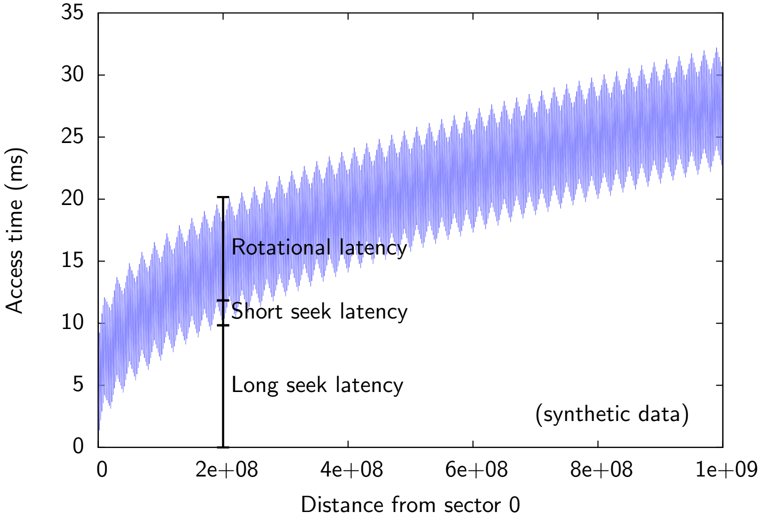
- Random
- Read-only
- Single-sector
- Full utilization
- First serpentine

Minimizes:

- Caching
- Readahead
- Write-back
- Transfer time
- Request arrival time sensitivity
- Track length variation

Workload emphasizes access time (which is a hard problem by itself) and de-emphasizes everything else. Other workloads are future work.

Access time breakdown



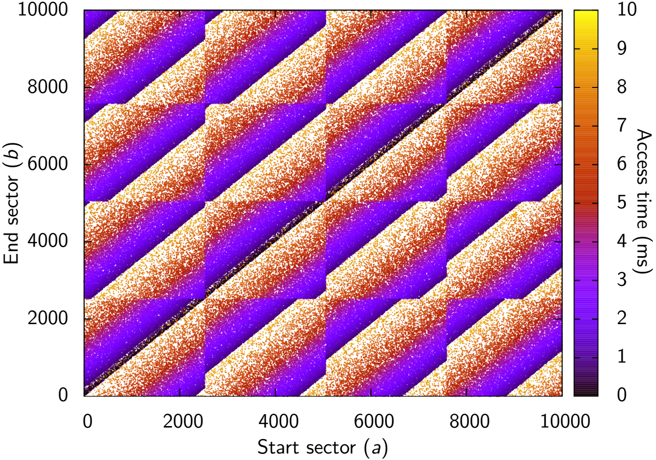
Access times: unpredictable?

- Why are access times hard to predict?
  - Rotational layout
  - Serpentines
  - Sector sparing
  - Skew

Access times: unpredictable?

- Why are access times hard to predict?
    - Rotational layout
    - Serpentines
    - Sector sparing
    - Skew
- What do these have in common?
    - Periodicity!
      Generic machine learning algorithms cannot directly predict periodic functions well.

Access time function



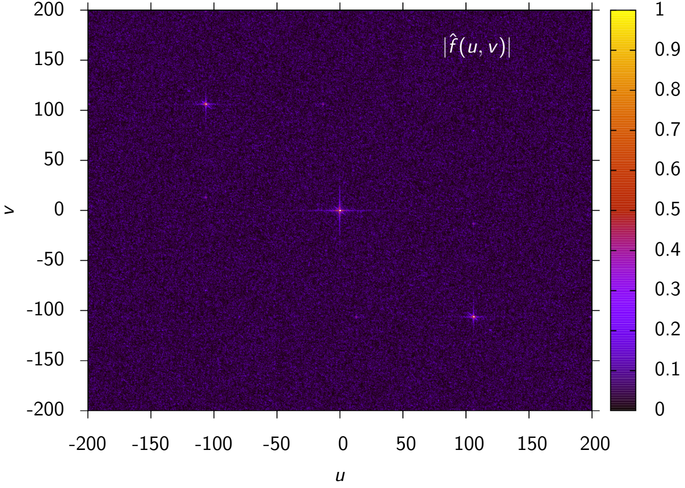
Full table is 1 billion by 1 billion entries, would take approximately 500 million years to capture data and 3.5 exabytes to store it.
Extremely sparse sampling is required, must compute on the fly.

Input augmentation

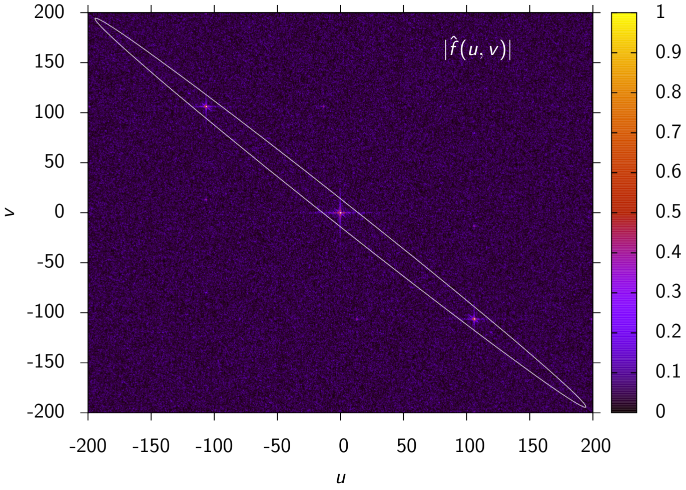
$$\begin{pmatrix} a \\ b \end{pmatrix} \rightarrow \begin{pmatrix} a \\ \sin(2\pi a/p_1) \\ \cos(2\pi a/p_1) \\ \sin(2\pi a/p_2) \\ \cos(2\pi a/p_2) \\ \vdots \\ b \\ \sin(2\pi b/p_1) \\ \cos(2\pi b/p_1) \\ \sin(2\pi b/p_2) \\ \cos(2\pi b/p_2) \\ \vdots \end{pmatrix}$$

(a is the start sector, b is the end sector)

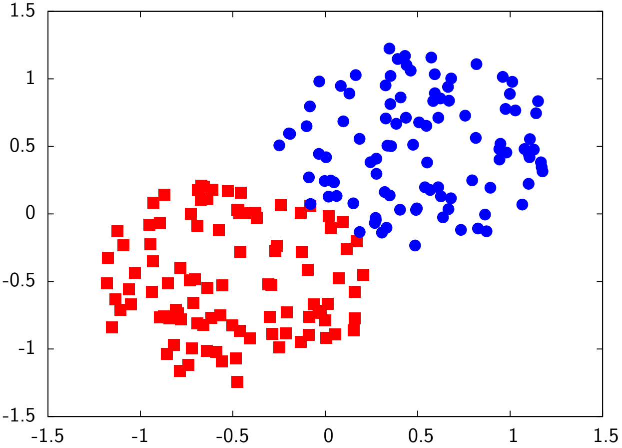
Fourier transform



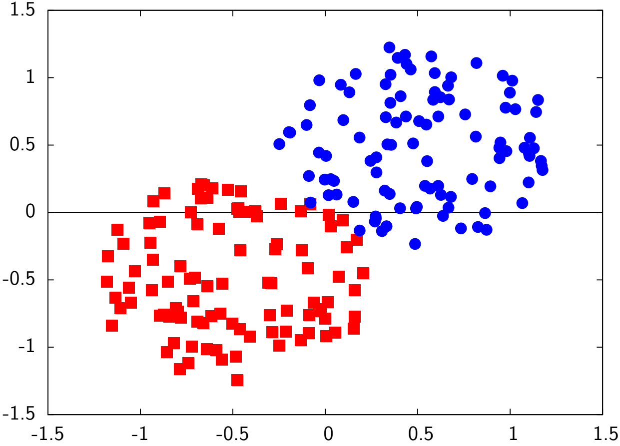
Fourier transform



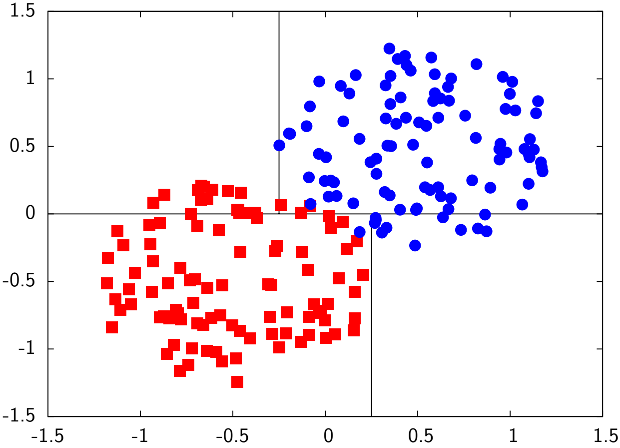
Decision trees



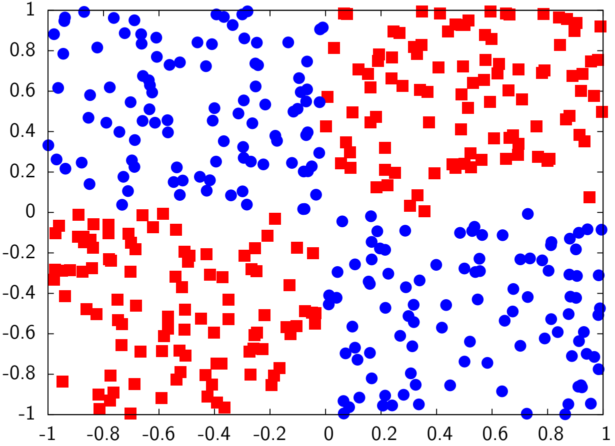
Decision trees



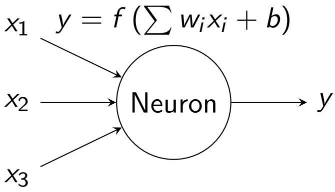
Decision trees



Interdependence

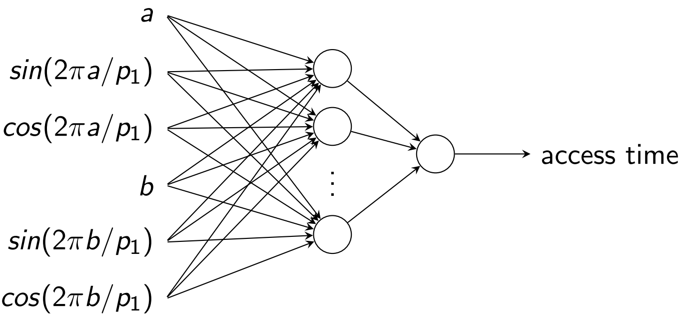


Neural net basics



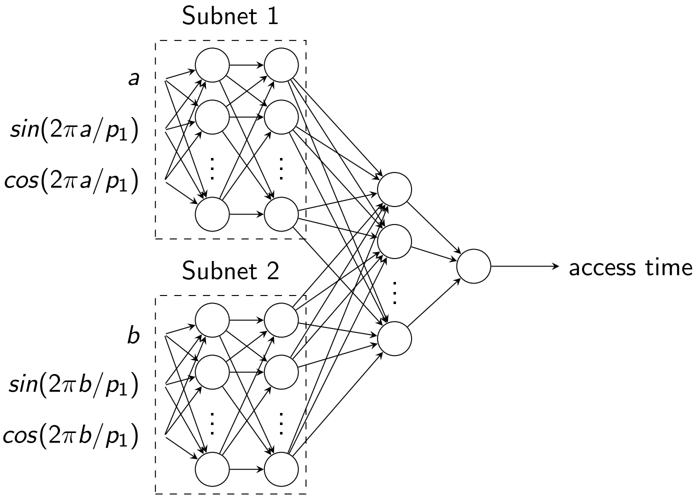
- Usually, $f(x) = \tanh(x)$ (or similar). Final output may use $f(x) = x$.
- Training: given input x_i and desired output y^*, adjust w_i and b such that $y = y^*$

Flat neural net



(a is the start sector, b is the end sector)

Neural net with shared weights



(a is the start sector, b is the end sector)

Hyperparameters

Decision trees

- Periods to include
- Maximum depth
- Minimum instance count per leaf
- Minimum variance proportion
- Prune or not
- Pruning folds
- Ensemble size

Neural nets

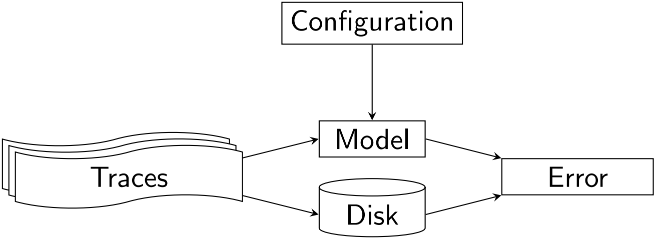
- Periods to include
- Layer sizes
- Initial weight distributions
- Learning rate
- Momentum

What's optimal?

Auto-tuning hyperparameters

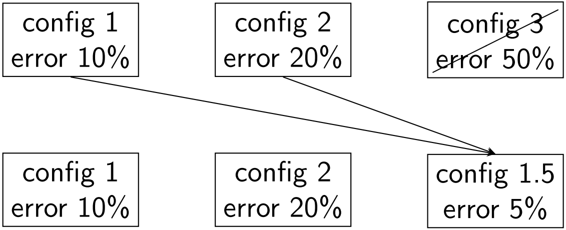
Genetic algorithm overview:

1. Population initialized with random configurations
2. Each configuration evaluated by training and evaluating model



Auto-tuning hyperparameters

3. Poor configurations are discarded, good configurations are mated and mutated



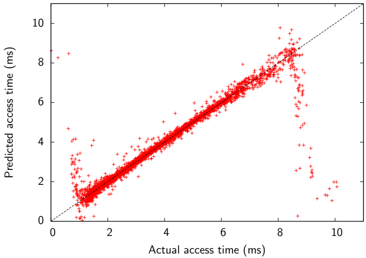
4. Go to step 2, repeat until error is low

L_1 and L_2 norms

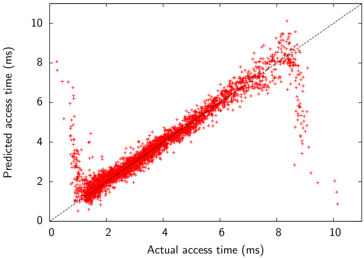
	L_1	L_2
Equivalent to	Mean Absolute Error	Mean Squared Error
Formula *	$\sum_i \lvert\hat{y}_i - y_i\rvert$	$\sum_i (\hat{y}_i - y_i)^2$
Advantages	Outlier resistant	Smooth
Disadvantages	Non-differentiable at 0	Assumes Gaussian noise

*Note: The actual formula for the L_2 norm is $\sqrt{\sum_i (\hat{y}_i - y_i)^2}$, but the above has identical optima and is easier to use.

L_1 versus L_2, empirically



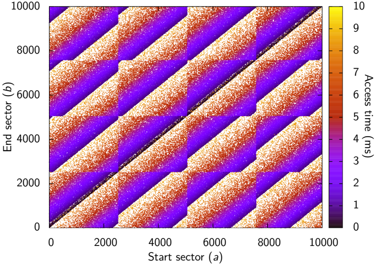
Trained with L_1



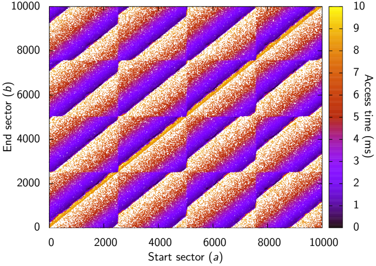
Trained with L_2

Comparison of L_1 versus L_2 norm for neural nets with subnets. (A narrow cluster along the diagonal $y = x$ is better.) High errors are seen at extreme values due to discontinuities in the access time function.

Access times

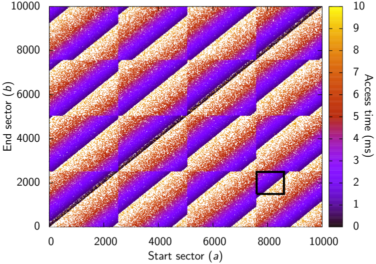


Actual

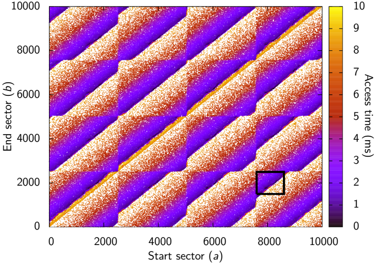


Predicted

Access times

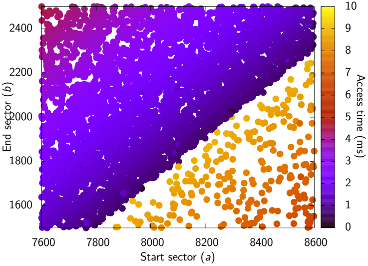


Actual

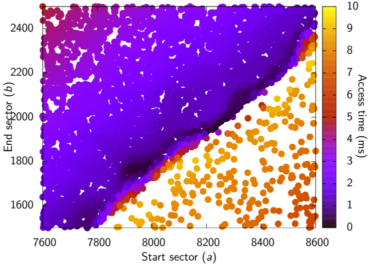


Predicted

Access times

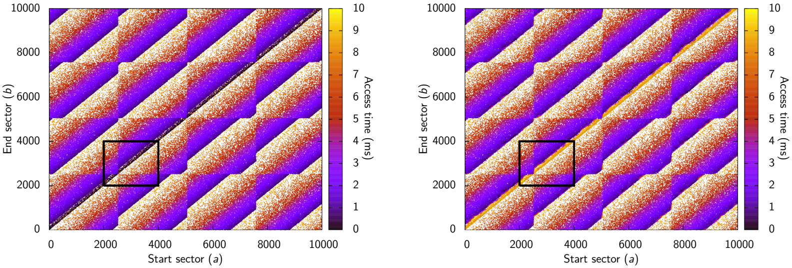


Actual



Predicted

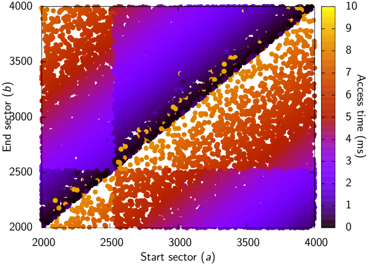
Access times



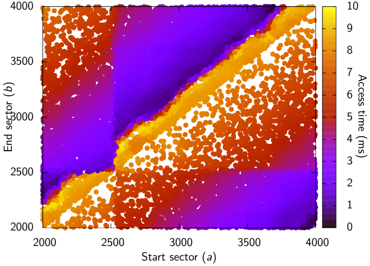
Actual

Predicted

Access times

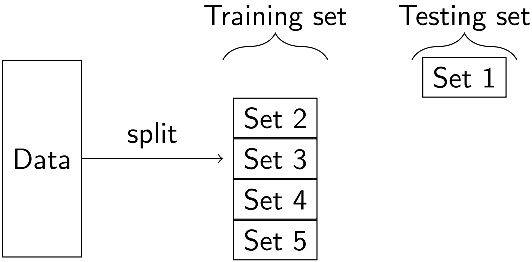


Actual

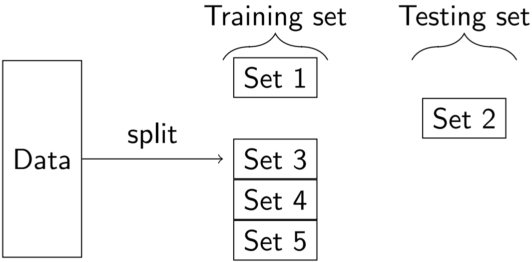


Predicted

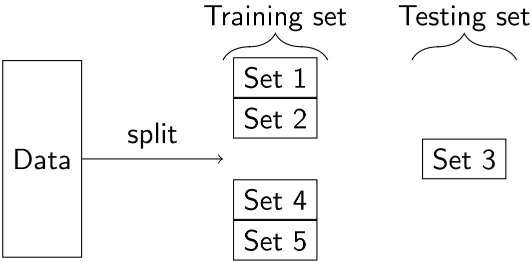
Cross validation



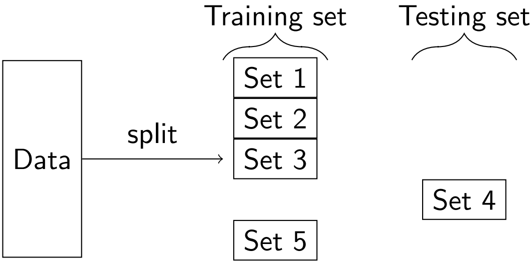
Cross validation



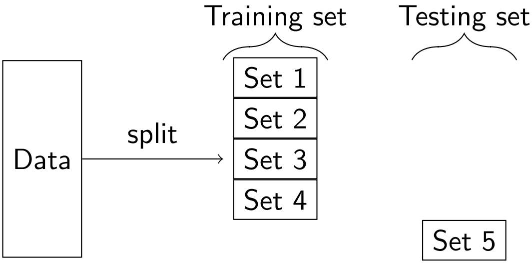
Cross validation



Cross validation

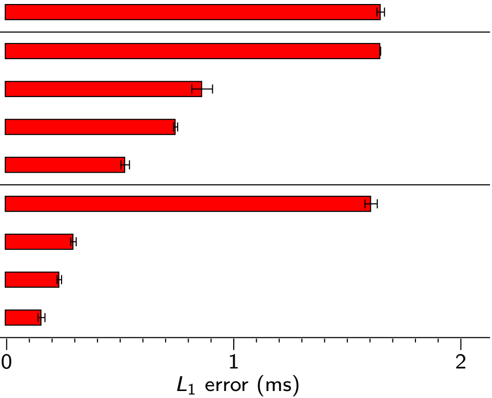


Cross validation



Results

	Configuration	L_1 error (ms)
	constant value	1.651 ± 0.017
Decision trees	no periods	1.649 ± 0.001
	no bagging	0.865 ± 0.046
	no auto-tuning	0.748 ± 0.009
	all enabled	0.526 ± 0.019
Neural nets	no periods	1.608 ± 0.027
	no subnets	0.298 ± 0.012
	no auto-tuning	0.236 ± 0.010
	all enabled	0.157 ± 0.015



RMS errors for predictions over the first 237,631 sectors (94 tracks) with a random read workload.

Conclusions

- Periodicity information improves multiple algorithms
- High-level assumption, likely to apply to many devices
- Machine learning of per-request latencies is possible