Real-time prediction using online learning:

Application to energy management in wireless networks

Claire Monteleoni (UCSD), Hari Balakrishnan (MIT), Nick Feamster (Georgia Tech), and Tommi Jaakkola (MIT)*

Abstract:

We showcase an original online learning algorithm, in an application to energy management in wireless networks. The goal is to manage an energy/performance tradeoff in IEEE 802.11 devices, using real-time prediction. The algorithm adapts to changing observations by tracking periods of stationarity, and simultaneously learning the level of non-stationarity (e.g. burstiness), online. Network properties can vary both with time and location, making this an appropriate application. We simulate our algorithm on a mobile wireless 802.11 node, yielding encouraging empirical results.

Online learning:

A useful model for many settings:
- Forecasting and real-time predictions (e.g. stock market, internet).
- Online classification (e.g. spam filtering, fraud detection).
- Streaming applications (e.g. high-dimensional, or real-time data).
- Resource-constrained learning (e.g. on small devices).

Online learning framework:

1. *Access to data is one-at-a-time only.*
   Once a data point is seen, it might not be seen again.
   Predictions are required in real-time (no training period).
2. *Time and memory use must not scale with the data.*
   Computation must be cheap, and light-weight:
   Must not store all the data seen so far, to use a "batch" method.

Algorithms:

We give a general online learning algorithm for regression/estimation, or classification:
- data need not be perfectly separable
- works for learning many hypothesis classes

We operate in the *non-stochastic* setting: no assumptions on the observations.
Could even be generated online, by an adaptive adversary!

Consider an algorithm that observes the predictions of a set of "experts," and predicts based on a probability distribution $p_t(i)$ over experts, representing how well each expert has been performing recently.
Prediction loss of expert *i*, $L(i, t)$, defined based on problem objective (modular).
Perform Bayesian updates: $p_{t+1}(i) \propto p_t(i)e^{-L(i,t)}$.

To model changing regimes (non-stationarity), maintain probability distribution via an HMM, with the identity of the current best expert as the hidden state.
Equate $L(i, t)$ with neg. log-likelihood of observation, given expert's prediction.
Then perform Bayesian updates: $p_{t+1}(i) \propto \sum_j p_t(j)e^{-L(j,t)}p(i|j)$

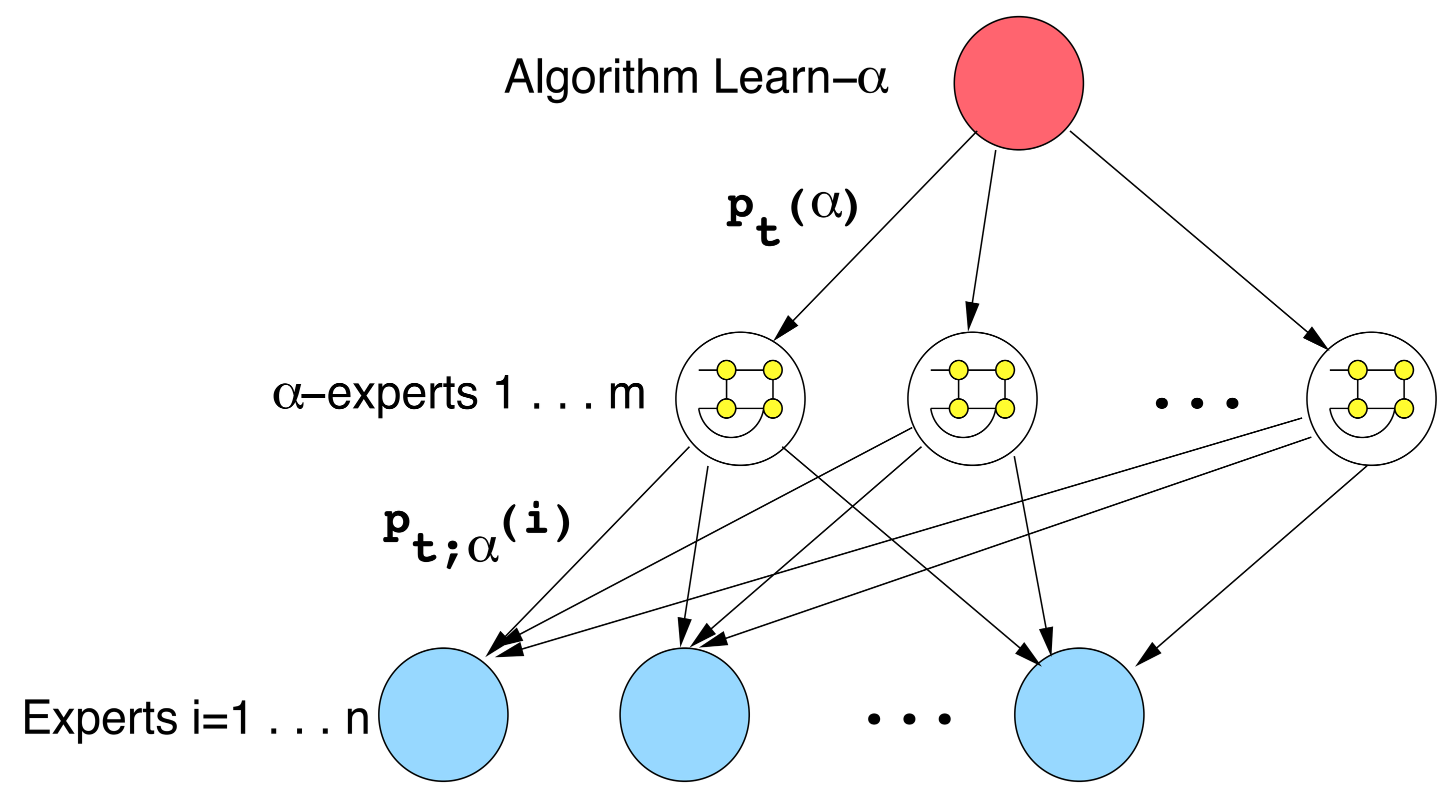
Transition dynamics: model of how the current best expert can change over time:

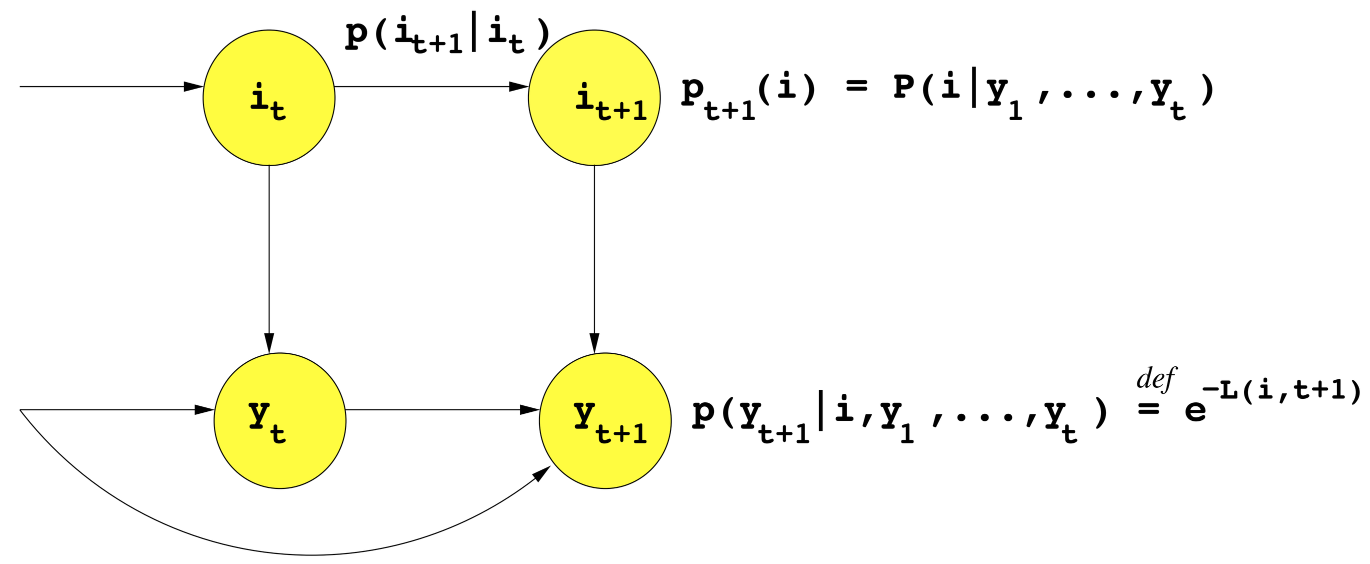
$$P(i|j;\alpha) = \begin{cases} (1-\alpha) & i = j \\ \frac{\alpha}{n-1} & i \neq j \end{cases}$$

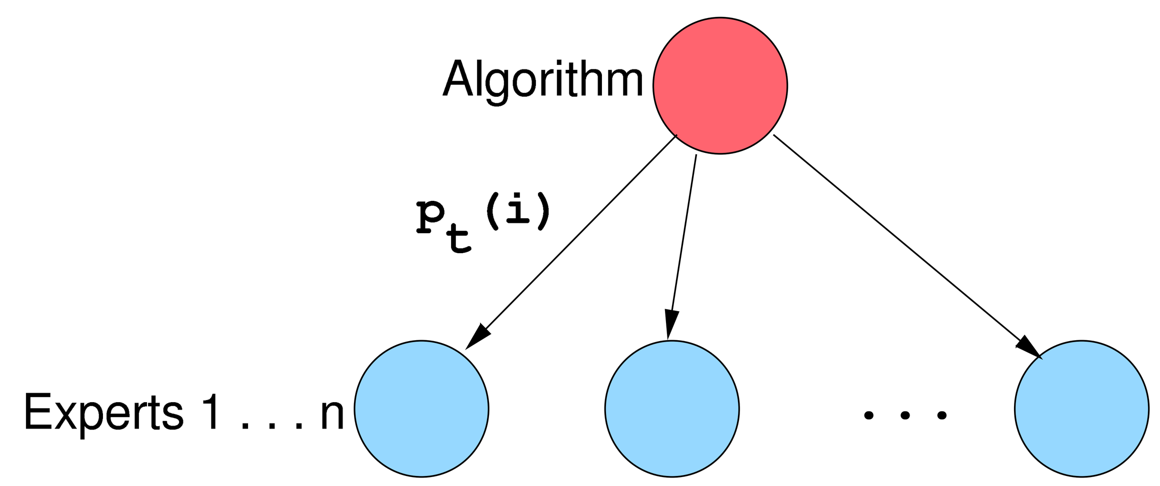
Learn level of non-stationarity, α, online, while performing original learning task!
Define a set of meta-experts, each updating with a different value of α.
Algorithm Learn-α maintains a distribution over α-experts, and uses Bayesian updates to track the best fixed α. $p_t(\alpha_j) \propto p_{t-1}(\alpha_j)e^{-L(\alpha_j,t)}$

Acknowledgments:

* Work done while all the authors were at the MIT Computer Science and Artificial Intelligence Laboratory.







Application to wireless:

Energy/Latency tradeoff for IEEE 802.11 wireless nodes:
- Awake state consumes too much energy.
- Sleep state cannot receive packets.

IEEE 802.11 Power Saving Mode:
- Base station buffers packets for sleeping node.
- Node wakes at regular intervals (T = 100*ms*) to process buffered packets, I_t.
- Latency is introduced due to buffering.

Apply Learn-α to adapt sleep duration, T_t, to changes in network activity.
Simultaneously learn change rate (non-stationarity) online.

Experts:
- 10 experts, each a fixed polling time from 100-1000*ms*, in multiples of 100*ms*.
- For example, the 802.11 protocol of 100*ms*, is one expert.

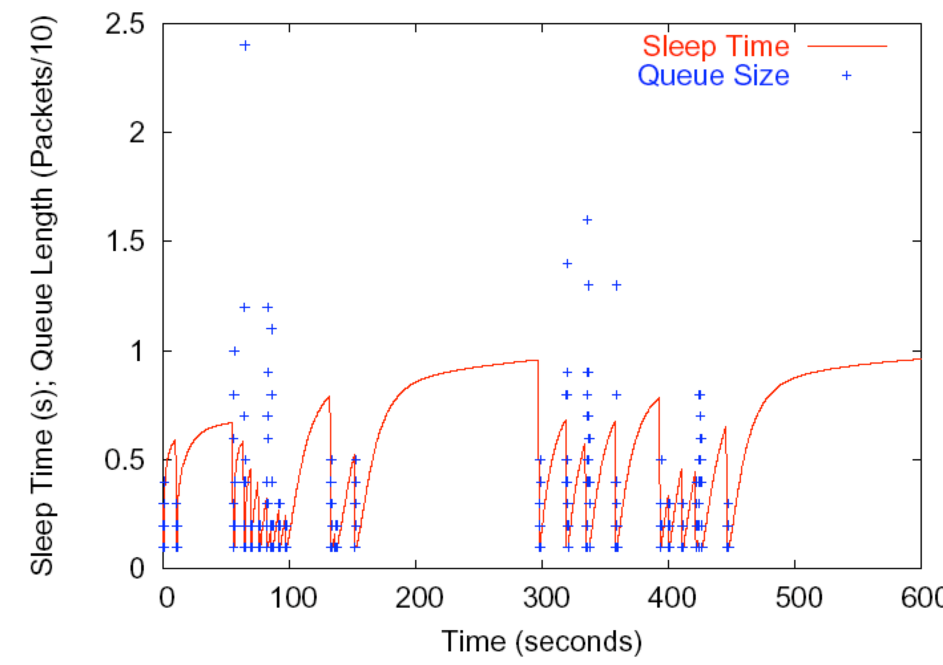
Loss function:
Optimize tradeoff by minimizing a function convex in both latency and energy. Algorithm is modular w.r.t. loss.
For example: $\gamma\frac{T_t I_t}{2} + \frac{1}{T_t}$

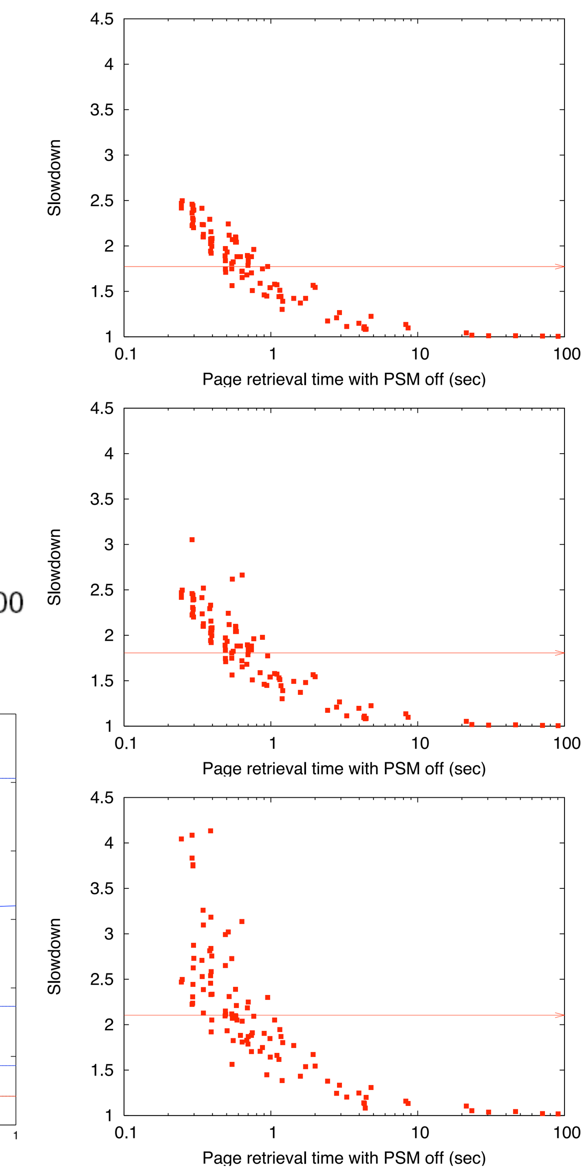
First term: average latency for buffering I_t bytes.
Second term: energy usage relates inversely to sleep time.

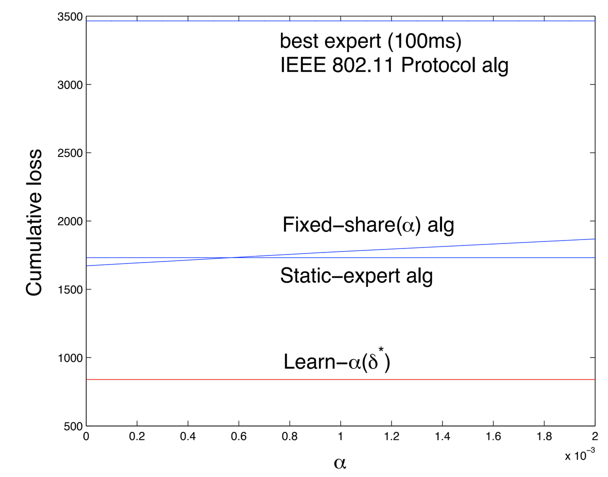
```
Algorithm LPSM
```
Initialization:
$\forall j,\ p_1(j) \leftarrow \frac{1}{m}$
$\forall i,j,\ p_{1,j}(i) \leftarrow \frac{1}{n}$
Upon tth wakeup:
$T_t \leftarrow$ number of ms just slept
$I_t \leftarrow$ # bytes stored at neighbor
Retrieve buffered data
For each $i \in \{1\dots n\}$:
  $\text{Loss}[i] \leftarrow \gamma\frac{I_t T_i^2}{2T_t} + \frac{1}{T_i}$
For each $j \in \{1\dots m\}$:
  $\text{AlphaLoss}[j] \leftarrow -\log\sum_{i=1}^n p_{t,j}(i)\,e^{-\text{Loss}[i]}$
  $p_{t+1}(j) \leftarrow p_t(j)e^{-\text{AlphaLoss}[j]}$
  For each $i \in \{1\dots n\}$:
    $p_{t+1,j}(i) \leftarrow \sum_{k=1}^n p_{t,j}(k)\,e^{-\text{Loss}[k]}\,P(i|k;\alpha_j)$
  Normalize $P_{t+1,j}$
  $\text{PollTime}[j] \leftarrow \sum_{i=1}^n p_{t+1,j}(i)\,T_i$
Normalize P_{t+1}
$T_{t+1} \leftarrow \sum_{j=1}^m p_{t+1}(j)\ \text{PollTime}[j]$
Goto sleep for T_{t+1} ms.

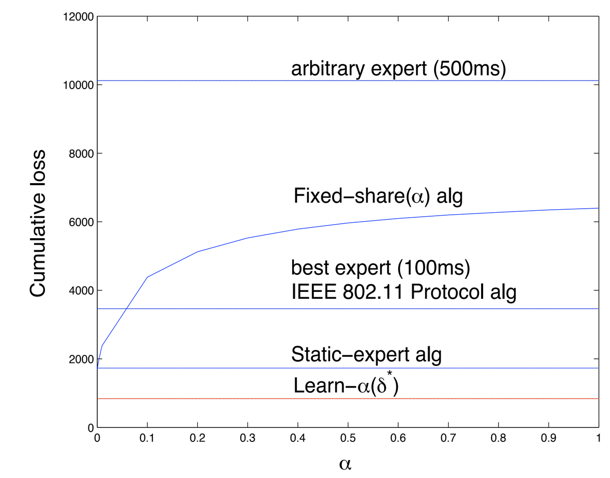
Results:

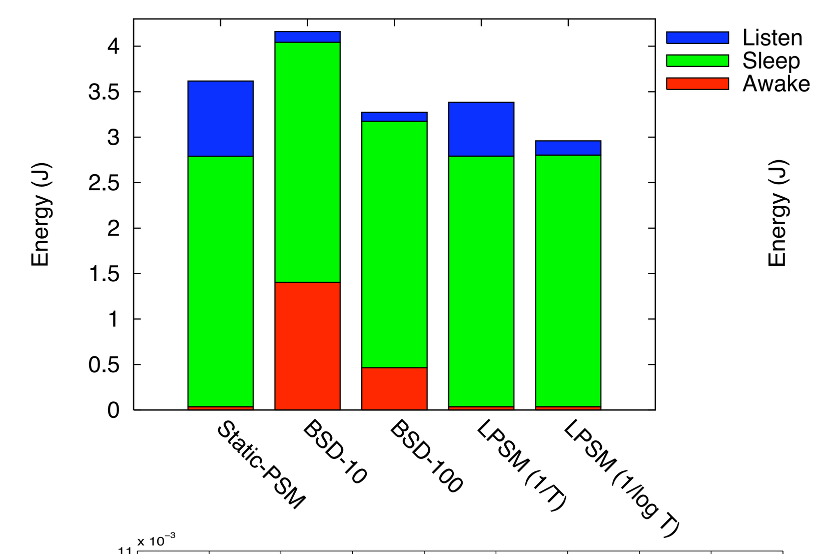
Energy usage: reduced by 7-20% from 802.11 PSM.
Average latency 1.02x that of 802.11 PSM. For details, see paper, and below:

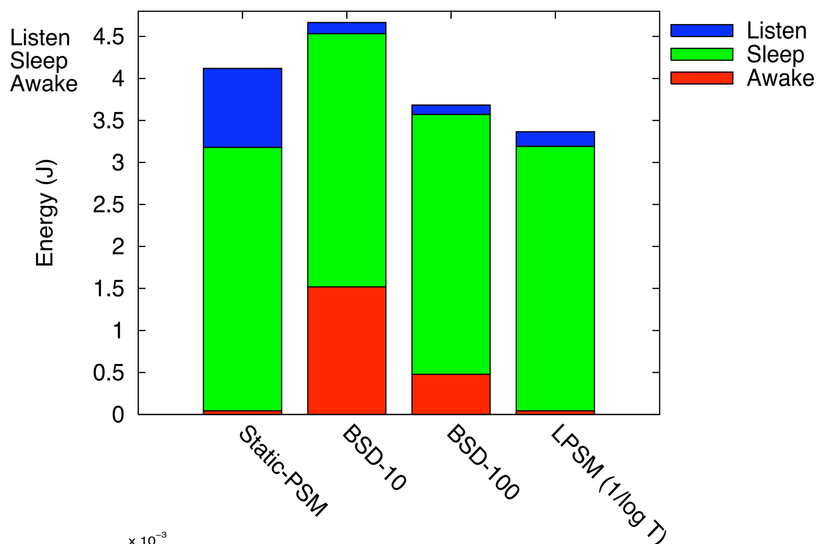


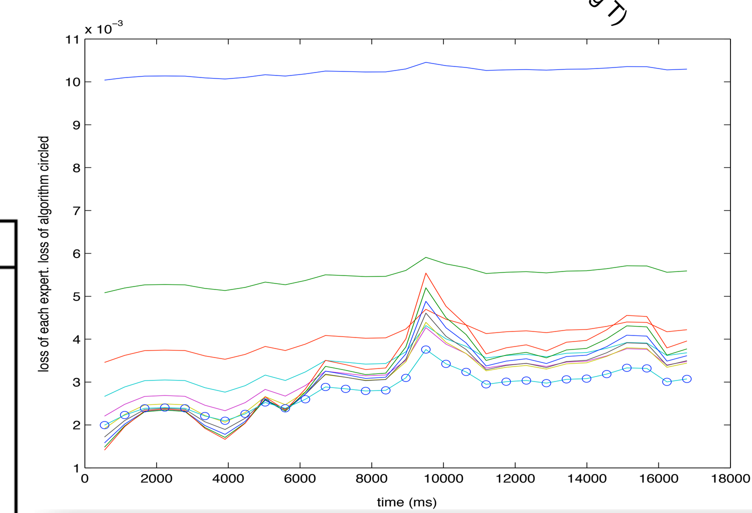


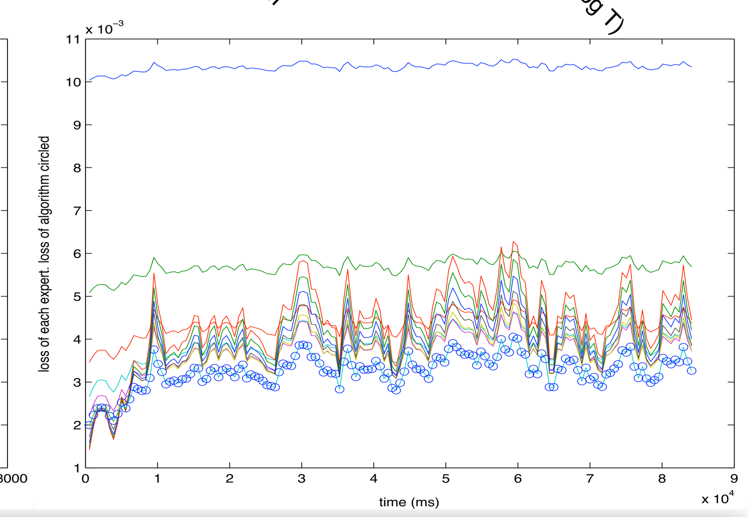












References:

This poster is based on work in several papers, and the first author's PhD thesis. For further reading, please see:

C.E. Monteleoni, "Learning with Online Constraints: Shifting Concepts and Active Learning," PhD Thesis in Computer Science, MIT, 2006.

C. Monteleoni, H. Balakrishnan, N. Feamster, and T. Jaakkola, "Managing the 802.11 Energy/Performance Tradeoff with Machine Learning." MIT-LCS-TR-971, 2004.

C. Monteleoni and T. Jaakkola, "Online Learning of Non-stationary Sequences," in Advances in Neural Information Processing Systems (NIPS) 16, 2003.

C.E. Monteleoni, "Online Learning of Non-stationary Sequences," SM Thesis in Computer Science, MIT, 2003.

Available at: http://people.csail.mit.edu/cmontel