Designing adaptive systems
by maintaining a mixture over a set of experts

Manfred K. Warmuth    Corrie Scalisi
Includes some earlier work with
Robert Gramacy, Scott Brandt and Ismail Ari

University of California - Santa Cruz

Dec 8, 2006 - NIPS workshop
Last update - December 13, 2006
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
1. Disk spindown problem

- When to spin down the disk on your laptop?
- Best time-out time/user/usage dependent

![First 3000 iterations of the Cello-2 Dataset]
Non-convex loss
2. Caching

- Whenever small, fast memory and larger, slower secondary memory
- Keep objects in faster memory which likely to be needed again soon
  - Hit if requested object resides in cache
  - Miss otherwise
Caching Policies

- Decides which objects to discard to make room for new requests

- 7 common policies: LRU, RAND, FIFO, LIFO, LFU and MFU

- 5 fancy recent policies: SIZE, GDS, GD*, GDSF, LFUDA

- Criteria:
  - Recency and frequency of access
  - Size of objects
  - Cost of fetching object from secondary memory

- De facto standard: LRU
Which Policy to Choose?

- **For which situation?**
  - Disk access on PC
  - Web proxy access via browser
  - File server on local network
  - Middle of the night during backup
  - Application as well as time dependent

- Choosing one is **suboptimal**

- All policies claimed to be on-line/adaptive
Two example problems

Characteristics Vary with Time

![Graph showing characteristics varying with time over time intervals from 205000 to 235000]
Two example problems

Best Policy Varies with time

![Graph showing varying best policies over time](image-url)
Outline

1. Two example problems

2. Measuring the on-lineness of the data

3. The expert framework

4. Shifting experts

5. Experimental results

6. Wrap up
First trick: Permute the data

- Data not on-line if permuting does not change things
- Algorithm not adaptive if same performance on permuted data
Measuring the on-lineness of the data

Permuting trick for disk spindown data

on-line :-)

not on-line :-(
Measuring the on-lineness of the data

Permuting caching data

highly on-line data

some caching policies already on-line
Good comparators?

- As good as BestFixed chosen in hind-sight
- But BestFixed does not capture on-lineness of data
  - Same performance on original and permuted data
BestShift($K$) for spindown problem

Comparator:
- Partition of the timeline into segments
- Best in each segment

| 2 | 4 | 7 |

![Diagram showing average energy vs. total # of shifts with a point representing the partition.](image-url)
Measuring the on-lineness of the data

BestFixed\((K)\)

Dynamic programming: \(O(KN^2T)\)

where \(K\) \# of partitions, \(N\) \# of discrete idle times, \(T\) \# of trials

[H]
Measuring the on-lineness of the data

BestShift curves

BestShift(K) on Cello-2 Data
50 experts exponentially spaced between 0 and 10

BestShift(K) on Intel dataset
50 experts exponentially spaced between 0 and 10

on-line

not on-line
Comparators for caching

- **BestFixed**: *a posteriori best* of 12 policies on entire request stream
- **BestRefetching**\((R)\):
  minimum number of misses with at most \(R\) refetches
  in any sequence of switching policies
Comparator: All sequences of the form

We plot miss rate v.s. refetches:
Measuring the on-lineness of the data

**BestRefetching**($R$)

Dynamic programming: $O(RN^2T)$
Goal for on-line algorithms

- Beat BestFixed (easy)
- Get close to BestShift / BestRefetching
- In caching reduce I/O’s and end-user latency
- Fast algorithms
Score card for caching algorithms

- Miss Rate (%)
- Refetches as % of Total Requests

- A: Total I/Os less than BestFixed
- B: Total I/Os less than LRU
- C: Total I/O more than LRU

- +: Better than BestRefetching
- -: Worse than BestFixed

miss + refetch ≤ LRU miss

miss + refetch ≤ BestFixed miss

Designing adaptive systems by maintaining a mixture over a set of experts

M. Warmuth (UCSC)
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
What experts?

Caching:
- 12 caching policies

Disk spin down:
- Discretize interval $[0, \text{spindowncost}]$
On-line algorithm for learning as well as best experts

One weight per expert

- Represent confidence of master algorithm in expert
- Master algorithm predicts with convex combination of experts
- Loss update: \( w_i^{t+1} \sim \frac{w_i^t e^{-\eta L_i^t}}{Z_t} \)
- Designed to do well against BestFixed
- In some cases \( \log N \) regret
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
Shifting experts

As well as best partition

| 2 | 4 | 7 |

- **Loss Update** follows too well
- Follow it by **Share Update**:
  - Mix in small in $\alpha = 5\%$ times past average weight
  - Updates recover after each shift
  - Faster recovery if expert was used before
  - In some cases regret = $\#$ of bits to encode best partition
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
Spindown results

**Experimental results**

Spindown results

M. Warmuth (UCSC)

Designing adaptive systems by maintaining a mixture over a set of experts

30 / 37
Experimental results

Caching - we “Tracks” best policy
Experimental results

WWk

WWk Master and Comparator Missrates

8.6% = LRU missrate
2.0% = Obligatory missrate

BestRefetching(R)
Rank Ideal
Rank 60% Ideal
Rank 40% Ideal
BestFixed = SIZE
AllVC

Missrates %

Refetches as % of Total Requests
UMo Master and Comparator Missrates

16.6% = LRU missrate
1.5% = Obligatory missrate

- BestRefetching(R)
- Rank Ideal
- Rank 60% Ideal
- Rank 40% Ideal
- BestFixed = GDS
- AllVC
Experimental results

SMoLRU Master and Comparator Missrates

59.8% = LRU missrate
15.3% = Obligatory missrate

Missrates %

Refetches as % of Total Requests

BestRefetching(R)
- Rank Ideal
- Rank 60% Ideal
- Rank 40% Ideal
- BestFixed = SIZE
- AllVC
Outline

1. Two example problems
2. Measuring the on-lineness of the data
3. The expert framework
4. Shifting experts
5. Experimental results
6. Wrap up
Pushing the theoretical analysis

Disk spindown:
- Non-convex loss, but in each trial only two loss values
- Experts are sorted
- Analyze with continuously many experts

Caching:
- Prove bounds for ARCing
The upshot

- Measure on-lineness of data
- Design algorithms that provably exploit on-lineness
- Many simple on-line problems amenable to theoretical analysis