Online Learning and Bregman Divergences
Tutorial at Machine Learning Summer School
Taipei, Taiwan, July 2006
Manfred K. Warmuth
University of California at Santa Cruz, USA
http://www.cse.ucsc.edu/~manfred

Help with the tutorial: Gunnar Rätsch

Content of this tutorial

- P I: Introduction to Online Learning
  - The Learning setting
  - Predicting as good as the best expert
  - Predicting as good as the best linear combination of experts
- P II: Bregman divergences and Loss bounds
  - Introduction to Bregman divergences
  - Relative loss bounds for the linear case
  - Nonlinear case & matching losses
  - Duality and relation to exponential families
  - On-line algorithms motivated by game theory
- P III: on-line to batch conversion, applications
  - Simple conversions
  - Caching and the disk spin down problem
  - Other applications and conclusion

Goal: How can we prove relative loss bounds?

Sources of Information

- Emphasizes "potentials" - Here "Bregman Divergences"
- JMLR, JML, COLT, NIPS, ICML

Predicting almost as good as the best expert

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>...</th>
<th>$E_n$</th>
<th>prediction true label loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>day 1</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>day 2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>day 3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>day $t$</td>
<td>$x_{t,1}$</td>
<td>$x_{t,2}$</td>
<td>$x_{t,3}$</td>
<td>...</td>
<td>$x_{t,n}$</td>
</tr>
</tbody>
</table>

Master Algorithm

For $t = 1$ To $T$ Do
Get instance $x_t \in \{0,1\}^n$
Predict $\hat{y}_t \in \{0,1\}$
Get label $y_t \in \{0,1\}$
Incur loss $|y_t - \hat{y}_t|$
**More general online settings**

**Protocol:**
For $t = 1$ to $T$
  - Get instance $x_t \in \mathbb{R}^n$
  - Predict $\hat{y}_t \in \mathcal{Y}$
  - Get label $y_t \in \mathcal{Y}$
  - Incur loss $L(y_t, \hat{y}_t, x_t)$

**Problem instances:**
- Classification: $\hat{y}_t, y_t \in \{0, 1\}$, e.g. $L(y, \hat{y}, x) = |y - \hat{y}|$
- Regression: $\hat{y}_t, y_t \in \mathbb{R}$, e.g. $L(y, \hat{y}, x) = (y - \hat{y})^2$
- Density estimation: no label, e.g. $L(y, \hat{y}, x) = -\log P(x | \theta)$

**Goal:** small total loss $\sum_t L(y_t, \hat{y}_t, x_t)$

**Expert setting: Halving Algorithm [BF]**

- Predicts with majority
- If mistake is made, then number of consistent experts is (at least) halved
- Any mistake is “converted” into knowledge on the learning problem: mistake driven learning

**A run of the Halving Algorithm**

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
<th>$E_6$</th>
<th>$E_7$</th>
<th>$E_8$</th>
<th>Majority</th>
<th>True label</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>↑</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>consistent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For any sequence with a consistent expert
Halving Algorithm makes at most $\leq \log_2 n$ mistakes

(Good bound, but not optimal)

**What if no expert is consistent?**

Sequence of examples $S = (x_1, y_1), \ldots, (x_T, y_T)$
- Total loss of algorithm $A$: $L_A(S) = \sum_{t=1}^T L(A(x_t), y_t)$
- Total loss of $i$-th expert $E_i$: $L_i(S) = \sum_{t=1}^T L(E_i(x_t), y_t)$

Want bounds of the form:

$\forall S : L_A(S) \leq a \min_i L_i(S) + b \log(n)$

where $a, b$ are constants

Bounds loss of algorithm relative to loss of best expert
Weighted Majority Algorithm [LW]

One weight per expert
Can’t wipe out experts!

- Predicts with larger side
- Weights of wrong experts are multiplied by $\beta \in [0, 1)$

Number of mistakes of the WM algorithm

\[ M_{t,i} = \# \text{ of mistakes of } E_i \text{ before trial } t \]
\[ w_{t,i} = \beta^{M_{t,i}} \text{ weight of } E_i \text{ at beginning of trial } t \]
\[ W_t = \sum_{i=1}^{n} w_{t,i} \text{ total weight at trial } t \]

Minority $\leq \frac{1}{2} W_t$, Majority $\geq \frac{1}{2} W_t$

If no mistake then minority multiplied by $\beta$:
\[ W_{t+1} \leq W_t \]

If mistake then majority multiplied by $\beta$:
\[ W_{t+1} \leq 1 + \frac{1}{2} W_t + \beta \cdot \frac{1}{2} W_t = \frac{1+\beta}{2} W_t \]

Relative Loss bound for Weighted Majority

Solving for $M$:
\[ M \leq \frac{\ln \left( \frac{1}{\beta} \right) M_i + \ln n}{\ln \left( \frac{2}{1+\beta} \right) n} \]
\[ \beta = \frac{1}{e} \]

For all sequences, loss of master algorithm is comparable to loss of best expert
\[ \Rightarrow \text{Relative loss bounds} \]

\[ \text{Fr} \]
Other Loss Functions

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute loss</td>
<td>( L(y, \hat{y}) =</td>
</tr>
<tr>
<td>square loss</td>
<td>( L(y, \hat{y}) = (y - \hat{y})^2 )</td>
</tr>
<tr>
<td>entropic loss</td>
<td>( L(y, \hat{y}) = y \ln \frac{\hat{y}}{y} + (1 - y) \ln \frac{1 - \hat{y}}{1 - y} ), ( y, \hat{y} \in [0, 1] )</td>
</tr>
<tr>
<td>entropic loss ±</td>
<td>( L(y, \hat{y}) = \frac{1 + y}{2} \ln \frac{y}{1 - y} + \frac{1 + \hat{y}}{2} \ln \frac{1 - y}{1 - \hat{y}} ), ( y, \hat{y} \in [-1, +1] )</td>
</tr>
<tr>
<td>hellinger loss</td>
<td>( L(y, \hat{y}) = \frac{1}{2} (\sqrt{1-y} - \sqrt{1-\hat{y}})^2 + \frac{1}{2} (\sqrt{y} - \sqrt{\hat{y}})^2 ), ( y, \hat{y} \in [0, 1] )</td>
</tr>
</tbody>
</table>

How does it work with other loss functions?

One weight per expert:

\[ w_{t,i} = \beta L_{t,i} = e^{-\eta L_{t,i}} \]

where \( L_{t,i} \) is total loss of \( E_i \) before trial \( t \) and \( \eta \) is a positive learning rate

Master predicts with the weighted average (WA)

\[ v_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}} \]

normalized weights

\[ \hat{y}_t = \sum_{i=1}^{n} v_{t,i} x_{t,i} = v_t \cdot x_t \]

where \( x_{t,i} \) is the prediction of \( E_i \) in trial \( t \)

Bounds for other Loss Functions

\forall \text{ sequences } S \text{ of examples } \langle (x_t, y_t) \rangle_{1 \leq t \leq T} \text{ where } x_t \in [0,1]^n \text{ and } y_t \in [0,1] \]

\[ L_{WA}(S) \leq \min_i \frac{1}{a} L_i(S) + 1/\eta \ln(n) \]

\[ L_{WA}(S) - \min_i L_i(S) \leq b \ln(n) \]

Worst-case regret bounds

<table>
<thead>
<tr>
<th>b</th>
<th>WA</th>
<th>fancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>entropic</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>square</td>
<td>2</td>
<td>1/2</td>
</tr>
<tr>
<td>hellinger</td>
<td>1</td>
<td>0.71</td>
</tr>
</tbody>
</table>

- Improved constants of \( b = 1/\eta \) when Master uses fancier pred. [V]
- For the discrete loss and the absolute loss: \( a > 1 \)
**Proof**

Potential: $-\frac{1}{\eta} \ln W_2$

Key inequality: $L(y, v_t \cdot x_t) \leq \frac{-1}{\eta} \ln W_{t+1} - \left( \frac{1}{\eta} \ln W_t \right)$

Telescoping:

$$L_{WA}(S) \leq -\frac{1}{\eta} \ln \frac{W_{T+1}}{W_1} = -\frac{1}{\eta} \ln \sum_{j=1}^{n} \frac{1}{n} e^{-\eta L_{T+1,j}(S)} = -\frac{1}{\eta} \ln \frac{1}{n} e^{-\eta L_{\arg\min_j L_{T+1,j}}(S)} = L_{T+1,i}(S) + \frac{1}{\eta} \ln n$$

**Usefulness:**

- Easy to combine many pretty good experts (algorithms) so that Master is guaranteed to be almost as good as the best
- Bounds logarithmic in number of experts (multiplicative updates)

**Questions:**

- How to obtain algorithms that do well compared to best linear combination or best thresholded linear combination of experts?
- How to motivate the updates?
- What are good measures of progress?
- What are good loss functions?
- Methods for proving relative loss bounds?
Example: Learning Disjunctions of Experts

<table>
<thead>
<tr>
<th>variables/experts</th>
<th>true label</th>
<th>$E_1 \lor E_3$</th>
<th>$E_3 \lor E_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1$</td>
<td>$E_2$</td>
<td>$E_3$</td>
<td>$E_4$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$E_1 \lor E_3$ becomes $u = (1, 0, 1, 0)$

$E_1 \lor E_3$ is one on $x_t \in \{0, 1\}^n$ iff $u \cdot x_t \geq 1$

Weighted Majority on k-literal Disjunctions

One expert per disjunction

$\binom{n}{k}$ weights

Do as well as best $k$ out of $n$ literal (monotone) disjunction

\[
\text{# of mistakes of WM} \leq 2.63 M + 2.63 k \ln \frac{n}{k}
\]

$M$ is # of mistakes of best

Time (and space) exponential in $k$

Efficient algorithm have only one weight per literal instead of one weight per disjunction

The Perceptron Algorithm

In trial $t$: Get instance $x_t \in \{0, 1\}^n$

If $w_t \cdot x_t \geq \theta$ then $\hat{y}_t = 1$

else $\hat{y}_t = 0$

Get label $y_t \in \{0, 1\}$

If mistake then

$w_{t+1} = w_t - \eta (\hat{y}_t - y_t) x_t$

Rotation invariant if $w_1 = (0, \ldots, 0)$

$k$-literal Disjunctions with Perceptron

Perceptron Convergence Theorem ($w_1 = (0, \ldots, 0)$, $\theta = \frac{1}{2}$, $\eta = \frac{1}{2^n}$)

\[
\text{# of mistakes} \leq 4 A + 4 k n
\]

where $A$ is # of attribute errors of best disjunction of size $k$, i.e., the minimum # of attributes that need to be flipped to make the disjunction consistent

$A \leq kM$

Lower bound for rotation invariant algorithms: [KWA]

\[
\text{# mistakes} = \Omega(n)
\]
The Winnow Algorithm [L]

In trial $t$: Get instance $x_t \in \{0,1\}^n$
If $w_t \cdot x_t \geq \theta$ then $\hat{y}_t = 1$
else $\hat{y}_t = 0$
Get label $y_t \in \{0,1\}$
If mistake then
$w_{t+1,i} = w_{t,i} e^{-\eta(y_t - \hat{y}_t)x_{t,i}}$

Mistake bound ($w_1 := \frac{k}{n}(1,\ldots,1)$, $\theta = \frac{3\ln 3}{8} n$, $e^{-\eta} = \frac{1}{3}$) [AW]

So far
- Learning relative to best expert for various loss functions
- Learning relative to best disjunction
- Perceptron versus Winnow and expansion into feature space

On-line Linear Regression

For $t = 1,\ldots,T$ do
Get instance $x_t \in \mathbb{R}^n$
Predict $\hat{y}_t = w_t \cdot x_t$
Get label $y_t \in \mathbb{R}$
Incur loss $L_t(w_t) = (y_t - \hat{y}_t)^2$
Update $w_t$ to $w_{t+1}$

Examples of Updates

Gradient descent ($w \in \mathbb{R}^n$)

$w_{t+1} = w_t - \eta \nabla L_t(w_t)$

$= w_t - \eta(\hat{y}_t - y_t)x_t$ [WH]

Exponentiated Gradient Algorithm [KW]
(w is probability vector)

$w_{t+1,i} = w_{t,i} \exp \left[ -\eta \frac{\partial L_t}{\partial w_{t,i}} \right] / \text{normalization}$
More examples of Updates

Unnormalized Exponentiated Gradient Algorithm \( [KW] \)
\( (w \geq 0) \)
\[
 w_{t+1,i} = w_{t,i} \exp \left[ -\eta \frac{\partial L_t(w_t)}{\partial w_{t,i}} \right]
\]

Binary Exponentiated Gradient Algorithm \( [By] \)
\( (w \in [0,1]^n) \)
\[
 w_{t+1,i} = \frac{w_{t,i} \exp \left[ -\eta \frac{\partial N_t(w_t)}{\partial w_{t,i}} \right]}{1 - w_{t,i} + w_{t,i} \exp \left[ -\eta \frac{\partial N_t(w_t)}{\partial w_{t,i}} \right]}
\]

Motivation of Updates \( [KW] \)

Gradient descent
\[
 w_{t+1} = \arg\min_w \left( \|w - w_t\|^2_2 + \eta (y_t - w \cdot x_t)^2/2 \right)
\]
\[
 w_{t+1} = w_t - \eta (w_{t+1} \cdot x_t - y_t) x_t \approx w_t \cdot x_t
\]

Exponentiated Gradient Algorithm
\[
 w_{t+1} = \arg\min_w \left( \sum_{i=1}^n w_i \ln \frac{w_i}{w_{t,i}} + \eta (y_t - w \cdot x_t)^2/2 \right)
\]
\[
 w_{t+1} = w_{t,i} \exp \left[ -\eta \left( w_{t+1} \cdot x_t - y_t \right) \right] \right] / \text{normalization}
\]

More examples of Updates (cont)

\( p \)-norm Algorithms \( [GLS,GL] \)
(\( w \in \mathbb{R}^n \))
\[
 w_{t+1} = f^{-1}(f(w_t) - \eta \nabla L_t(w_t) )
\]

where
\[
 f(w) = \frac{1}{2} \|w\|^2_2 = \frac{1}{2} \left( \sum_i |w_i|^q \right)^{2/q}
\]
and \( q \) dual to \( p \) (i.e., \( \frac{1}{p} + \frac{1}{q} = 1 \))

- \( p = 2 \) becomes gradient descent
- \( p = O(\log n) \) becomes EG-like algs
- \( 2 < p < O(\log n) \) interpolates between the two extremes

Families of update algorithms

| parameter $||w-w_t||^2_2$ | name of family | update algorithms |
|--------------------------|----------------|-------------------|
| “divergence” Gradient Descent Linear Least Squares. Backpropagation Perceptron Algorithms kernel based algorithms,... |
| $\sum_{i=1}^n w_i \ln \frac{w_i}{w_{t,i}}$ | Exponentiated Gradient Algorithm expert algs / Bayes update Normalized Winnow “AdaBoost” |
 Families of update algorithms (cont)

<table>
<thead>
<tr>
<th>“divergence”</th>
<th>name of update family</th>
<th>update algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sum_{i=1}^n w_i \ln \frac{w_i}{w_{t,i}} + w_{t,i} - w_i$</td>
<td>Unnormalized</td>
<td>Winnow</td>
</tr>
<tr>
<td>$\sum_{i=1}^n w_i \ln \frac{w_i}{w_{t,i}} + (1 - w_i) \ln \frac{1 - w_i}{1 - w_{t,i}}$</td>
<td>Binary</td>
<td>Exp. Grad. Alg.</td>
</tr>
</tbody>
</table>

Members of different families exhibit different behavior

$||||^2_2$ versus entropic regularization

 Alternate motivation: continuous updates [WJ]

Continuous time $\Rightarrow$ Ordinary differential equations

**Gradient Descent**

$w \in \mathbb{R}^n$

$$\dot{w}_t = -\eta \nabla w L_t(w_t)$$

**Unnormalized Exponentiated Gradient Alg.**

$w \geq 0$

$$\log(w_t) = -\eta \nabla w L_t(w_t)$$

 Alternate motivation: continuous updates (cont)

Discretization

$$f(w_{t+h}) - f(w_t) \approx \frac{-\eta \nabla L_t(w_t)}{h}$$

$$w_{t+h} = f^{-1}(f(w_t) - \eta h \nabla w L_t(w_t))$$

We use $h = 1$

$$w_{t+1} = f^{-1}(f(w_t) - \eta \nabla L_t(w_t))$$

Conjecture: **Forward Euler** better:

Replace $\nabla w L_t(w_t)$ by $\nabla w L_t(w_{t+h})$

 Two main families

- **Additive**: $w = \sum_i \alpha_i x_i$
- **Multiplicative**: $w_i \sim \exp(\sum_j \alpha_j x_{i,j})$

$||||_2^2$ entropic

- Gradient descent
- Exponentiated Gradient
- Rotation invariant

- We now give a problem that favors the multiplicative updates

Linear versus logarithmic dependence in $n$
Let’s keep it simple

Linear Regression

- Examples \((x_t, y_t)\)
- Linear hypothesis \(w\)
- Predicts with \(\hat{y}_t = w \cdot x_t\)

A hard problem?

Hadamard Matrix:

\[
\begin{array}{cccc}
\rightarrow & +1 & +1 & +1 & +1 \\
\text{instances} & \rightarrow & +1 & -1 & +1 & -1 \\
& \rightarrow & +1 & +1 & -1 & -1 \\
& \rightarrow & +1 & -1 & -1 & +1 \\
& \uparrow & \uparrow & \uparrow & \uparrow \\
\text{targets}
\end{array}
\]

- \(n\) instances and \(n\) targets
- Instances are orthogonal
- Target weight vectors are units

Without embeddings I

Any linear combination of \(k\) training instances predicts zero on all \(n - k\) test instances \[\text{LLW95, KWA97}\]
So loss 1 on \(n - k\) of the \(n\) instances
Average square loss over all \(n\) instances is \(\geq 1 - \frac{k}{n}\)

Without embeddings II

Theorem

For any linear combination of \(k\) rows of the \(n\)-dimensional Hadamard matrix and any of the \(n\) targets
the average square loss over all \(n\) instances is
\[\geq 1 - \frac{k}{n}\]
**Spiffing it up**

Embed instance into a feature space

\( \phi : \mathbb{R}^n \rightarrow \mathbb{R}^m \)

\[
\phi(x_1, x_2) = (\sin(x_1), x_2)
\]

---

**The Kernel Trick** [BGV92]

If \( w \) linear combination of expanded instances, then

\[
\tilde{y} = \sum_t \alpha_t \phi(x_t) \cdot \phi(x) = \sum_t \alpha_t \phi(x_t) \cdot \phi(x)
\]

Kernel function \( K(x_t, x) \) often efficient to compute

\[
\phi\left(\frac{x_1, \ldots, x_n}{n}\right) = \left(1, \ldots, x_i, \ldots, x_i, x_j \ldots, x_j, x_k \ldots\right)
\]

2^n products

Kernel magic

\[
K(x, z) = \phi(x) \cdot \phi(z) = \sum_{I \subseteq 1..n} \prod_{i \in I} x_i \prod_{i \notin I} z_i = \prod_{i=1}^n (1 + x_i z_i)
\]

\( O(2^n) \) time \( O(n) \) time

---

**Good news**

Many of our favorite algorithms can be “kernelized”:

Linear Least Squares, Widrow-Hoff, Support Vector Machines, PCA, Simplex Algorithm, ...

**Kernel Trick**

- Weight vector linear combination of embedded instances
- Individual features never accessed

---

**Linear combinations?**

**Representer Theorem:** [KW71]

\[
w = \arg \inf_{w'} \left( ||w'||^2 + \eta \sum_t (w' \cdot \phi(x_t) - y_t)^2 \right)
\]

Solution \( w \) linear combination of the \( \phi(x_t) \)

**Rotation invariance:** [KWA97]

Any algorithm whose predictions are not affected by rotating the instances in feature space

must predict with linear combination of embedded instances

**Sufficient conditions!**
Linear or non-linear?

- We give a problem for which kernel algorithms behave like linear algorithms
- Embeddings don’t help

:-(

Main Result

Theorem

- No matter how the instances are embedded
- No matter what $k$ training instances chosen by the learner
- No matter what linear combination used

For one of the targets
average square loss on all $n$ instances is $1 - \frac{k}{n}$

Proof idea

- Use SVD spectrum
- After $k$ instances weight space has rank $k$
  and only $k$ singular values can be "captured"

\[
\text{Avg. square loss} \geq \frac{1}{n^2} \sum_{i=k+1}^{n} s_i^2 \\
= 1 - \frac{k}{n}
\]

Additional Constraints

\[w_i \geq 0 \text{ and } \sum_{i=1}^{n} w_i = 1\]

- Problem matrix
  \[
  \begin{bmatrix}
  -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\
  -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\
  \end{bmatrix}
  \]

- For above $k$ instances, labeled by one of the $2^k$ columns, only consistent weight vector is unit identifying that column
- Weight space can have rank $2^k$
Rows of random ±1 matrix labeled by first column
Left: components of shortest weight vector over time
Right: same but with additional constraints

Maintain additional constraints?

Use Exponentiated Gradient Algorithm [KW97]

Kernel methods \( w = \sum \alpha_t \Phi(x_t) \)

\[
E_{G} w_j = \exp \sum_t \alpha_t \Phi(x_t) / \text{const}
\]

Now log weights linear combination of expanded instances

Average Squared Error

\[
E_{G} \text{kernel alg} s \frac{\ln(n)}{k} \quad 1 - \frac{k}{n} \quad \text{and} \quad (1 - \frac{1}{n})^k
\]
Incorporating side info

Kernel algorithms: none \( \geq 1 - k/n \)
EG: \( w_i \geq 0 \) and \( \sum_i w_i = 1 \)
\( O(\log n/k) \)
\[
\begin{align*}
\text{instances} & \quad \rightarrow +1.001 \quad +1.002 \quad +1.003 \quad +1.004 \\
\text{targets} & \quad \uparrow \\
\end{align*}
\]

Now target determined by any single example
Trivial algorithm beats EG

Which matrix?

Kernel matrix dot products of instances
Problem matrix instances as rows - targets as columns
- If eigen-spectrum of kernel matrix has heavy tail
  then kernel not useful
  - Picked wrong kernel
  - Problem too hard
- If svd-spectrum of problem matrix has heavy tail
  then problem not learnable

We showed:
- Hadamard problem matrix has heavy tail
- Adding random features makes tail of kernel matrix heavy

Questions

Gave problem that cannot be learned by kernel base algs
- What is the optimal kernel for a given problem?
- What is the hard problem for multiplicative updates?
- Can multiplicative updates be kernelized?
  Some cases are given by [TW]
- Can entropy regularization be replaced by \( \| \|_2 \) regularization
  plus non-negativity and total weight constraints?

Content of this tutorial

- P I: Introduction to Online Learning
  - The Learning setting
  - Predicting as good as the best expert
  - Predicting as good as the best linear combination of experts
- P II: Bregman divergences and Loss bounds
  - Introduction to Bregman divergences
  - Relative loss bounds for the linear case
  - Nonlinear case & matching losses
  - Duality and relation to exponential families
  - On-line algorithms motivated by game theory
- P III: on-line to batch conversion, applications
  - Simple conversions
  - Caching and the disk spin down problem
  - Other applications and conclusion

Goal: How can we prove relative loss bounds?
Bregman Divergences [Br,CL,Cs]

For any differentiable convex function $F$

$$\Delta_F(\tilde{w}, w) = F(\tilde{w}) - F(w) - (\tilde{w} - w) \cdot \nabla w F(w)$$

Bregman Divergences: Simple Properties

1. $\Delta_F(\tilde{w}, w)$ is convex in $\tilde{w}$
2. $\Delta_F(\tilde{w}, w) \geq 0$
   If $F$ convex equality holds iff $\tilde{w} = w$
3. Usually not symmetric: $\Delta_F(\tilde{w}, w) \neq \Delta_F(w, \tilde{w})$
4. Linearity (for $a \geq 0$):
   $\Delta_F(\tilde{w}, w) + a \Delta_F(\tilde{w}, w) = \Delta_F(\tilde{w}, w) + a \Delta_F(\tilde{w}, w)$
5. Unaffected by linear terms ($a \in \mathbb{R}$, $b \in \mathbb{R}^n$):
   $\Delta_H(\tilde{w}, w) = \Delta_H(\tilde{w}, w)$

Bregman Divergences: more properties

6. $\nabla \tilde{w} \Delta_F(\tilde{w}, w)$
   $$\nabla \tilde{w} \Delta_F(\tilde{w}, w) = \nabla F(\tilde{w}) - \nabla \tilde{w} (\tilde{w} \nabla w F(w))$$
   $$= f(\tilde{w}) - f(w)$$

7. $\Delta_F(w_1, w_2) + \Delta_F(w_2, w_3)$
   $$\Delta_F(w_1, w_2) + \Delta_F(w_2, w_3)$$
   $$= F(w_1) - F(w_2) - (w_1 - w_2)f(w_2)$$
   $$F(w_2) - F(w_3) - (w_2 - w_3)f(w_3)$$
   $$= \Delta_F(w_1, w_3) + (w_1 - w_2) \cdot (f(w_3) - f(w_2))$$

A Pythagorean Theorem [Br,Cs,A,HW]

$w^*$ is projection of $w$ onto convex set $W$ w.r.t. Bregman divergence $\Delta_F$:

$$w^* = \arg \min_{u \in W} \Delta_F(u, w)$$

Theorem:

$$\Delta_F(u, w) \geq \Delta_F(u, w^*) + \Delta_F(w^*, w)$$
Examples

Squared Euclidean Distance
\[ F(w) = \frac{||w||^2}{2} \]
\[ f(w) = w \]
\[ \Delta_F(\tilde{w}, w) = \frac{||\tilde{w}||^2}{2} - \frac{||w||^2}{2} - (\tilde{w} - w) \cdot w \]

(Unnormalized) Relative Entropy
\[ F(w) = \sum_i (w_i \ln w_i - w_i) \]
\[ f(w) = \ln w \]
\[ \Delta_F(\tilde{w}, w) = \sum_i \left( \frac{\tilde{w}_i}{w_i} \ln \frac{\tilde{w}_i}{w_i} + w_i - \tilde{w}_i \right) \]

61

Examples-2 [GLS,GL]

\( p \)-norm Algs (\( q \) is dual to \( p \): \( \frac{1}{p} + \frac{1}{q} = 1 \))
\[ F(w) = \frac{1}{2} ||w||^2_q \]
\[ f(w) = \nabla \frac{1}{2} ||w||^2_q \]
\[ \Delta_F(\tilde{w}, w) = \frac{1}{2} ||\tilde{w}||^2_q + \frac{1}{2} ||w||^2_q - \tilde{w} \cdot f(w) \]

When \( p = q = 2 \) this reduces to squared Euclidean distance (Widrow-Hoff).

62

Examples-3

Burg entropy
\[ F(w) = \sum_i - \ln w_i \]
\[ f(w) = -\frac{1}{w} \]
\[ \Delta_F(\tilde{w}, w) = \sum_i \left( - \ln \frac{\tilde{w}_i}{w_i} + \frac{\tilde{w}_i}{w_i} \right) - n \]

63

Examples-4: div. between density matrices

Umegaki Divergence
\[ F(W) = \text{tr}(W \ln W - W) \]
\[ f(W) = \ln W \]
\[ \Delta_F(\tilde{W}, W) = \text{tr} \left( \tilde{W}(\ln \tilde{W} - \ln W) + W - \tilde{W} \right) \]

LogDet Divergence
\[ F(W) = -\ln |W| \]
\[ f(W) = W^{-1} \]
\[ \Delta_F(\tilde{W}, W) = -\ln \left( \frac{|\tilde{W}|}{|W|} \right) + \text{tr}(\tilde{W}W^{-1}) - n \]

64
**General Motivation of Updates [KW]**

Trade-off between two terms:

\[
w_{t+1} = \arg\min_w (\Delta_F(w, w_t) + \eta_t L_t(w))
\]

\(\Delta_F(w, w_t)\) is “regularization term” and serves as measure of progress in the analysis.

When loss \(L\) is convex (in \(w\))

\[
\nabla w(\Delta_F(w, w_t) + \eta_t L_t(w)) = 0
\]

iff

\[
f(w) - f(w_t) + \eta_t \nabla L_t(w) = 0
\]

\[
\Rightarrow w_{t+1} = f^{-1}(f(w_t) - \eta_t \nabla L_t(w_t))
\]

**Quadratic Loss**

\[
L_t(w) = (y_t - w \cdot x_t)^2 / 2
\]

\(w_t = (-3/2, 1)\)

\(x_t = (1, -0.5)\)

\(y_t = 1\)

**Divergence: Euclidean Distance Squared**

\[
\Delta_F(w, w_t) = \|w - w_t\|^2 / 2
\]

\(w_t = (-3/2, 1)\)

\(x_t = (1, -0.5)\)

\(y_t = 1\)

**Loss + \(\eta\) Divergence**

\[
\Delta_F(w, w_t) = \|w - w_t\|^2 / 2
\]

\(w_t = (-3/2, 1)\)

\(x_t = (1, -0.5)\)

\(y_t = 1\)

\(\eta = 0.2\)
Divergence: 10-norm algorithm divergence

$$\Delta_F(w, w_t) = \nabla \frac{1}{2} ||w||_q^2$$

$$w_1 = (-3/2, 1)$$
$$x_t = (1, -0.5)$$
$$y_t = 1$$

How to prove relative loss bounds?

Loss:
$$L_t(w) = L((x_t, y_t), w) \text{ convex in } w$$

Divergence:
$$\Delta_F(u, w) = F(u) - F(w) - (u - w) \cdot f(w)$$

Update:
$$f(w_{t+1}) - f(w_t) = -\eta \nabla_w L_t(w_t)$$

convexity
$$L_t(u) \geq L_t(w_t) + (u - w_t) \cdot \nabla_w L_t(w_t)$$

update
$$= L_t(w_t) - \frac{1}{\eta} (u - w_t) \cdot (f(w_{t+1}) - f(w_t))$$

prop. 7 of $$\Delta_F$$

$$= L_t(w_t) + \frac{1}{\eta} (\Delta_F(u, w_{t+1}) - \Delta_F(u, w_t) - \Delta_F(w_t, w_{t+1}))$$

Loss + $$\eta$$ Divergence

$$\Delta_F(w, w_t) = ||w - w_t||_2^2/2$$

$$w_1 = (-3/2, 1)$$
$$x_t = (1, -0.5)$$
$$y_1 = 1$$
$$\eta = 0.2$$

First step: Teleskoping

Summing over $$t$$

$$\sum_t L_t(w_t) \leq \sum_t L_t(u) + \frac{1}{\eta} \sum_t \left( \Delta_F(u, w_t) - \Delta_F(u, w_{t+1}) + \Delta_F(w_t, w_{t+1}) \right)$$

$$\leq \sum_t L_t(u) + \frac{1}{\eta} \left( \Delta_F(u, w_1) - \Delta_F(u, w_{T+1}) \right) \geq 0$$

$$+ \frac{1}{\eta} \sum_t \Delta_F(w_t, w_{t+1})$$

$$\leq \sum_t L_t(u) + \frac{1}{\eta} \Delta_F(u, w_1) + \frac{1}{\eta} \sum_t \Delta_F(w_t, w_{t+1})$$

Any convex loss and any Bregman divergence!
**Second step: Relate $\Delta F(w_t, w_{t+1})$ to loss $L_t(w_t)$**

Loss & divergence are dependent
Get $\Delta F(w_t, w_{t+1}) \leq \text{const. } L_t(w_t)$
Then solve for $\sum_t L_t(w_t)$
Yield bounds of the form
$$\sum_t L_t(w_t) \leq a \sum_t L_t(u) + b \Delta F(u, w_1)$$
$a, b$ constants, $a > 1$
Regret bounds ($a = 1$):
time changing $\eta$, subtler analysis [AG]

**Bounds for Linear Regression with Square Loss**

Gradient Descent
$$\sum_t L_t(w_t) \leq (1 + c) \sum_t L_t(u) + \frac{1 + c}{c} X_2^2 U_2^2$$
$$||x_t||_2 \leq X_2, ||u||_2 \leq U_2, c > 0, \eta = f(c, X_2)$$

Scaled Exponentiated Gradient
$$\sum_t L_t(w_t) \leq (1 + c) \sum_t L_t(u) + \frac{1 + c}{c} \ln n X_\infty^2 U_1^2$$
$$||x_t||_\infty \leq X_\infty, ||u||_1 \leq U_1, c > 0, \eta = f(c, X_\infty)$$

$p$-norm Algorithm
$$\sum_t L_t(w_t) \leq (1 + c) \sum_t L_t(u) + \frac{1 + c}{c} (p - 1) X_p^2 U_q^2$$
$$||x_t||_p \leq X_p, ||u||_q \leq U_q, c > 0, \eta = f(c, X_\infty)$$

**Nonlinear Regression**

$y = h(w \cdot x)$

- Sigmoid function $h(z) = \frac{1}{1 + e^{-z}}$
- For a set of examples $(x_1, y_1), \ldots, (x_T, y_T)$
total loss $\sum_{t=1}^T h(w \cdot x) - y_t)^2/2$
can have exponentially many minima in weight space [Bu,AHW]

Want loss that is convex in $w$
Bregman Div. Lead to Good Loss Function

\[ \nabla_w \Delta_H(w \cdot x, h^{-1}(y)) = h(w \cdot x) - y \]

Then update has simple form:

\[ f(w_{t+1}) = f(w_t) - \eta_t (h(w_t \cdot x) - y_t)x_t \]

This can be exploited in proofs

But not absolutely necessary

One only needs convexity of \( L(h(w \cdot x), y) \) in \( w \)

\[ \int_{h^{-1}(y)} (h(z) - y) dz = H(w \cdot x) - H(h^{-1}(y)) - (w \cdot x - h^{-1}(y)) y \]

= \( \Delta_H(w \cdot x, h^{-1}(y)) \)

Idea behind the matching loss

If transfer function and loss match, then

\[ \nabla_w \Delta_H(w \cdot x, h^{-1}(y)) = h(w \cdot x) - y \]

Then update has simple form:

\[ f(w_{t+1}) = f(w_t) - \eta_t (h(w_t \cdot x) - y_t)x_t \]

This can be exploited in proofs

But not absolutely necessary

One only needs convexity of \( L(h(w \cdot x), y) \) in \( w \)

\[ \int_{h^{-1}(y)} (h(z) - y) dz = H(w \cdot x) - H(h^{-1}(y)) - (w \cdot x - h^{-1}(y)) y \]

= \( \Delta_H(w \cdot x, h^{-1}(y)) \)

Use \( \Delta_H(w \cdot x, h^{-1}(y)) \) as loss of \( w \) on \( (x, y) \)

Called matching loss for \( h \)

Matching loss is convex in \( w \) [AHW,HKW]

\[ \begin{array}{|c|c|c|}
\hline
\text{transfer f.} & H(z) & \text{match. loss} \\
\hline
h(z) & & \Delta_H(w \cdot x, h^{-1}(y)) \\
\hline
z & \frac{1}{2} z^2 & \frac{1}{2}(w \cdot x - y)^2 \\
& & \text{square loss} \\
\hline
\frac{e^z}{1+e^z} & \ln(1 + e^z) & \ln(1 + ew \cdot x) - yw \cdot x \\
& +y \ln y + (1 - y) \ln(1 - y) & \text{logistic loss} \\
\hline
\text{sign(z)} & |z| & \max\{0, -yw \cdot x\} \\
& & \text{hinge loss} \\
\hline
\end{array} \]

Sigmoid in the Limit

For transfer function \( h(z) = \text{sign}(z) \)

\[ H(z) = |z| \]

Matching loss is hinge loss [GW]

\[ H_L(w \cdot x, h^{-1}(y)) = \max\{0, -w \cdot x\} \]

Convex in \( w \) but not differentiable
Motivation of linear threshold algs

Gradient descent
with
Hinge Loss

Expon. gradient
with
Hinge Loss

Known linear threshold algorithms for \( \pm 1 \)-classification case are gradient-based algorithms with hinge loss

Perceptron

\[
\begin{align*}
\mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \left( \frac{1}{2} ||\mathbf{w} - \mathbf{w}_t||^2 / 2 + \eta \cdot HL(\mathbf{w} \cdot \mathbf{x}_t, g^{-1}(y_t)) \right) \\
&= \mathbf{w}_t - \eta \left( \sign(\mathbf{w}_{t+1} \cdot \mathbf{x}_t) - y_t \right) \mathbf{x}_t \\
&\approx \mathbf{w}_t - \eta \left( \sign(\mathbf{w}_t \cdot \mathbf{x}_t) - y_t \right) \mathbf{x}_t
\end{align*}
\]

Normalized Winnow

\[
\begin{align*}
\mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \left( \sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}} + \eta \cdot HL(\mathbf{w} \cdot \mathbf{x}_t, g^{-1}(y_t)) \right) \\
&= \mathbf{w}_{t,i} e^{-\eta \left( \sign(\mathbf{w} \cdot \mathbf{x}_t) - y_t \right) \mathbf{x}_t} \Bigg/ \text{normalization} \\
&\approx \mathbf{w}_{t,i} e^{-\eta \left( \sign(\mathbf{w}_t \cdot \mathbf{x}_t) - y_t \right) \mathbf{x}_t} \Bigg/ \text{normalization}
\end{align*}
\]

Trade-off between two divergences \([KW]\)

\[
\begin{align*}
\mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \left( \Delta F(\mathbf{w}, \mathbf{w}_t) + \eta \cdot \Delta H(\mathbf{w} \cdot \mathbf{x}_t, h^{-1}(y_t)) \right) \\
&= \text{parameter divergence} + \text{matching loss divergence}
\end{align*}
\]

Both divergences are convex in \( \mathbf{w} \)

\[
\begin{align*}
\mathbf{w}_{t+1} &= f^{-1} \left( f(\mathbf{w}_t) - \eta \left( h(\mathbf{w}_t \cdot \mathbf{x}_t) - y_t \right) \mathbf{x}_t \right)
\end{align*}
\]

Generalization of the “delta”-rule
Duality

Special case:
\[
\min_w \Delta_F(w, w_t) + \Delta_H(x_t \cdot w, h^{-1}(y_t)) \\
= -\min_{\alpha} \Delta_F(\alpha + y_t, h(0)) + \Delta_F(f(w_t) - \alpha x_t, f(0)) + \text{const.}
\]

General:
\[
\min_w \Delta_F(w + \mu, f^{-1}(\phi)) + \Delta_H(Xw + \nu, h^{-1}(y)) \\
= -\min_{\alpha} \Delta_F(\alpha + y, h(\nu)) + \Delta_F(\phi - X^\top \alpha, f(\nu)) + \text{const.}
\]

where \( F \) and \( \mathcal{F} \) are convex conjugate functions:
\[
\mathcal{F}(x) = \sup_y x \cdot y - F(y) = x \cdot (\nabla F)^{-1}(x) - F((\nabla F)^{-1}(x))
\]

Relation to Boosting

The AdaBoost update of the probability vector \( w_t \):
\[
w_{i+1} = w_i \exp(-\alpha_i y_i h_t(x_i))
\]

Is a projection w.r.t. divergence \[\text{CKW,La,KW,CSS}\]
\[
\Delta_F(w, w_t) = \sum_i w_i \ln \frac{w_i}{w_{t,i}}
\]

Such that the weighted training error of \( h_t \) w.r.t. \( w^{(t+1)} \) is \( \frac{1}{2} \) (“diversification” of Boosting mentioned in Ron Meir’s talk)

Projections onto Hyperplanes

\[
w_{t+1} = \arg\min_w (\Delta_F(w, w_t) + \eta(w \cdot x_t - y_t)^2)
\]

When \( \eta \) is large then \( w_{t+1} \) is projection of \( w_t \) onto plane \( w \cdot x_t = y_t \)

\[
\min \{ w : w \cdot x_t = y_t \}
\]

where \( \mathcal{F} \) and \( \mathcal{F}_t \) are convex conjugate functions:
\[
\mathcal{F}(x) = \sup_y x \cdot y - F(y) = x \cdot (\nabla F)^{-1}(x) - F((\nabla F)^{-1}(x))
\]

Relation to Boosting

The AdaBoost update of the probability vector \( w_t \):
\[
w_{i+1} = w_i \exp(-\alpha_i y_i h_t(x_i))
\]

Is a projection w.r.t. divergence \[\text{CKW,La,KW,CSS}\]
\[
\Delta_F(w, w_t) = \sum_i w_i \ln \frac{w_i}{w_{t,i}}
\]

Such that the weighted training error of \( h_t \) w.r.t. \( w^{(t+1)} \) is \( \frac{1}{2} \) (“diversification” of Boosting mentioned in Ron Meir’s talk)
Bregman divergences and exponential families?

- Exponential family of distributions
- Inherent duality

\[ w_{t+1} = f^{-1}(f(w_t) - \eta \nabla L_t(w_t)) \]

primal param.          dual param.

| \( w_t \) | \( \frac{f}{f} \) | \( f(w_t) \) |
| \( w_{t+1} \) | \( \frac{f^{-1}}{-\eta \nabla L_t(w_t)} \) |

Exponential Family of Distributions

- Parametric density functions
  \[ P_G(x|\theta) = e^{\theta \cdot x - G(\theta)} P_0(x) \]
- \( \theta \) and \( x \) vectors in \( \mathbb{R}^d \)
- Cumulant function \( G(\theta) \) assures normalization
  \[ G(\theta) = \ln \int e^{\theta \cdot x} P_0(x) \, dx \]
- \( G(\theta) \) is convex function on convex set \( \Theta \subseteq \mathbb{R}^d \)
- \( G \) characterizes members of the family
- \( \theta \) is natural parameter

Expectation parameter

\[ \mu = \int x P_G(x|\theta) \, dx = E_{\theta}(x) = g(\theta) \]

where \( g(\theta) = \nabla_{\theta} G(\theta) \)
- Second convex function \( F(\mu) \) on space \( g(\Theta) \)
  \[ F(\mu) = \theta \cdot \mu - G(\theta) \]
- \( G(\theta) \) and \( F(\mu) \) are convex conjugate functions
- Let \( f(\mu) = \nabla_{\mu} F(\mu) \)
- \( f(\mu) = g^{-1}(\mu) \)
**Primal & Dual Parameters**

natural expectation parameter parameter

\[ \theta \xrightarrow{g} \mu \]

\[ G(\theta) \xrightarrow{f} F(\mu) \]

- \( \theta \) and \( \mu \) are dual parameters
- Parameter transformations \( g(\theta) = \mu \) and \( f(\mu) = \theta \)

**Gaussian (unit variance)**

\[ P(x|\theta) \sim e^{-\frac{1}{2}(\theta - x)^2} \]

\[ = e^{\theta x - \frac{1}{2} \theta^2} e^{\frac{1}{2} x^2} \]

Cumulant function: \( G(\theta) = \frac{1}{2} \theta^2 \)

Parameter transformations:
\[ g(\theta) = \theta = \mu \quad \text{and} \quad f(\mu) = \mu = \theta \]

Dual convex function: \( F(\mu) = \theta \cdot \mu - G(\theta) \)

\[ = \frac{1}{2} \mu^2 \]

Square loss: \( L_t(\theta) = \frac{1}{2}(\theta_t - x_t)^2 \)

**Bernoulli**

Examples \( x_t \) are coin flips in \( \{0, 1\} \)

\[ P(x|\mu) = \mu^x (1 - \mu)^{1-x} \]

\( \mu \) is the probability (expectation) of 1

Natural parameter: \( \theta = \ln \frac{\mu}{1-\mu} \)

\[ P(x|\theta) = \exp \left( \theta x - \ln(1 + e^{\theta}) \right) \]

Cumulant function: \( G(\theta) = \ln(1 + e^{\theta}) \)

Parameter transformations:
\[ \mu = g(\theta) = \frac{e^{\theta}}{1 + e^{\theta}} \quad \text{and} \quad \theta = f(\mu) = \ln \frac{\mu}{1 - \mu} \]

Dual function: \( F(\mu) = \mu \ln \mu + (1 - \mu) \ln(1 - \mu) \)

Log loss: \( L_t(\theta) = -x_t \theta + \ln(1 + e^{\theta}) \)

\[ = -x_t \ln \mu - (1 - x_t) \ln(1 - \mu) \]

**Poisson**

Examples \( x_t \) are natural numbers in \( \{0, 1, \ldots\} \)

\[ P(x|\mu) = \frac{e^{-\mu} \mu^x}{x!} \]

\( \mu \) is expectation of \( x \)

Natural parameter: \( \theta = \ln \mu \)

\[ P(x|\theta) = \exp \left( \theta x - e^{\theta} \right) \frac{1}{x!} \]

Cumulant function: \( G(\theta) = e^{\theta} \)

Parameter transformations:
\[ \mu = g(\theta) = e^{\theta} \quad \text{and} \quad \theta = f(\mu) = \ln \mu \]

Dual function: \( F(\mu) = \mu \ln \mu - \mu \)

Loss: \( L_t(\theta) = -x_t \theta + e^{\theta} + \ln x_t! \)

\[ = -x_t \ln \mu + \mu + \ln x_t! \]
Bregman Div. as Rel. Ent. between Distributions

Let \( P(x|\theta) \) and \( P(x|\tilde{\theta}) \) denote two distributions with cumulant function \( G \)

\[
\Delta_G(\theta, \tilde{\theta}) = \int_x P_G(x|\theta) \ln \frac{P_G(x|\theta)}{P_G(x|\tilde{\theta})} dx
\]

\[
= \int_x P_G(x|\theta)(\theta \cdot x - G(\theta) - \tilde{\theta} \cdot x + G(\tilde{\theta})) dx
\]

\[
= G(\theta) - G(\tilde{\theta}) - (\theta - \tilde{\theta}) \cdot (\int_x P_G(x|\theta) x dx)
\]

\[
= G(\theta) - G(\tilde{\theta}) - (\theta - \tilde{\theta}) \cdot \mu
\]

\[
F(\mu) = \frac{G(\theta) - G(\tilde{\theta}) - (\theta - \tilde{\theta}) \cdot \mu}{\Delta G(\theta, \tilde{\theta})}
\]

\[
\Delta F(\mu, \tilde{\mu}) = \Delta F(\tilde{\mu}, \mu)
\]

Dual divergence for Bernoulli

\[
G(\theta) = \ln(1 + e^\theta) \quad F(\mu) = \mu \ln \mu + (1 - \mu) \ln(1 - \mu)
\]

\[
g(\theta) = \frac{e^\theta}{1 + e^\theta} = \mu \quad f(\mu) = \ln \frac{\mu}{1 - \mu} = \theta
\]

\[
\Delta_G(\tilde{\theta}, \theta) = \ln(1 + e^\tilde{\theta}) - \ln(1 + e^\theta) - (\tilde{\theta} - \theta) \frac{e^\theta}{1 + e^\theta}
\]

\[
\Delta F(\mu, \tilde{\mu}) = \mu \ln \frac{\mu}{\tilde{\mu}} + (1 - \mu) \ln \frac{1 - \mu}{1 - \tilde{\mu}}
\]

Binary relative entropy

Sum of binary relative entropies is parameter divergence for BEG
Dual divergence for Poisson

\[ G(\theta) = e^\theta \quad F(\mu) = \mu \ln \mu - \mu \]

\[ g(\theta) = e^\theta = \mu \quad f(\mu) = \ln \mu = \theta \]

\[ \Delta_G(\tilde{\theta}, \theta) = e^{\tilde{\theta}} - e^\theta - (\tilde{\theta} - \theta)e^\theta \]

\[ \Delta_F(\mu, \tilde{\mu}) = \mu \ln \frac{\mu}{\tilde{\mu}} + \tilde{\mu} - \mu \]

Unnormalized relative entropy

Sum of unnormalized relative entropies is parameter for UEG (e.g. Winnow)

Dual matching loss for sigmoid transfer func.

\[ H(z) = \ln(1 + e^z) \quad K(r) = r \ln r + (1 - r) \ln(1 - r) \]

\[ h(z) = \frac{e^z}{1 + e^z} = r \quad k(r) = \ln \frac{r}{1 - r} = z \]

\[ K \text{ dual to } H \text{ and } k = h^{-1} \]

\[ \Delta_H(w \cdot x, h^{-1}(y)) = \ln(1 + e^{w \cdot x}) - yw \cdot x + y \ln y + (1 - y) \ln(1 - y) \]

By duality logistic loss is same as entropic loss

\[ \Delta_K(y, h(w \cdot x)) = y \ln \frac{y}{h(w \cdot x)} + (1 - y) \ln \frac{1 - y}{1 - h(w \cdot x)} \]

Matching loss for logistic transfer function

Example: Gaussian density estimation

\[ L_t(\theta) = - \ln P(x_t | \theta) = \frac{1}{2}(x_t - \theta)^2 \]

Off-line versus on-line

- Loss on example \( x_t \)

Derivation of Updates

- Want to bound

\[ \sum_{t=1}^{T} L_t(\theta_t) - \inf_{\theta} L_{1..T}(\theta) \]

- Off-line algorithm has all \( T \) examples \( \{x_1, x_2, \ldots, x_T\} \)

- Setup for choosing best parameter setting

\[ \theta_B = \arg\min_{\theta} (n_B^{-1} \Delta_G(\theta, \theta_1) + L_{1..T}(\theta)) \]

Here \( n_B^{-1} > 0 \) is a tradeoff parameter
**On-line Algorithm [AW]**

- In trial $t$, the first $t$ examples
  \[ \{x_1, x_2, \ldots, x_t\} \]
  have been presented
- Motivation for on-line parameter update:
  do as well as best off-line algorithm up to trial $t$
- At end of trial $t$ algorithm minimizes
  \[ \theta_{t+1} = \arg \min_{\theta} (\eta_t^{-1} \Delta_G(\theta, \theta_1) + L_{1:t}(\theta)) \]
  divergence to initial so far
  Tradeoff parameter $\eta_t^{-1} \geq 0$

**Alternate Motivation of Same On-Line Update**

\[ \theta_{t+1} = \arg \min_{\theta} (\eta_t^{-1} \Delta_G(\theta, \theta_t) + L_t(\theta)) \]

where \[ \eta_t = \frac{1}{\eta_t^{-1} + t - 1} \]

**Parameter Updates**

Off-line:
\[ \mu_B = \eta_B^{-1} \mu_1 + \frac{\sum_{t=1}^T x_t}{\eta_B^{-1} + T} \]

On-Line in trial $t$:
\[ \mu_{t+1} = \frac{\eta_t^{-1} \mu_1 + \sum_{q=1}^t x_q}{\eta_t^{-1} + t} = \mu_t - \eta_{t+1}(\mu_t - x_t) \]

\[ \theta_{t+1} = g^{-1}(g(\theta_t) - \eta_{t+1}(\mu_t - x_t)) \]

**Shrinkage Towards Initial**

\[ \mu_B = \bar{x}_T - \eta_B^{-1} (\eta_B^{-1} + T)^{-1} (\bar{x}_T - \mu_1) \]

where \[ \bar{x}_T = \frac{\sum_{t=1}^T x_t}{T} \]

Shrinkage factor \[ \eta_B^{-1}(\eta_B^{-1} + T)^{-1} \]
Key Lemma \([\text{AW}]\)

For any example \(x_t\) and any \(\theta \in \Theta\)

\[
L_t(\theta_t) - L_t(\theta) = \eta_t^{-1} \Delta_G(\theta, \theta_t) - \eta_{t+1}^{-1} \Delta_G(\theta, \theta_{t+1})
\]

divergence to initial par. divergence to last par.

\[
+ \eta_t^{-1} \Delta_G(\theta_t, \theta_{t+1})
\]
cost of update

Main Theorem

For any sequence of examples and any \(\theta \in \Theta\)

\[
\sum_{t=1}^{T} L_t(\theta_t) - \inf_{\theta} L_1..T(\theta) = \sum_{t=1}^{T} \eta_t \frac{x_t^2}{2} - \sum_{t=1}^{T-1} \eta_{t+1} \frac{\mu_{t+1}^2}{2} \quad [\text{AW}]
\]

\[
\leq \frac{X^2}{2} \ln(1 + \frac{T}{\eta_t^{-1} - 1})
\]

\[
\text{Bernoulli} \leq \frac{1}{2} \ln(T+1) + \frac{\ln \pi}{2} \quad [\text{Fr, XB, AW}]
\]

\[
\text{lin. regr.} \leq \frac{1}{2} Y^2 n \ln \left(1 + \frac{T X^2}{a}\right) \quad [V, Fo, AW]
\]

\[
X^2 = \max_{t=1}^{T} x_t^2, \quad Y = \max_{t=1}^{T} \eta_t, \quad w_t = \left( a I + \sum_{q=1}^{t} x_q x_q' \right)^{-1} \sum_{q=1}^{t-1} x_q y_q
\]
Why Bregman divergences?

- No need to check whether there is an underlying exponential family
- More general than exponential families
- As parameter divergence and matching loss
- Used in motivation and analysis of updates
- When $\eta \to \infty$, updates morph into Bregman projection
- Generalized Pythagorean Theorem for Bregman projections

General setup of on-line learning

- We hide some information from the learner
- The relative loss bound quantifies the price for hiding the information
- So far the future examples are hidden
  Off-line algorithm knows all examples
  On-line algorithm knows past examples

Minimax Algorithm for $T$ Trials

Learner against adversary

\[
\inf_{\mu_1} \sup_{x_1} \inf_{\mu_2} \sup_{x_2} \inf_{\mu_3} \sup_{x_3} \ldots \inf_{\mu_T} \sup_{x_T}
\]

\[
\sum_{t=1}^{T} \frac{1}{2} (\mu_t - x_t)^2 - \inf_{\mu} \sum_{t=1}^{T} \frac{1}{2} (\mu - x_t)^2
\]

Total loss of on-line algorithm

Total loss of off-line algorithm

Instances must be bounded: $||x_t||_2 \leq X$

Minimax algorithm usually intractable

Gaussian

Forward Alg.

\[
\mu_t = \frac{\sum_{s=1}^{t-1} x_s}{t}
\]

Bound

\[
\frac{1}{2}X^2(1 + \ln T)
\]

Minimax Alg.

\[
\mu_t = \frac{\sum_{s=1}^{t-1} x_s}{t + \ln T - \ln(t + \ln T)}
\]

Bound

\[
\frac{1}{2}X^2(\ln T - \ln \ln T) + o(1)
\]

Minimax alg. needs to know $T$
Last-step Minimax

Assumes that current trial is last trial \[ \text{[Fo,TW]} \]

\[
\mu_t = \arg\inf_{\mu} \sup_{x_t} \sum_{q=1}^{t} L_q(\mu_q) - \inf_{\mu} L_{1..t}(\mu)
\]

\[
= \arg\inf_{\mu} \sup_{x_t} L_t(\mu_t) - \inf_{\mu} L_{1..t}(\mu)
\]

For Gaussian and linear regression

Last-step Minimax is same as Forward Alg.

### Last-step Minimax: Bernoulli

Forward alg:

\[
\mu_t = \frac{s + \frac{1}{2}}{t - 1 + 1}, \text{ where } s = \sum_{q=1}^{t-1} x_q
\]

Last-step:

\[
\mu_t = \frac{(s + 1)^{s + 1}(t - s - 1)^{t-s-1}}{s^s (t - s)^{t-s} + (s + 1)^{s+1}(t - s - 1)^{t-s-1}}
\]

Worst-case regret bounds: \( \ln(T + 1) + c \)

Forward: \( c = \frac{\ln \pi}{2} \)

Last step: \( c = \frac{1}{2} \)

### Synopsis of methods

- Game theoretic
  - Slightly better bounds
  - Harder to find
- Bregman divergences
  - Closer to Bayes and standard convex optimization

### Content of this tutorial

- P I: Introduction to Online Learning
  - The Learning setting
  - Predicting as good as the best expert
  - Predicting as good as the best linear combination of experts
- P II: Bregman divergences and Loss bounds
  - Introduction to Bregman divergences
  - Relative loss bounds for the linear case
  - Nonlinear case & matching losses
  - Duality and relation to exponential families
  - On-line algorithms motivated by game theory
- P III: on-line to batch conversion, applications
  - Simple conversions
  - Caching and the disk spin down problem
  - Other applications and conclusion

**Goal:** How can we prove relative loss bounds?
Simple conversions

Worst case loss bounds for on-line algs are converted to algorithms with good performance bounds in the i.i.d. case

- Expected loss bounds [HW,CB+,KW]
- Tail bounds [CCG]

Expected loss bounds [HW]

Loss function $L : \mathbb{R}^2 \rightarrow \mathbb{R}$

$S = (x_1, y_1), \ldots, (x_T, y_T) \sim D^T$

Instantaneous loss of hypothesis $h$ w.r.t. distribution $D$

$$\text{InstLoss}(h, D) = E_{e \sim D} L(h, e)$$

$$E_{S \sim D^T} (\text{TotLoss}(A, S))$$

$$= E_{(e_1, \ldots, e_T) \sim D^T} \left( \sum_{t=1}^{T} L(A((e_1, \ldots, e_{t-1}), e_t) \right)$$

$$= \sum_{t=1}^{T} E_{(e_1, \ldots, e_{t-1}) \sim D^{t-1}} \left( E_{e \sim D} \left( L(A((e_1, \ldots, e_{t-1}), e) \right) \right)$$

$$= \sum_{t=1}^{T} E_{(e_1, \ldots, e_{t-1}) \sim D^{t-1}} \left( \text{InstLoss}(A(e_1, \ldots, e_{t-1}), D) \right)$$

So expected total loss is total instantaneous loss

| Run | X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|X|T+1 hyp's |
| A   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

- Choose $h_i$ uniformly at random from the $T+1$ hypotheses
- On new instance $x$ predict with $h_i(x)$
- Instantaneous loss of this algorithm is expected total loss of original algorithm over $T+1$
- Applied to the Perceptron Algorithm [FS]
Convex Loss $L : \mathbb{R}^2 \rightarrow [0, L_{\text{max}}]$ 

If total worst case loss is $M$, then with probability $1 - \delta$

$$err_D(\overline{h}) \leq \frac{M}{T} + L_{\text{max}} \sqrt{\frac{2}{T} \log \frac{1}{\delta}}$$

- Application: Adaptive Channel Equalization

- Online Linear Regression Problem:
  ⇒ Find $w$ such that $(y - w \cdot x)^2$ is minimized

  - Common approach:
    $$w_{t+1} = w_t - \eta (y - w_t \cdot x_t)x_t$$

  - But: Many coefficients are zero, or close to zero [MSWJ]
    ⇒ Use Unnormalized Exponentiated Gradient update or the approximate version
    $$w_{t+1} = w_t (1 - \eta (y - w_t \cdot x_t)x_t)$$

- Application: Caching [GBW]

  - Whenever small, fast memory and larger, slower secondary memory
  - Keep objects in fast memory which are 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

Best Policy Varies with time
Want “Adaptive” Policy

- Good compared to off-line comparator
  - **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

---

**Goal for On-line policies**

- Beat BestFixed
- Get close to BestRefetching
- Reduce I/O’s and end-user latency

---

**Dynamic programming in time $O(RN^2T)$**
### Score Card

**Key Idea: Virtual Caches**

- Simulates a cache for each baseline policy
- Per object keep only (ID, size and calculated priority)
- Maintenance cost negligible
- Observe current miss rates of all 12
- Virtual Caches reside in the total cache space:
  \[
  \text{Size(real cache)} = \text{Size(full cache)} - \sum_{i=1}^{12} \text{Size}(V_{C_i}).
  \]

### Virtual Caches

- Size(full cache)

  **Fixed Policy, e.g. LRU**
  (with object data)

  **Real Cache**
  (with object data)

  Size(real cache) = \text{Sum}(\text{Size}(V_{C_i}))

### Window Algorithm

- **Real** cache governed by currently **best** policy
- Best means lowest number of hits in window of \( W \) (say 300) requests
- Works reasonably well - but
  - Hard to tune the window size
  - \( O(NW) \) Additional space required for \( N \) policies.
Better Master Policy

- Use Expert Framework from On-line learning
- Maintain one weight $w_i$ for each base policy / expert
- $w_i$ is estimate of current relative performance of policy $i$
- Weights updated after each request:
  - Loss update punishes policies quickly that score misses
  - Share update [LW94, HW98, BW01]
    Keeps weights of poor policies from becoming too small
    Helps recovery

\[ w_{t,i}' = \frac{w_{t,i} \beta \text{miss}_{t,i}}{\text{normaliz}}, \quad \beta \in (0, 1) \]

Share Update:

\[ w_{t+1} = (1 - \alpha) w_t' + \alpha r_{t-1}, \]

where \( r_{t-1} = \sum_{q=1}^{t-1} w_q' / (t - 1) \)

- Prevents weights that did well in past from becoming too small
  Helps when these weights need to recover

Weights of baseline policies under FSUP

Digression

More on On-line Learning and Share Updates
On-line Learning

experts

<table>
<thead>
<tr>
<th>day</th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_n$</th>
<th>prediction</th>
<th>true label</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>day 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>day 2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>day 3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>day $t$</td>
<td>$x_{t,1}$</td>
<td>$x_{t,2}$</td>
<td>$x_{t,3}$</td>
<td>$x_{t,n}$</td>
<td>$\hat{y}_t$</td>
<td>$y_t$</td>
<td>$(y_t - \hat{y}_t)^2$</td>
</tr>
</tbody>
</table>

- Choose comparison class of predictors (experts)
- Master Algorithm combines predictions of experts
- $x_t$ vector of expert’s predictions

Protocol of Master Algorithm

Loop for each trial $t = 1, \ldots, T$
- Get next instance $x_t$
- Make prediction $\hat{y}_t$
- Get label $y_t$ (“true outcome”)
- Incur loss $L(\hat{y}_t, y_t)$

- No statistical assumptions on the data

Goal
- Do well compared to the best off-line comparator / best expert

What kind of performance can we expect?

- $L_{1.T,A}$ be the total loss of algorithm $A$
- $L_{1.T,i}$ be the total loss of $i$-th expert $E_i$

- Form of bounds

\[ \forall S: \quad L_{1.T,A} \leq \min_i \left( L_{1.T,i} + c \log n \right) \]

where $c$ is constant

- Bounds the loss of the algorithm relative to the loss of best expert

Algorithm that Achieves Bound

- Master algorithm predicts with weighted average

\[ \hat{y}_t = w_t \cdot x_t \]

- The weights are updated according to the Loss Update

\[ w_{t+1,i} := \frac{w_{t,i} e^{-\eta L_{t,i}}}{\text{normaliz.}}, \quad e^{-\eta} = \beta \]

where $L_{t,i}$ is loss of expert $i$ in trial $t$

→ Weighted Majority Algorithm \[ [LW89] \]

→ Generalized by Vovk \[ [Vovk90] \]
What if Comparator Changes with Time?

- Off-line algorithm partitions sequence into sections and chooses best expert in each section.
- Goal: Do well compared to the best off-line partition.
- Problem: Loss Update learns too well and does not recover fast enough.

Mixing Update

- Predict $\hat{y}_t = w_t \cdot x_t$.
- Loss Update $w_{t+1}^i = \frac{w_t^i e^{-\eta L_t^i}}{\text{normaliz.}}$.
- Mixing Update $w_{t+1} = \sum_{q=0}^{t} \beta_{t+1,q} w_q^t$, where $\sum_{q=0}^{t} \beta_{t+1} = 1$.
- Mixing schemes:
  - FS to Start Vector
  - FS to Uniform Past
  - FS to Decaying Past

Total Loss Plots

- $T = 1400$ trials, $n = 20000$ experts.
- $k = 6$ shifts (every 200 trials).

Weights of Fixed Share to Start Vector Alg.
Weights of Fixed Share to Decaying Past Alg.

- Improved recovery when expert used before

Fixed Share to Decaying Past - Log Weights

- Past good experts remain at higher level

More Experts Remembered

- $T = 6000$ trials, $n = 20000$ experts
- $k = 29$ shifts (every 200 trials)

Fixed Share to Decaying Past - Log Weights

- Past good experts are cached
Fixed Share to Start Vector - Log Weights

- No memory

Relative Loss Bounds

- Always have the form
  \[ L_{1,T,A} \leq \min_P (L_{1,T,P} + O(\# \text{ of bits for } P)) \]

→ Boundaries are encoded twice
→ Off-line problem NP-complete

Fixed Share to Decaying Past - Log Weights

- Larger alpha gives better long-term memory
Memory from many short sections accumulates

- Fixed Share to Decaying Past - Log Weights

![Graph showing log weights over trials for different experts and max others.]

Bigger memory

- Cycling thru 10 different short sections

![Graph showing log weights over trials for different experts and max others.]

Back to Caching

- Share-update crucial
- Fixed Share to Uniform Past cheap one of the best
- Bounds do not apply but we are using recovery properties
  - parameter settings $\langle \alpha, \beta \rangle$ or $\eta$ not crucial
  - fix at

$$\beta = 1/e \quad \alpha = 0.005$$

Master Policy Protocol

- Process request on virtual caches
- Apply Loss and Share Updates
- Process request on real cache
  - based on combined weightings of all caches
- Refetch objects into real cache (if desired)
Virtual Cache Rankings

- Priorities induce ranks over virtually cached objects:

<table>
<thead>
<tr>
<th>object</th>
<th>$o_{12}$</th>
<th>$o_7$</th>
<th>$o_2$</th>
<th>$o_{22}$</th>
<th>$o_3$</th>
<th>$o_6$</th>
<th>$o_2$</th>
<th>$o_{15}$</th>
<th>$o_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>31.2</td>
<td>30.2</td>
<td>24.1</td>
<td>17.1</td>
<td>9.3</td>
<td>8</td>
<td>4.1</td>
<td>2.5</td>
<td>1.2</td>
</tr>
<tr>
<td>rank</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
- $o_9$ first discarded
- $o_{12}$ last

Master Rank

- Master priority $P$ constructed from weights and ranks of virtual caches

$$P_o = \begin{cases} \sum_{n:o \in VC_n} w_n r_{n,o} & \text{if } \exists n: o \in VC_n \\ 0 & \text{if } \forall n: o \notin VC_n \end{cases}$$
- $R$ is corresponding master rank

Ideal Cache

- Highest ranked objects fill the ideal cache to capacity
- $\text{IdealCache} =$

Managing Real Cache: Instantaneous Rollover

- Keep $\text{RealCache} = \text{IdealCache}$
  - i.e. Refetch all $o \in \text{IdealCache} - \text{RealCache}$
- Too much refetching
**Demand Rollover**

- Lowest $R$-ranked objects are discarded to make room for a new request
- No refetching

**Compromise: Background Rollover**

- Refetch objects $o \in \text{IdealCache} - \text{RealCache}$ when system is idle
- Model idleness as Poisson process
  - Draw $d \sim \text{Pois}(\lambda)$
  - Refetch (at most) $d$ objects $o \in \text{IdealCache} - \text{RealCache}$

**Smart Refetching**

- Most hits in real cache have high $R$-rank
- Refetch only top 40-60% of $R$-ranked objects

**Experimental Results: Filesystem Data**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Work-Week (WWk)</th>
<th>User-Month (UMo)</th>
<th>Server-Month-LRU (SMoLRU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Requests</td>
<td>138k</td>
<td>382k</td>
<td>48k</td>
</tr>
<tr>
<td>Cache size</td>
<td>900KB</td>
<td>2MB</td>
<td>4MB</td>
</tr>
<tr>
<td>%Skipped</td>
<td>6.5%</td>
<td>12.8%</td>
<td>15.7%</td>
</tr>
<tr>
<td>%Compute</td>
<td>0.020</td>
<td>0.015</td>
<td>0.152</td>
</tr>
<tr>
<td>LRU Miss Rate</td>
<td>0.166</td>
<td>0.076</td>
<td>0.870</td>
</tr>
<tr>
<td>RealFixed Pol / MR</td>
<td>SIZE 0.055</td>
<td>GDS 0.075</td>
<td>GDSF 0.399</td>
</tr>
<tr>
<td>%&lt;LRU</td>
<td>36.8%</td>
<td>54.7%</td>
<td>54.2%</td>
</tr>
</tbody>
</table>

CMU DFStrace
 Demand Rollover “Tracks” best policy

Miss-rates under FSUP with Master

<table>
<thead>
<tr>
<th>lru</th>
<th>fifo</th>
<th>mem</th>
<th>life</th>
<th>size</th>
<th>lfu</th>
<th>mru</th>
<th>rand</th>
<th>gdfs</th>
<th>ftada</th>
<th>gel</th>
<th>squ</th>
</tr>
</thead>
</table>

Requests Over Time

<table>
<thead>
<tr>
<th>205000</th>
<th>210000</th>
<th>215000</th>
<th>220000</th>
<th>225000</th>
<th>230000</th>
<th>235000</th>
</tr>
</thead>
</table>

WWk

WWk Master and Comparator Missrates

- 0.5% = LRU missrate
- 2.0% = Obligatory missrate

BestRet: BestRetrieving(R)
- Rank Ideal
- Rank 60% Ideal
- Rank 40% Ideal
- BestFixed = SIZE
- AIVC

Retrakes as % of Total Requests

UMo

UMo Master and Comparator Missrates

- 16.7% = LRU missrate
- 1.9% = Obligatory missrate

BestRet: BestRetrieving(R)
- Rank Ideal
- Rank 60% Ideal
- Rank 40% Ideal
- BestFixed = CDS
- AIVC

Retrakes as % of Total Requests

SMoLRU

SMoLRU Master and Comparator Missrates

- 59.8% = LRU missrate
- 15.3% = Obligatory missrate

BestRet: BestRetrieving(R)
- Rank Ideal
- Rank 60% Ideal
- Rank 40% Ideal
- BestFixed = SIZE
- AIVC

Retrakes as % of Total Requests
**Summary**

- **Demand Rollover** is already as good or better than BestFixed
- Small amounts of refetching always beats Best Fixed
  - 15-22% fewer misses than BestFixed
  - 45-70% fewer misses than LRU
- Can be as good as BestRefetching
  - always less I/O’s than LRU
  - can result in less I/O than BestFixed

**Conclusion**

- Operating Systems have many parameter tweaking problems suitable for on-line learning
- Previous work using same updates:
  - Tuning time-out for spinning down disk of a PC [HLSS00]
  - Load balancing between processors [BB97]
  - Tracking with GPS

**Too expensive?**

- Not for web caching and filesystem’s caching
- Not clear for paging
- Implement in Linux kernel

**Two approaches**

- Use existing caching strategies as experts
- Use set of fine-grained experts from which all existing caching policies are built
- Machine Learners will get interested if there are realistic benchmark data sets
Application: Disk Spin Down [HLSS]

Problem of adapt. spinning down hard disks in mobile computers

Common approach

- Fixed time-out (e.g. 2 min)
- Does not exploit changing usage patterns

Idea

- Use about 20 experts with different time-outs
- Apply shifting expert algorithm with mixing to decaying past
- Efficient but proofs don’t apply because of unusual loss function

Which loss?

Costs for spinning up/down, running machine in idle mode, ...
L("idle-time", "time-out") \sim \text{total energy consumed}
**Does it work?**

Comparators:
- Best fixed idle time chosen in hindsight
- Optimal algorithm: Spin down if cost of next idle period > spin down cost

Performance
- Better than best fixed
- Close to optimal
- Parameters easy to tune and algorithm very stable over a large variety of data
- Better than other algorithms that provable have good competitive ratios

---

**Other Applications**

- Calendar managing
  Many features (sleeping experts) [Bl,FSSW]
- Text categorization
  One attribute per word in text [LSCP]
- Spelling correction [Ro]
- Portfolio prediction [Co,CO,HSSW,BK]
- Boosting [Sc,Fr,SS]
- Load Balancing based on shifting expert algorithms [BB]