Fourier-Assisted Machine Learning of Hard Disk Drive Access Time Models

Adam Crume[1] Carlos Maltzahn[1] Lee Ward[2]
Thomas Kroeger[2] Matthew Curry[2] Ron Oldfield[2]
Patrick Widener[2]

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

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

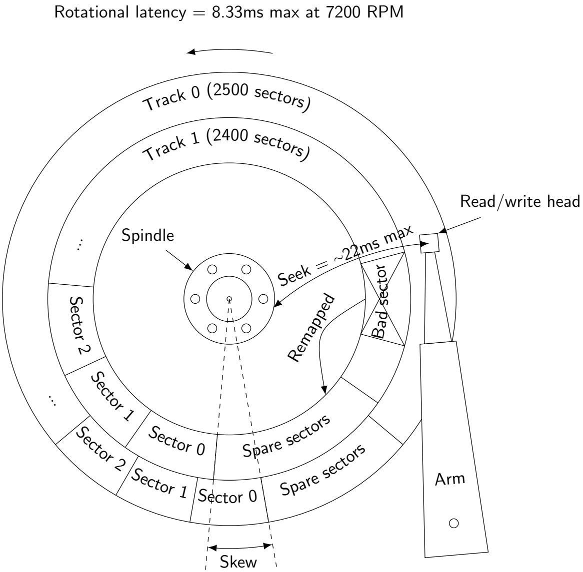
November 18, 2013

Predicting hard drive performance

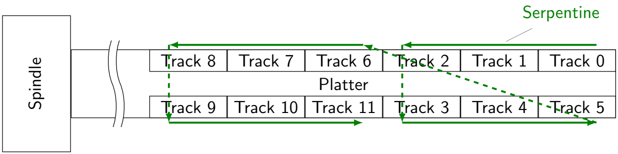
Use cases:

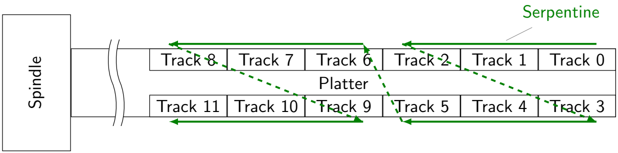
- System simulations
- File system design
- Quality of service / real-time guarantees (Anna Povzner's work with Fahrrad)

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):

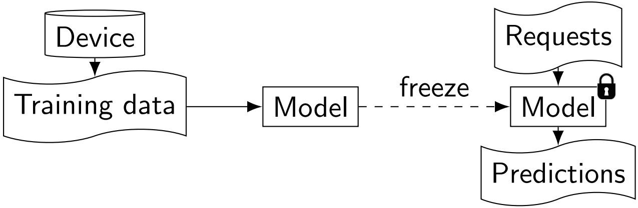
- Automated
- Fast

Long term goals (future work):

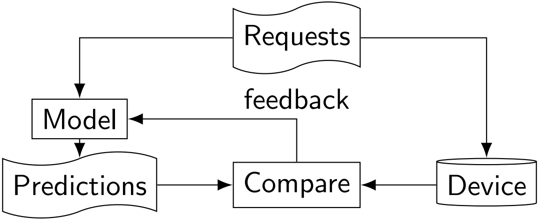
- Flexible
- Future-proof
- Device-independent

Offline vs. online machine learning

Offline: separate train and test phases



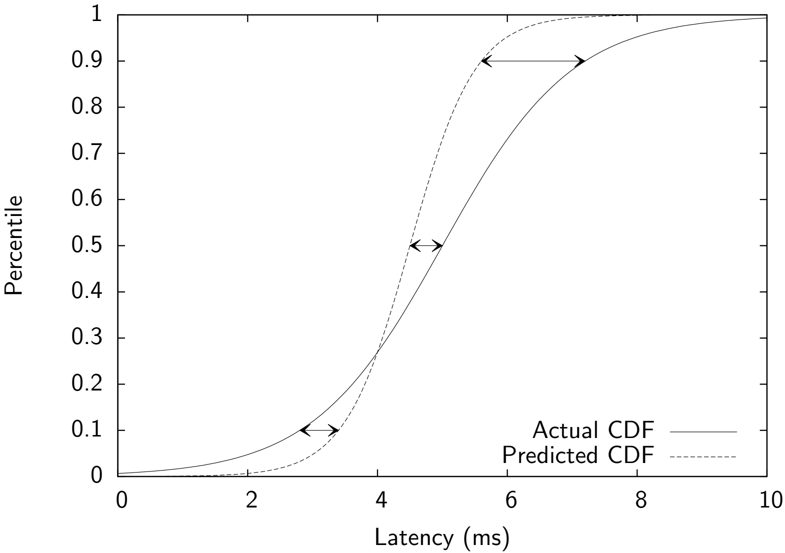
Online: feedback from real device



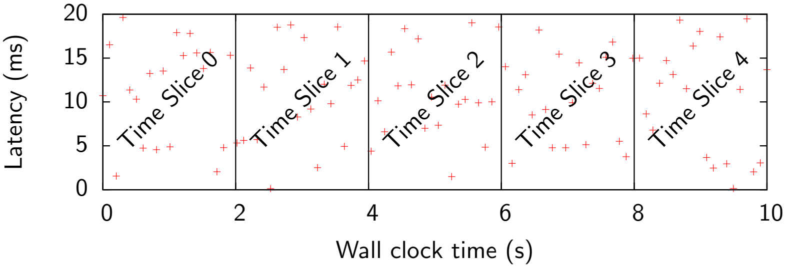
Offline vs. online use cases

- System simulations (offline)
- File system design (offline)
- Quality of service / real-time guarantees (offline/online)

Existing machine learning approaches: demerit



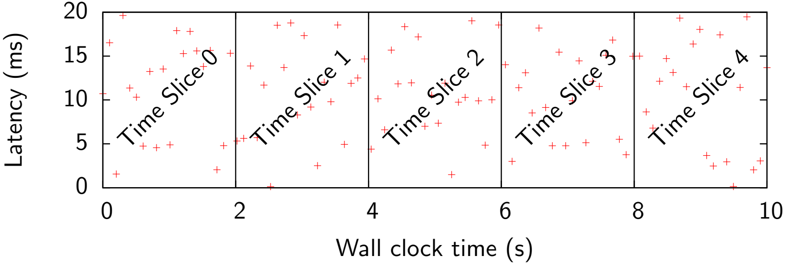
Existing machine learning approaches: average in time slice



For each time slice:

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

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:

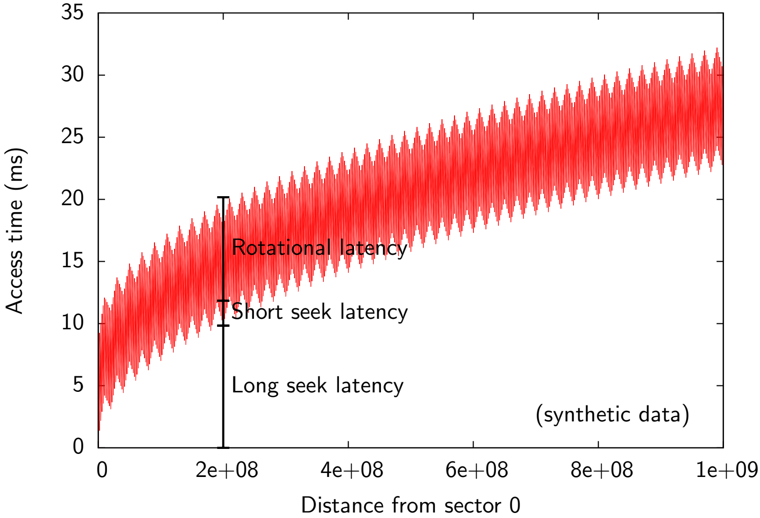
- Random
- Read-only
- Single-sector
- Full utilization
- First serpentine

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?

Access times: unpredictable?

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

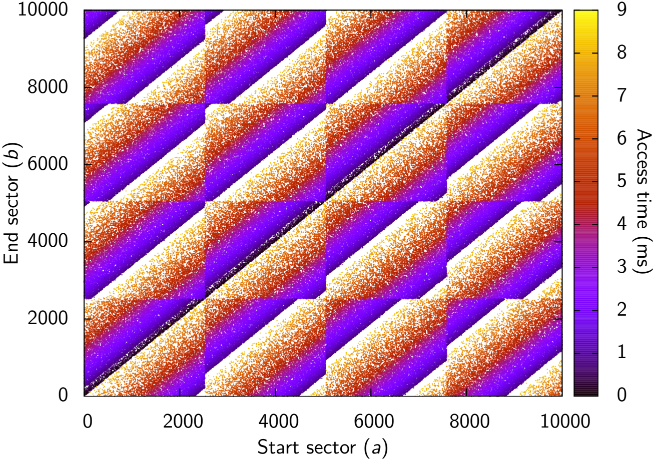
Access times: unpredictable?

- Why are access times hard to predict?
  - Rotational layout
  - Serpentines
  - Sector sparing
  - Skew
- What do these have in common?

Access times: unpredictable?

- Why are access times hard to predict?
  - Rotational layout
  - Serpentines
  - Sector sparing
  - Skew
- What do these have in common?
  - Periodicity!
    Most 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.

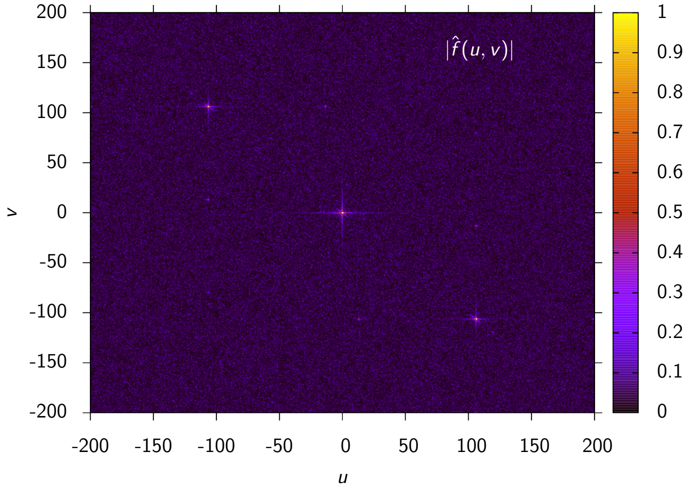
Input augmentation

Key idea 1 of 2: add sines and cosines to inputs

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

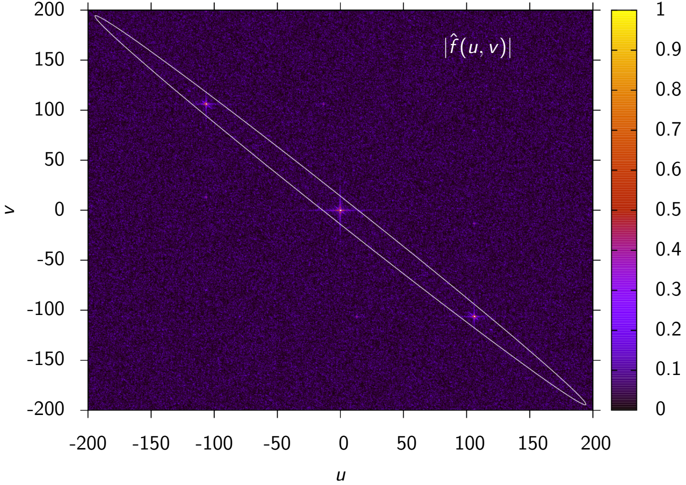
(a is the start sector, b is the end sector)

Fourier transform

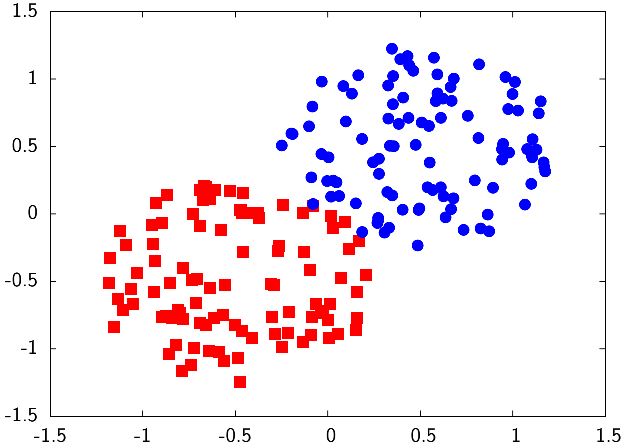


Fourier transform

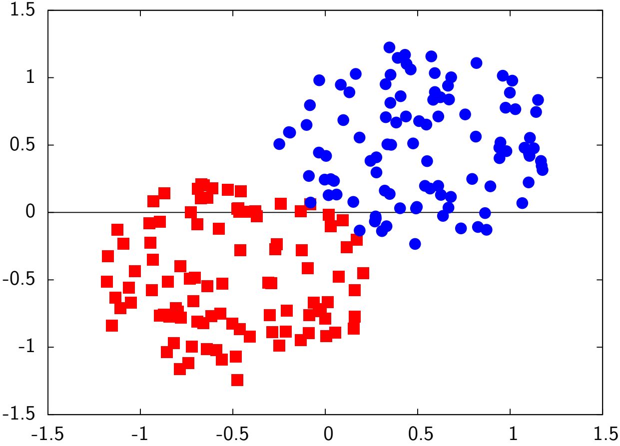
Key idea 2 of 2: search on diagonal to limit computation



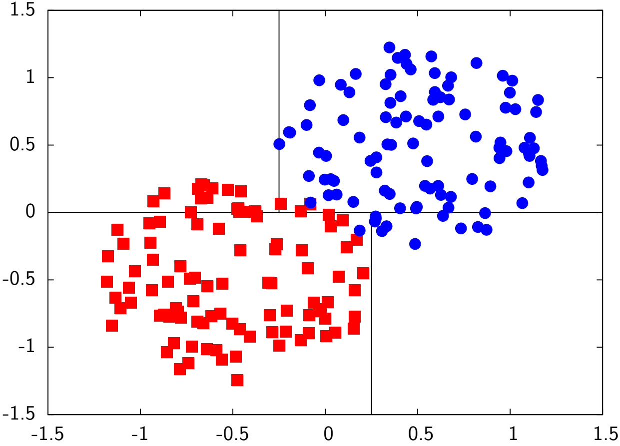
Decision trees



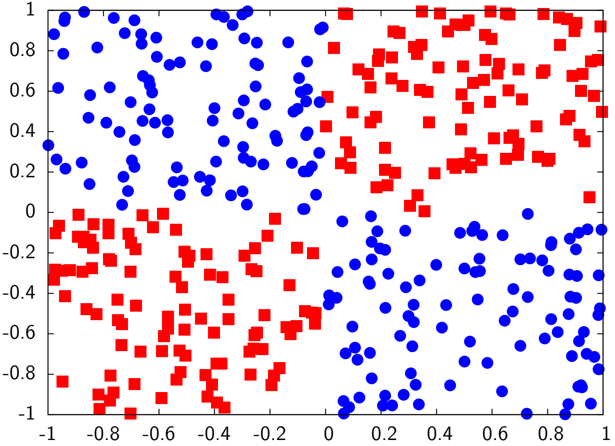
Decision trees



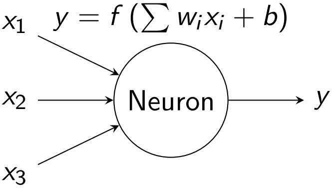
Decision trees



Interdependence

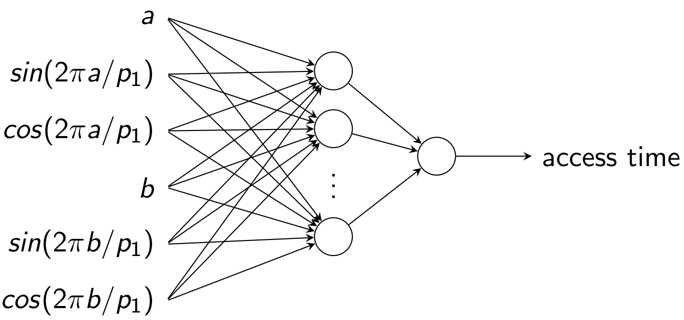


Neural net basics



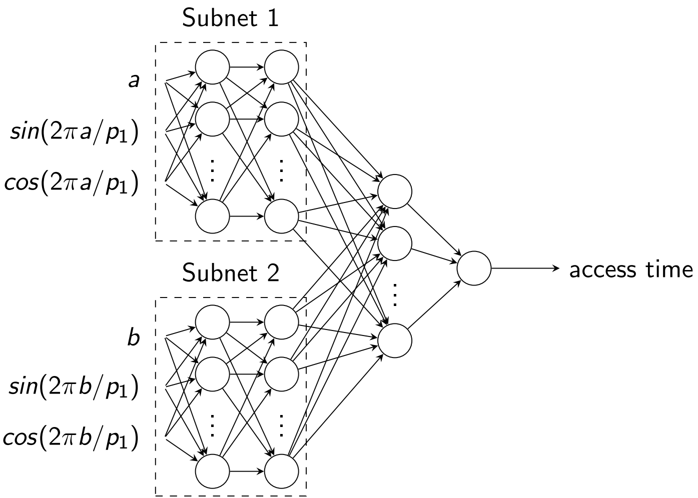
- Usually, $f(x) = \tanh(x)$ (or similar). Final output may use $f(x) = x$.
- Training: given input x_i and desired output y^*, adjust w_i and b such that $y = y^*$

Flat neural net



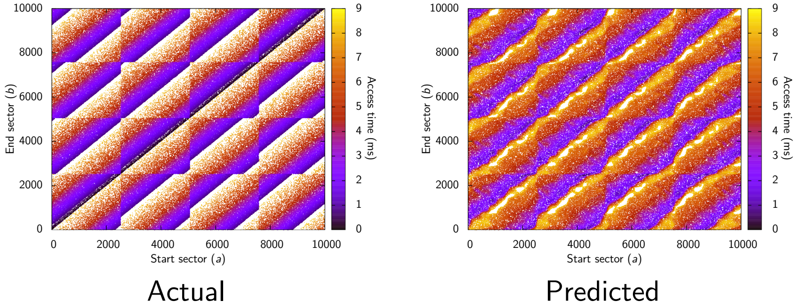
(a is the start sector, b is the end sector)

Neural net with shared weights

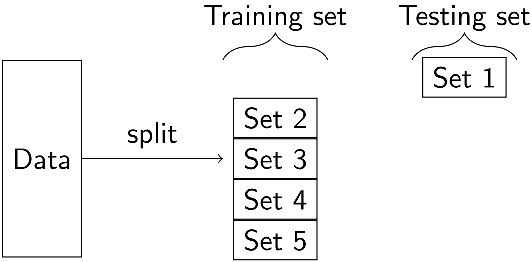


(a is the start sector, b is the end sector)

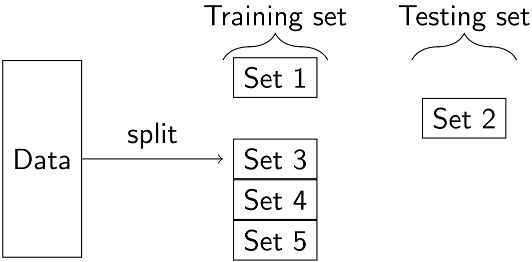
Access times



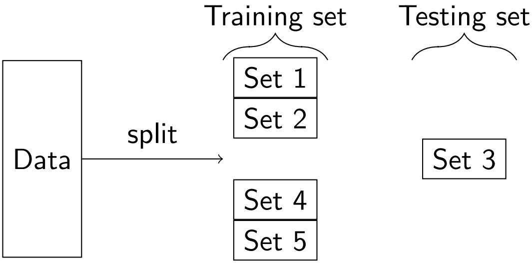
Cross validation



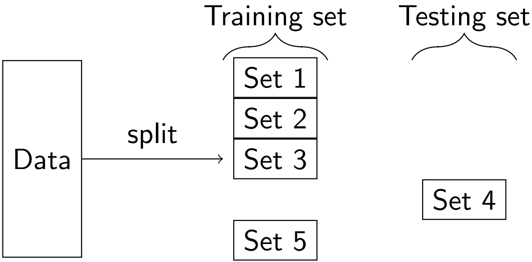
Cross validation



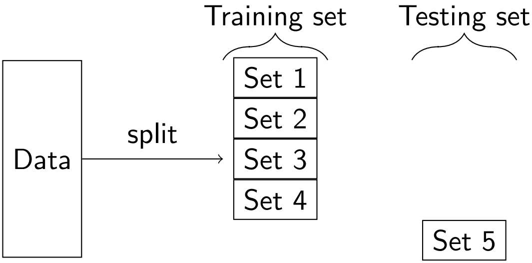
Cross validation



Cross validation



Cross validation



Results

	Configuration	Error (ms)
	constant value	2.013 ± 0.024
Decision trees	no periods, w/o bagging	2.075 ± 0.014
Decision trees	no periods, with bagging	2.067 ± 0.001
Decision trees	6 random periods, w/o bagging	2.019 ± 0.013
Decision trees	6 random periods, with bagging	2.015 ± 0.013
Decision trees	6 periods, w/o bagging	1.649 ± 0.154
Decision trees	6 periods, with bagging	1.123 ± 0.009
Neural nets	no periods, w/o subnets	2.014 ± 0.034
Neural nets	no periods, with subnets	2.012 ± 0.019
Neural nets	6 random periods, w/o subnets	1.924 ± 0.176
Neural nets	6 random periods, with subnets	1.992 ± 0.059
Neural nets	6 periods, w/o subnets	0.954 ± 0.052
Neural nets	6 periods, with subnets	0.830 ± 0.031

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

Speedup of 40X compared to DiskSim (not counting trace load time)

Conclusions

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