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`

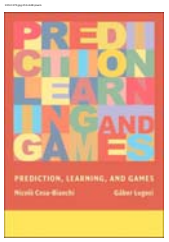
Help with the tutorial: Gunnar Rätsch

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

	E_1	E_2	E_3	$\cdots$	E_n	prediction	*true label*	loss
day 1	1	1	0	$\cdots$	0	0	1	1
day 2	1	0	1	$\cdots$	0	1	0	1
day 3	0	1	1	$\cdots$	1	1	1	0
day t	$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$\cdots$	$x_{t,n}$	$\hat{y}_t$	y_t	$\lvert y_t - \hat{y}_t \rvert$

Master Algorithm

For $t = 1$ To T Do

Get instance	$\boldsymbol{x}_t \in \{0,1\}^n$
Predict	$\hat{y}_t \in \{0,1\}$
Get label	$y_t \in \{0,1\}$
Incur loss	$\lvert y_t - \hat{y}_t \rvert$

More general online settings

Protocol:

For $t = 1$ To T Do

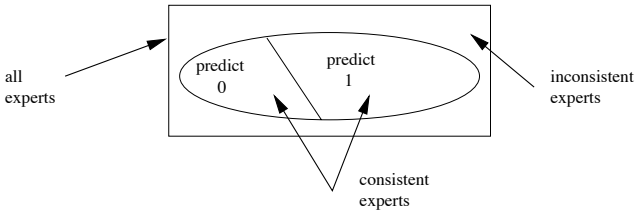
Get instance	$\boldsymbol{x}_t \in \mathbf{R}^n$
Predict	$\hat{y}_t \in \mathcal{Y}$
Get label	$y_t \in \mathcal{Y}$
Incur loss	$L(y_t, \hat{y}_t, \boldsymbol{x}_t)$

Problem instances:

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

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

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

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	*majority*	*true label*	loss
1	1	0	0	1	1	0	0	1	0	1
x	x	0	1	x	x	1	1	1	1	0
x	x	x	1	x	x	0	0	0	1	1
x	x	x	↑	x	x	x	x			
			consistent							

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 $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$

- total loss of algorithm A: $L_A(S) = \sum_{t=1}^{T} L(A(\boldsymbol{x}_t), y_t)$
- total loss of i-th expert E_i: $L_i(S) = \sum_{t=1}^{T} L(E_i(\boldsymbol{x}_t), y_t)$

Want bounds of the form:

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

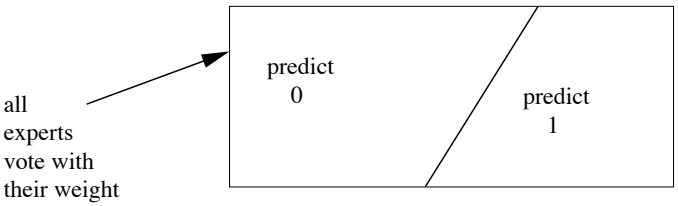
where a, b are constants

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

$$\begin{aligned} M_{t,i} &= \#\text{ of mistakes of } E_i \text{ before trial } t \\ w_{t,i} &= \beta^{M_{t,i}} \text{ weight of } E_i \text{ at beginning of trial } t \\ W_t &= \sum_{i=1}^{n} w_{t,i} \quad \text{total weight at trial } t \end{aligned}$$

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

If no mistake then minority multiplied by β:

$W_{t+1} \leq 1\ W_t$

If mistake then majority multiplied by β:

$$W_{t+1} \leq \underbrace{1 \quad \tfrac{1}{2}W_t}_{\text{minority}} + \underbrace{\beta \quad \tfrac{1}{2}W_t}_{\text{majority}} = \frac{1+\beta}{2}\ W_t$$

M = Total number of mistakes of WM

M_i = Total number of mistakes of expert E_i

Hence

$$\underset{\substack{total\ final \\ weight}}{W_{T+1}} \leq \left(\frac{1+\beta}{2}\right)^M W_1$$

$$W_{T+1} = \sum_{j=1}^{n} w_{T+1,j} = \sum_{j=1}^{n} \beta^{M_{T+1,j}} \geq \beta^{M_{T+1,i}}$$

where $i = \text{argmin}_j\, M_{T+1,j}$

We got:

$$\left(\frac{1+\beta}{2}\right)^M \underbrace{W_1}_{n} \geq \beta^{M_i}$$

Relative Loss bound for Weighted Majority

Solving for M:

$$M \leq \frac{\ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}} M_i + \frac{1}{\ln \frac{2}{1+\beta}} \ln n$$

$$\underset{\beta = 1/e}{M \leq} \underbrace{2.63}_{a} \min_i M_i + \underbrace{2.63}_{b} \ln n$$

For all sequences, loss of master algorithm is comparable to loss of best expert

$\Rightarrow$ Relative loss bounds [Fr]

Other Loss Functions

absolute loss	$L(y,\hat{y})$	$=$	$\lvert y-\hat{y}\rvert$
square loss	$L(y,\hat{y})$	$=$	$(y-\hat{y})^2$
entropic loss	$L(y,\hat{y})$	$=$	$y\ln\frac{y}{\hat{y}}+(1-y)\ln\frac{1-y}{1-\hat{y}}$, $y,\hat{y}\in[0,1]$
entropic loss $\pm$	$L(y,\hat{y})$	$=$	$\frac{1+y}{2}\ln\frac{1+y}{1+\hat{y}}+\frac{1-y}{2}\ln\frac{1-y}{1-\hat{y}}$, $y,\hat{y}\in[-1,+1]$
hellinger loss	$L(y,\hat{y})$	$=$	$\frac{1}{2}\left(\sqrt{1-y}-\sqrt{1-\hat{y}}\right)^2+\frac{1}{2}\left(\sqrt{y}-\sqrt{\hat{y}}\right)^2$ $y,\hat{y}\in[0,1]$

How does it work with other loss functions?

One weight per expert: [V]

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

where $L_{t,i}$ is total loss of E_i before trial t
and η is a positive learning rate

Master predicts with the weighted average (WA) [KW]

$$v_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^n w_{t,i}} \quad \text{normalized weights}$$

$$\hat{y}_t = \sum_{i=1}^n v_{t,i}\,x_{t,i} = \boldsymbol{v}_t\cdot\boldsymbol{x}_t$$

where $x_{t,i}$ is the prediction of E_i in trial t

Bounds for other Loss Functions

$\forall$ sequences S of examples $\langle(\boldsymbol{x}_t,y_t)\rangle_{1\le t\le T}$
where $\boldsymbol{x}_t\in[0,1]^n$ and $y_t\in[0,1]$

$$L_{WA}(S) \le \min_i \underbrace{1}_{a} L_i(S) + \underbrace{1/\eta}_{b}\ln(n)$$

$$\underbrace{L_{WA}(S)-\min_i L_i(S)}_{\text{regret}} \le b\ln(n)$$

Worst-case regret bounds

b	WA	fancy
entropic	1	1
square	2	1/2
hellinger	1	.71

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

Proof

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

Key inequality: $L(y, \boldsymbol{v}_t \cdot \boldsymbol{x}_t) \leq \overbrace{-\frac{1}{\eta}\ln W_{t+1}}^{\text{new potential}} - \overbrace{(-\frac{1}{\eta}\ln W_t)}^{\text{old potential}}$

$= \underbrace{-\frac{1}{\eta}\ln\frac{W_{t+1}}{W_t}}_{\text{drop of potential}}$

Proof continued

Telescoping:

$$\begin{aligned}
L_{\mathrm{WA}}(S) &\leq -\frac{1}{\eta}\ln\frac{W_{T+1}}{W_1} \\
&= -\frac{1}{\eta}\ln\sum_{j=1}^{n}\frac{1}{n}e^{-\eta L_{T+1,j}(S)} \\
&\leq -\frac{1}{\eta}\ln\frac{1}{n}e^{-\eta L_i(S)} \qquad i = \operatorname*{argmin}_j L_{T+1,j} \\
&= -\frac{1}{\eta}\ln\frac{1}{n}e^{-\eta L_{T+1,i}(S)} \\
&= L_{T+1,i}(S) + \frac{1}{\eta}\ln n
\end{aligned}$$

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

variables/experts

E_1	E_2	E_3	E_4	*true* *label*	$E_1 \vee E_3$	$E_3 \vee E_4$
1	1	0	0	0	1	0
1	0	1	0	1	1	1
0	1	1	1	0	1	1
0	1	0	0	1	0	0
$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$x_{t,4}$		↑	↑
					3	2
						mistakes

$E_1 \vee E_3$ becomes $\boldsymbol{u} = (1, 0, 1, 0)$

$E_1 \vee E_3$ is one on $\boldsymbol{x}_t \in \{0,1\}^n$ iff $\boldsymbol{u} \cdot \boldsymbol{x}_t \geq 1$

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 $\boldsymbol{x}_t \in \{0,1\}^n$

If $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \geq \theta$ then $\hat{y}_t = 1$

else $\hat{y}_t = 0$

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

If mistake then

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\,(\hat{y}_t - y_t)\,\boldsymbol{x}_t$$

Rotation invariant if $\boldsymbol{w}_1 = (0, \ldots, 0)$

k-literal Disjunctions with Perceptron

Perceptron Convergence Theorem ($\boldsymbol{w}_1 = (0, \ldots, 0)$, $\theta = \frac{1}{2}$, $\eta = \frac{1}{2n}$)

$$\#\text{ of mistakes} \leq 4\, A + 4\, k\, n$$

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 $\boldsymbol{x}_t \in \{0,1\}^n$

If $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \geq \theta$ then $\hat{y}_t = 1$

else $\hat{y}_t = 0$

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

If mistake then

$$w_{t+1,i} = w_{t,i}\, e^{-\eta\,(\hat{y}_t - y_t)\; x_{t,i}}$$

Mistake bound ($\boldsymbol{w}_1 := \frac{k}{n}(1,\ldots,1)$, $\theta = \frac{3\ln 3}{8}n$, $e^{-\eta} = \frac{1}{3}$) [AW]

$$\#\text{ of mistakes} \leq 4\,A \;+\; 3.6\,k \ln \frac{n}{k}$$

Not rotation invariant!

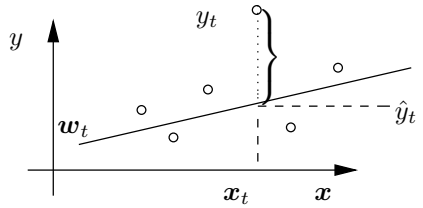
So far

- 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	$\boldsymbol{x}_t \in \mathbf{R}^n$
Predict	$\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t$
Get label	$y_t \in \mathbf{R}$
Incur loss	$L_t(\boldsymbol{w}_t) = (y_t - \hat{y}_t)^2$
Update	$\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$



Assume comparison class $\{\boldsymbol{u}\}$ is a set of linear predictors

$$\boldsymbol{u} \;:\; \boldsymbol{x} \to \boldsymbol{u} \cdot \boldsymbol{x}$$

Examples of Updates

Gradient descent

($\boldsymbol{w} \in \mathbf{R}^n$)

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla L_t(\boldsymbol{w}_t)$$

$$= \boldsymbol{w}_t - \eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\,\boldsymbol{x}_t \qquad \text{[WH]}$$

Exponentiated Gradient Algorithm [KW]

($\boldsymbol{w}$ is probability vector)

$$\boldsymbol{w}_{t+1,i} = w_{t,i}\; \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right] / \text{ normalization}$$

More examples of Updates

Unnormalized Exponentiated Gradient Algorithm [KW]
($\boldsymbol{w} \geq \mathbf{0}$)

$$\boldsymbol{w}_{t+1,i} = w_{t,i} \ \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]$$

Binary Exponentiated Gradient Algorithm [By]
($\boldsymbol{w} \in [0,1]^n$)

$$\boldsymbol{w}_{t+1,i} = \frac{w_{t,i} \exp\left[-\eta \frac{\partial Nt(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]}{1 - w_{t,i} + w_{t,i} \exp\left[-\eta \frac{\partial L_t(\boldsymbol{w}_t)}{\partial w_{t,i}}\right]}$$

More examples of Updates (cont)

p-norm Algorithms [GLS,GL]
($\boldsymbol{w} \in \mathbf{R}^n$)

$$\boldsymbol{w}_{t+1} = f^{-1}\big(f(\boldsymbol{w}_t) - \eta \nabla L_t(\boldsymbol{w}_t)\big)$$

where

$$f(\boldsymbol{w}) = \nabla \frac{1}{2}||\boldsymbol{w}||_q^2 = \nabla \frac{1}{2}\left(\sum_i |w_i|^q\right)^{2/q}$$

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

Motivation of Updates [KW]

Gradient descent

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{w}} \left(||\boldsymbol{w} - \boldsymbol{w}_t||_2^2/2 + \eta(y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2/2\right) \\ &= \boldsymbol{w}_t - \eta(\underbrace{\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t}_{\approx \boldsymbol{w}_t \cdot \boldsymbol{x}_t} - y_t)\, \boldsymbol{x}_t \end{aligned}$$

Exponentiated Gradient Algorithm

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{w}} \left(\sum_{i=1}^n w_i \ln \frac{w_i}{w_{t,i}} + \eta(y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2/2\right) \\ &= w_{t,i} \ \exp\left[-\eta\,(\underbrace{\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t}_{\approx \boldsymbol{w}_t \cdot \boldsymbol{x}_t} - y_t)\, x_{t,i}\right] / \text{ normalization} \end{aligned}$$

Families of update algorithms

parameter "divergence"	name of family	update algorithms
$\|\|\boldsymbol{w} - \boldsymbol{w}_t\|\|_2^2$	Gradient Descent	Widrow Hoff (LMS) 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)

parameter "divergence"	name of family	update algorithms
$\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}}$ $+w_{t,i} - w_i$	Unnormalized Exp. Grad. Alg.	Winnow
$\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}}$ $+(1-w_i) \ln \frac{1-w_i}{1-w_{t,i}}$	Binary Exp. Grad. Alg.	
any "Bregman divergence"	-	-

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

$\boldsymbol{w} \in \mathbf{R}^n$

$$\dot{\boldsymbol{w}}_t = -\eta \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$$

Unnormalized Exponentiated Gradient Alg.

$\boldsymbol{w} \geq \mathbf{0}$

$$\log(\dot{\boldsymbol{w}}_t) = -\eta \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$$

Alternate motivation: continuous updates (cont)

Discretization

$$\frac{\boldsymbol{f}(\boldsymbol{w}_{t+h}) - \boldsymbol{f}(w_t)}{h} = -\eta \nabla L_t(\boldsymbol{w}_t)$$

$$w_{t+h} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(w_t) - \eta \, h \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)\right)$$

We use $h = 1$

$$\boldsymbol{w}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{w}_t) - \eta_t \nabla L_t(\boldsymbol{w}_t)\right)$$

Conjecture: Forward Euler better:

Replace $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$ by $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_{t+h})$

Two main families

- Additive: $\boldsymbol{w} = \sum_t \alpha_t \boldsymbol{x}_t$ Multiplicative: $w_i \sim \exp(\sum_t \alpha_t x_{t,i})$

  $|| \, ||_2^2$ entropic

  Gradient descent Exponentiated Gradient

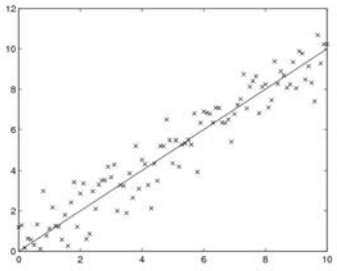
  Rotation invariant ???

- We now give a problem that favors the multiplicative updates

  Linear versus logarithmic dependence in n

Let's keep it simple

Linear Regression



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

A hard problem?

Hadamard Matrix:

$$\begin{array}{rcccc} & \rightarrow & +1 & +1 & +1 & +1 \\ instances & \rightarrow & +1 & -1 & +1 & -1 \\ & \rightarrow & +1 & +1 & -1 & -1 \\ & \rightarrow & +1 & -1 & -1 & +1 \\ & & \uparrow & \uparrow & \uparrow & \uparrow \\ & & \multicolumn{4}{c}{targets} \end{array}$$

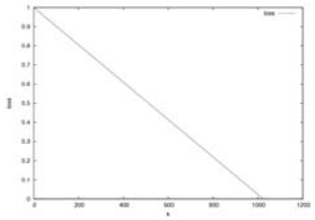
- n instances and n targets
- Instances are orthogonal
- Target weight vectors are units

Without embeddings I

Any linear combination of k training instances predicts zero on all $n - k$ test instances [LLW95,KWA97]

So loss 1 on $n - k$ of the n instances

Average square loss over all n instances is $\geq 1 - \frac{k}{n}$



$n = 1024$ $\lg(n) = 10$

Without embeddings II

Theorem

For any linear combination of k rows of the n-dimensional Hadamard matrix and any of the n targets

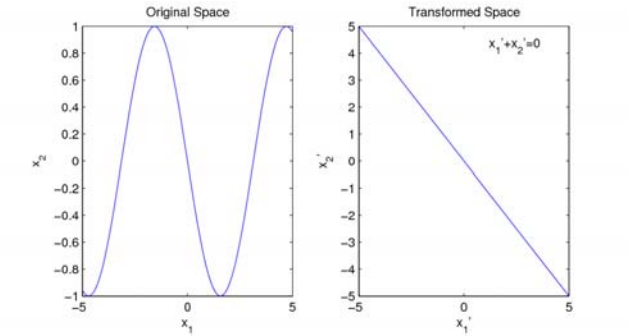
the average square loss over all n instances is

$$\geq 1 - \frac{k}{n}$$

Spiffing it up

Embed instance into a feature space

$$\phi : \mathbf{R}^n \rightarrow \mathbf{R}^m$$



$$\phi(x_1, x_2) = (\underbrace{\sin(x_1)}_{x_1'}, \underbrace{x_2}_{x_2'})$$

The Kernel Trick [BGV92]

If $\boldsymbol{w}$ linear combination of expanded instances, then

$$\hat{y} = \underbrace{\sum_t \alpha_t \, \phi(\boldsymbol{x}_t)}_{\boldsymbol{w}} \cdot \phi(\boldsymbol{x}) = \sum_t \alpha_t \, \underbrace{\phi(\boldsymbol{x}_t) \cdot \phi(\boldsymbol{x})}_{\mathbf{K}(\boldsymbol{x_t}, \boldsymbol{x})}$$

Kernel function $\mathbf{K}(\boldsymbol{x_t}, \boldsymbol{x})$ often efficient to compute

$$\phi(\underbrace{(x_1, \ldots, x_n)}_{n}) = \underbrace{(1, \ldots, x_i, \ldots, x_i x_j \ldots, x_i x_j x_k \ldots)}_{2^n \text{ products}}$$

Kernel magic

$$K(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{z}) = \underbrace{\sum_{I \subseteq 1..n} \prod_{i \in I} x_i \prod_{i \in I} z_i}_{O(2^n) \text{ time}} = \underbrace{\prod_{i=1}^{n} (1 + x_i z_i)}_{O(n) \text{ time}}$$

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]

$$\boldsymbol{w} = \underset{\boldsymbol{w}'}{\text{arginf}} \left(||\boldsymbol{w}'||^2 + \eta \sum_t (\boldsymbol{w}' \cdot \ \phi(\boldsymbol{x}_t) - y_t)^2 \right)$$

Solution $\boldsymbol{w}$ linear combination of the $\phi(\boldsymbol{x}_t)$

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

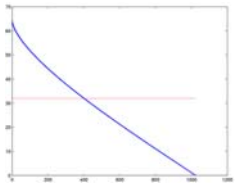
Theorem

- 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"



Hadamard Random

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

Additional Constraints

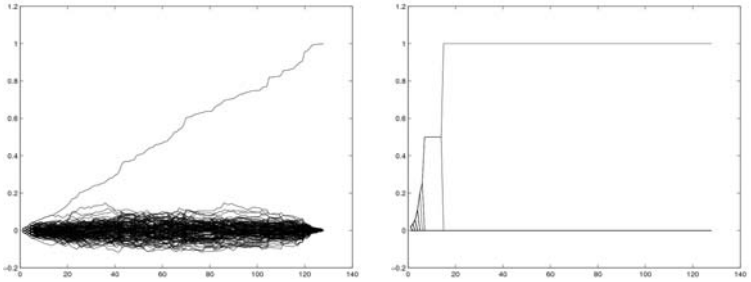
$$w_i \geq 0 \text{ and } \sum_{i=1}^{n} w_i = 1$$

Problem matrix

$$\begin{array}{cccccccc} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{array}$$

- 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

Effect?



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?

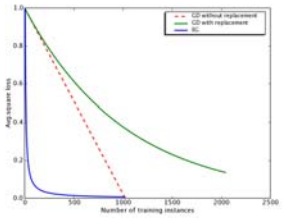
Use **Exponentiated Gradient Algorithm** [KW97]

Kernel methods $\boldsymbol{w} = \sum_t \alpha_t \Phi(\boldsymbol{x}_t)$

EG $w_j = \exp^{\sum_t \alpha_t \Phi(\boldsymbol{x}_t)_j} / \text{const}$

Now log weights linear combination of expanded instances

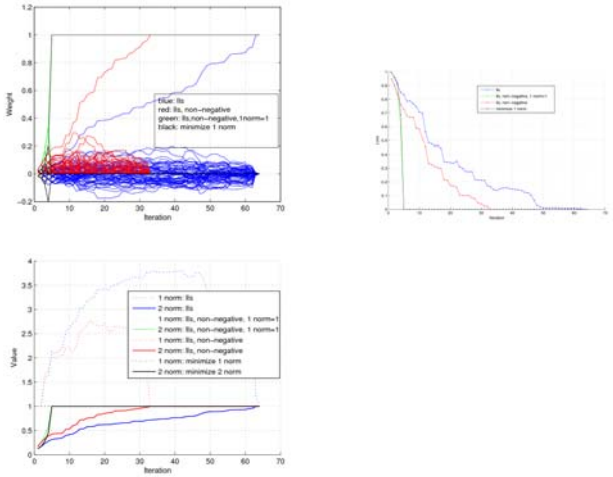
Average Squared Error



EG kernel algs

$\frac{\ln(n)}{k}$ $1 - \frac{k}{n}$ and $(1 - \frac{1}{n})^k$

Which constraints?



Incorporating side info

Kernel algorithms: **none** $\geq 1 - \mathbf{k/n}$
EG: $w_i \geq 0$ **and** $\sum_i w_i = 1$ $\mathbf{O}(\frac{\log \mathbf{n}}{\mathbf{k}})$

	$\rightarrow$	$+1.001$	$+1.002$	$+1.003$	$+1.004$
instances	$\rightarrow$	$+1.001$	-1.002	$+1.003$	-1.004
	$\rightarrow$	$+1.001$	$+1.002$	-1.003	-1.004
	$\rightarrow$	$+1.001$	-1.002	-1.003	$+1.004$
		$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
			targets		

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 kernal for a given problem?
- What is the hard problem for multiplicative updates?
- Can multiplicative updates be kernalized?
  Some cases are given by [TW]
- Can entropy regularization be replaced by $||\ ||_2^2$ 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

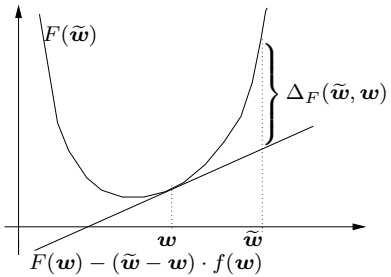
Goal: How can we prove relative loss bounds?

Bregman Divergences [Br,CL,Cs]

For any differentiable convex function F

$$\Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) = F(\widetilde{\boldsymbol{w}}) - F(\boldsymbol{w}) - (\widetilde{\boldsymbol{w}} - \boldsymbol{w}) \cdot \underbrace{\nabla_{\boldsymbol{w}} F(\boldsymbol{w})}_{f(\boldsymbol{w})}$$

$$= F(\widetilde{\boldsymbol{w}}) - \begin{matrix}\text{supporting hyperplane} \\ \text{through } (\boldsymbol{w}, F(\boldsymbol{w}))\end{matrix}$$



Bregman Divergences: Simple Properties

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

Bregman Divergences: more properties

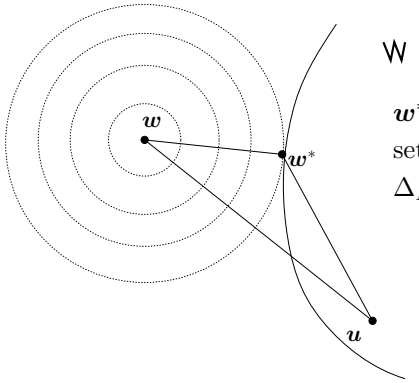
6. $\nabla_{\widetilde{\boldsymbol{w}}} \Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w})$

$$= \nabla F(\widetilde{\boldsymbol{w}}) - \nabla_{\widetilde{\boldsymbol{w}}}(\widetilde{\boldsymbol{w}} \nabla_{\boldsymbol{w}} F(\boldsymbol{w}))$$
$$= f(\widetilde{\boldsymbol{w}}) - f(\boldsymbol{w})$$

7. $\Delta_F(\boldsymbol{w}_1, \boldsymbol{w}_2) + \Delta_F(\boldsymbol{w}_2, \boldsymbol{w}_3)$

$$= F(\boldsymbol{w}_1) - F(\boldsymbol{w}_2) - (\boldsymbol{w}_1 - \boldsymbol{w}_2) f(\boldsymbol{w}_2)$$
$$F(\boldsymbol{w}_2) - F(\boldsymbol{w}_3) - (\boldsymbol{w}_2 - \boldsymbol{w}_3) f(\boldsymbol{w}_3)$$
$$= \Delta_F(\boldsymbol{w}_1, \boldsymbol{w}_3) + (\boldsymbol{w}_1 - \boldsymbol{w}_2) \cdot (f(\boldsymbol{w}_3) - f(\boldsymbol{w}_2))$$

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



$\boldsymbol{w}^*$ is projection of $\boldsymbol{w}$ onto convex set $\mathcal{W}$ w.r.t. Bregman divergence Δ_F:

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{W}} \Delta_F(\boldsymbol{u}, \boldsymbol{w})$$

Theorem:

$$\Delta_F(\boldsymbol{u}, w) \geq \Delta_F(\boldsymbol{u}, \boldsymbol{w}^*) + \Delta_F(\boldsymbol{w}^*, \boldsymbol{w})$$

Examples

Squared Euclidean Distance

$$\begin{aligned} F(\boldsymbol{w}) &= ||\boldsymbol{w}||_2^2/2 \\ f(\boldsymbol{w}) &= \boldsymbol{w} \\ \Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) &= ||\widetilde{\boldsymbol{w}}||_2^2/2 - ||\boldsymbol{w}||_2^2/2 - (\widetilde{\boldsymbol{w}} - \boldsymbol{w}) \cdot \boldsymbol{w} \\ &= ||\widetilde{\boldsymbol{w}} - \boldsymbol{w}||_2^2/2 \end{aligned}$$

(Unnormalized) Relative Entropy

$$\begin{aligned} F(\boldsymbol{w}) &= \sum_i (w_i \ln w_i - w_i) \\ f(\boldsymbol{w}) &= \ln \boldsymbol{w} \\ \Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) &= \sum_i \left(\widetilde{w_i} \ln \frac{\widetilde{w_i}}{w_i} + w_i - \widetilde{w_i} \right) \end{aligned}$$

Examples-2 [GLS,GL]

p-norm Algs (q is dual to p: $\frac{1}{p} + \frac{1}{q} = 1$)

$$\begin{aligned} F(\boldsymbol{w}) &= \frac{1}{2}||\boldsymbol{w}||_q^2 \\ f(\boldsymbol{w}) &= \nabla \frac{1}{2}||\boldsymbol{w}||_q^2 \\ \Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) &= \frac{1}{2}||\widetilde{\boldsymbol{w}}||_q^2 + \frac{1}{2}||\boldsymbol{w}||_q^2 - \widetilde{\boldsymbol{w}} \cdot f(\boldsymbol{w}) \end{aligned}$$

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

Examples-3

Burg entropy

$$\begin{aligned} F(\boldsymbol{w}) &= \sum_i -\ln w_i \\ f(\boldsymbol{w}) &= -\frac{1}{\boldsymbol{w}} \\ \Delta_F(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) &= \sum_i \left(-\ln \frac{\widetilde{w_i}}{w_i} + \frac{\widetilde{w_i}}{w_i} \right) - n \end{aligned}$$

Examples-4: div. between density matrices

Umegaki Divergence

$$\begin{aligned} F(\mathbf{W}) &= \mathrm{tr}\,(\mathbf{W} \ln \mathbf{W} - \mathbf{W}) \\ f(\mathbf{W}) &= \ln \mathbf{W} \\ \Delta_F(\widetilde{\mathbf{W}}, \mathbf{W}) &= \mathrm{tr}\left(\widetilde{\mathbf{W}}(\ln \widetilde{\mathbf{W}} - \ln \mathbf{W}) + \mathbf{W} - \widetilde{\mathbf{W}} \right) \end{aligned}$$

LogDet Divergence

$$\begin{aligned} F(\mathbf{W}) &= -\ln |\mathbf{W}| \\ f(\mathbf{W}) &= \mathbf{W}^{-1} \\ \Delta_F(\widetilde{\mathbf{W}}, \mathbf{W}) &= -\ln \frac{|\widetilde{\mathbf{W}}|}{|\mathbf{W}|}) + \mathrm{tr}(\widetilde{\mathbf{W}} \mathbf{W}^{-1}) - n \end{aligned}$$

General Motivation of Updates [KW]

Trade-off between two term:

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}}(\underbrace{\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t)}_{weight\ domain} + \eta_t \underbrace{L_t(\boldsymbol{w})}_{label\ domain})$$

$\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t)$ is "regularization term" and serves as measure of progress in the analysis.

When loss L is convex (in $\boldsymbol{w}$)

$$\nabla_{\boldsymbol{w}}(\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) + \eta_t L_t(\boldsymbol{w})) = 0$$

iff

$$f(\boldsymbol{w}) - f(\boldsymbol{w}_t) + \eta_t \underbrace{\nabla L_t(\boldsymbol{w})}_{\approx \nabla L_t(\boldsymbol{w}_t)} = 0$$

$$\Rightarrow \quad \boldsymbol{w}_{t+1} = f^{-1}\left(f(\boldsymbol{w}_t) - \eta_t \nabla L_t(\boldsymbol{w}_t)\right)$$

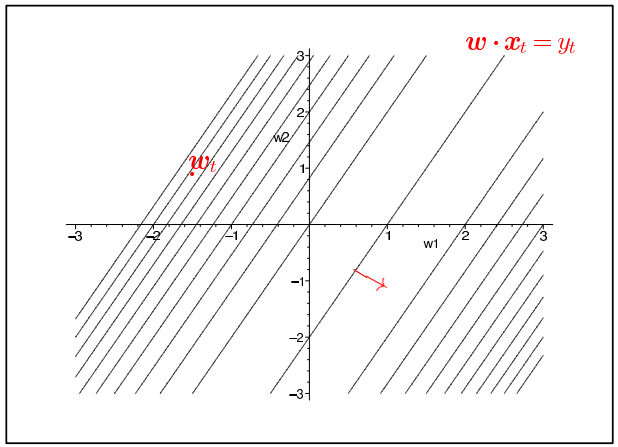
Quadratic Loss

$$L_t(\boldsymbol{w}) = (y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2/2$$

$\boldsymbol{w}_t = (-3/2, 1)$

$\boldsymbol{x}_t = (1, -0.5)$

$y_t = 1$



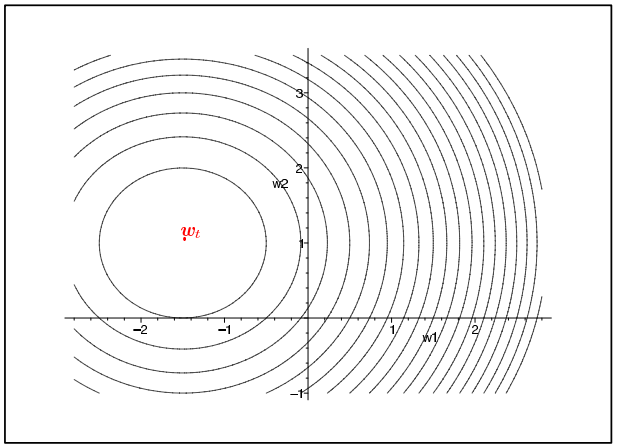
Divergence: Euclidean Distance Squared

$$\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) = \|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2/2$$

$\boldsymbol{w}_t = (-3/2, 1)$

$\boldsymbol{x}_t = (1, -0.5)$

$y_t = 1$



Loss + η Divergence

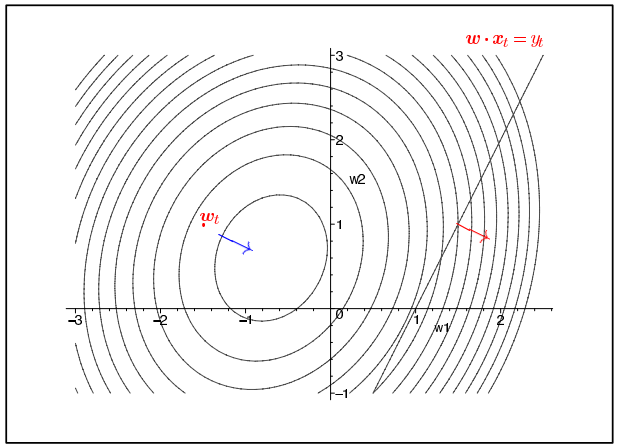
$$\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) = \|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2/2$$

$\boldsymbol{w}_t = (-3/2, 1)$

$\boldsymbol{x}_t = (1, -0.5)$

$y_t = 1$

$\eta = 0.2$



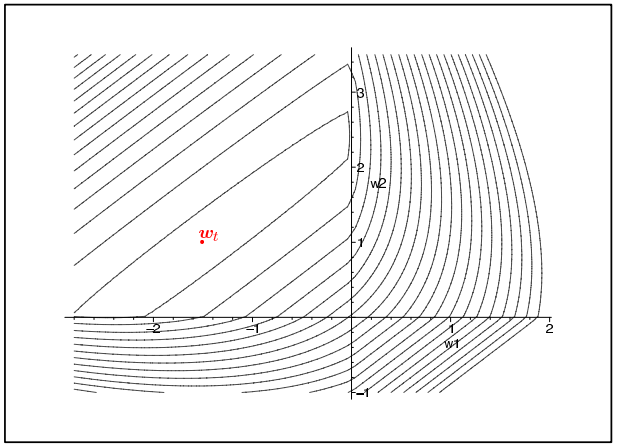
Divergence: 10-norm algorithm divergence

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

$\boldsymbol{w}_t = (-3/2, 1)$

$\boldsymbol{x}_t = (1, -0.5)$

$y_t = 1$



Loss + η Divergence

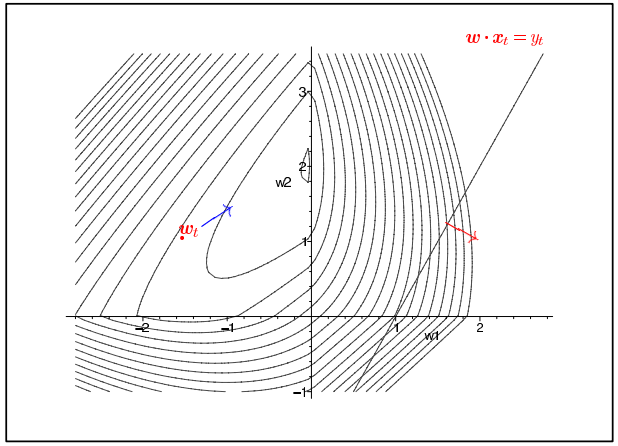
$\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) = \|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2/2$

$\boldsymbol{w}_t = (-3/2, 1)$

$\boldsymbol{x}_t = (1, -0.5)$

$y_t = 1$

$\eta = 0.2$



How to prove relative loss bounds?

Loss: $L_t(\boldsymbol{w}) = L((\boldsymbol{x}_t, y_t), \boldsymbol{w})$ convex in $\boldsymbol{w}$

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

Update: $f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t) = -\eta \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$

$$L_t(\boldsymbol{u}) \overbrace{\geq}^{\text{convexity}} L_t(\boldsymbol{w}_t) + (\boldsymbol{u} - \boldsymbol{w}_t) \cdot \underbrace{\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)}_{\text{update}}$$

$$= L_t(\boldsymbol{w}_t) - \frac{1}{\eta} \underbrace{(\boldsymbol{u} - \boldsymbol{w}_t) \cdot (f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t))}_{\text{prop. 7 of } \Delta_F}$$

$$= L_t(\boldsymbol{w}_t) + \frac{1}{\eta} (\Delta_F(\boldsymbol{u}, \boldsymbol{w}_{t+1}) - \Delta_F(\boldsymbol{u}, \boldsymbol{w}_t) - \Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1}))$$

First step: Teleskoping

Summing over t [WJ,KW]

$$\sum_t L_t(\boldsymbol{w}_t) \leq \sum_t L_t(\boldsymbol{u}) + \frac{1}{\eta} \sum_t \Big(\Delta_F(\boldsymbol{u}, \boldsymbol{w}_t) - \Delta_F(\boldsymbol{u}, \boldsymbol{w}_{t+1}) + \Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1}) \Big)$$

$$\leq \sum_t L_t(\boldsymbol{u}) + \frac{1}{\eta} \Big(\Delta_F(\boldsymbol{u}, \boldsymbol{w}_1) - \underbrace{\Delta_F(\boldsymbol{u}, \boldsymbol{w}_{T+1})}_{\geq 0} \Big) + \frac{1}{\eta} \sum_t \Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1})$$

$$\leq \sum_t L_t(\boldsymbol{u}) + \frac{1}{\eta} \Delta_F(\boldsymbol{u}, \boldsymbol{w}_1) + \frac{1}{\eta} \sum_t \Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1})$$

Any convex loss and any Bregman divergence!

Second step: Relate $\Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1})$ to loss $L_t(\boldsymbol{w}_t)$

Loss & divergence are dependent

Get $\Delta_F(\boldsymbol{w}_t, \boldsymbol{w}_{t+1}) \leq$ const. $L_t(\boldsymbol{w}_t)$

Then solve for $\sum_t L_t(\boldsymbol{w}_t)$

Yield bounds of the form

$$\sum_t L_t(\boldsymbol{w}_t) \leq a \sum_t L_t(\boldsymbol{u}) + b\, \Delta_F(\boldsymbol{u}, \boldsymbol{w}_1)$$

a, b constants, $a > 1$.

Regret bounds ($a = 1$):

time changing η, subtler analysis [AG]

Bounds for Linear Regression with Square Loss

Gradient Descent

$$\sum_t L_t(\boldsymbol{w}_t) \leq (1+c) \sum_t L_t(\boldsymbol{u}) + \frac{1+c}{c} X_2^2 U_2^2$$

$||\boldsymbol{x}_t||_2 \leq X_2$, $||\boldsymbol{u}||_2 \leq U_2$, $c > 0$, $\eta = f(c, X_2)$

Scaled Exponentiated Gradient

$$\sum_t L_t(\boldsymbol{w}_t) \leq (1+c) \sum_t L_t(\boldsymbol{u}) + \frac{1+c}{c} \ln n\, X_\infty^2 U_1^2$$

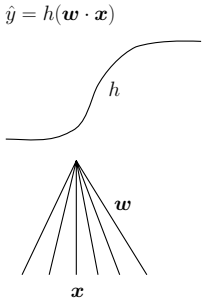
$||\boldsymbol{x}_t||_\infty \leq X_\infty$, $||\boldsymbol{u}||_1 \leq U_1$, $c > 0$, $\eta = f(c, X_\infty)$

p-norm Algorithm

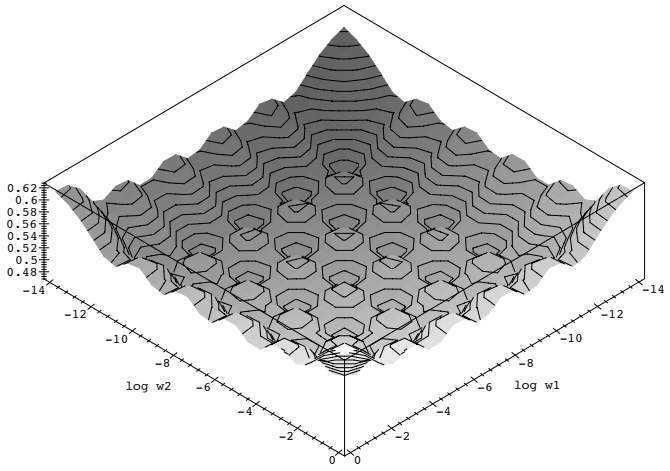
$$\sum_t L_t(\boldsymbol{w}_t) \leq (1+c) \sum_t L_t(\boldsymbol{u}) + \frac{1+c}{c} (p-1)\, X_p^2 U_q^2$$

$||\boldsymbol{x}_t||_p \leq X_p$, $||\boldsymbol{u}||_q \leq U_q$, $c > 0$, $\eta = f(c, X_\infty)$

Nonlinear Regression

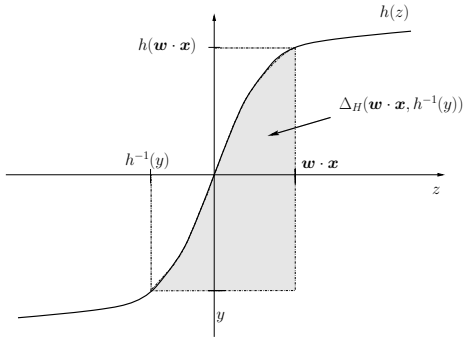


- Sigmoid function $h(z) = \frac{1}{1+e^{-z}}$
- For a set of examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$
  total loss $\sum_{t=1}^T h(\boldsymbol{w} \cdot \boldsymbol{x}) - y_t)^2/2$
  can have exponentially many minima
  in weight space [Bu,AHW]



Want loss that is convex in $\boldsymbol{w}$

Bregman Div. Lead to Good Loss Function



$(h = \nabla H)$

$$\int_{h^{-1}(y)}^{\boldsymbol{w}\cdot\boldsymbol{x}} (h(z) - y)\, dz = H(\boldsymbol{w}\cdot\boldsymbol{x}) - H(h^{-1}(y)) - (\boldsymbol{w}\cdot\boldsymbol{x} - h^{-1}(y))\, y$$
$$= \Delta_H(\boldsymbol{w}\cdot\boldsymbol{x}, h^{-1}(y))$$

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

Called matching loss for h [AHW,HKW]

Matching loss is convex in $\boldsymbol{w}$

transfer f. $h(z)$	$H(z)$	match. loss $d_H(\boldsymbol{w}\cdot\boldsymbol{x}, h^{-1}(y)$
z	$\frac{1}{2}z^2$	$\frac{1}{2}(\boldsymbol{w}\cdot\boldsymbol{x} - y)^2$ square loss
$\frac{e^z}{1+e^z}$	$\ln(1+e^z)$	$\ln(1+e^{\boldsymbol{w}\cdot\boldsymbol{x}}) - y\boldsymbol{w}\cdot\boldsymbol{x}$ $+y\ln y + (1-y)\ln(1-y)$ logistic loss
$\text{sign}(z)$	$\lvert z\rvert$	$\max\{0, -y\boldsymbol{w}\cdot\boldsymbol{x}\}$ hinge loss

Idea behind the matching loss

If transfer function and loss match, then

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

Then update has simple form:

$$f(\boldsymbol{w}_{t+1}) = f(\boldsymbol{w}_t) - \eta_t(h(\boldsymbol{w}_t\cdot\boldsymbol{x}) - y_t)\boldsymbol{x}_t$$

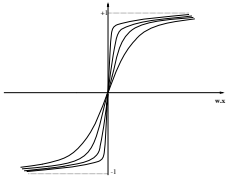
This can be exploited in proofs

But not absolutely necessary

One only needs convexity of $L(h(\boldsymbol{w}\cdot\boldsymbol{x}), y)$ in $\boldsymbol{w}$ [Ce]

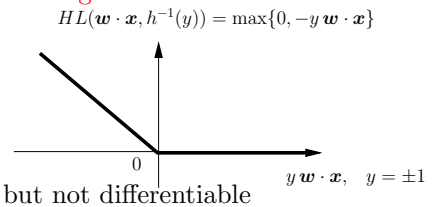
Sigmoid in the Limit

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



$H(z) = |z|$

Matching loss is hinge loss [GW]



Convex in $\boldsymbol{w}$ but not differentiable

Motivation of linear threshold algs

Gradient descent with Hinge Loss	Perceptron
Expon. gradient with Hinge Loss	Normalized Winnow

Known linear threshold algorithms for ± 1-classification case are gradient-based algorithms with hinge loss

Perceptron

$$\begin{aligned}
&\boldsymbol{w}_{t+1} \\
&= \operatorname*{argmin}_{\boldsymbol{w}} \left(||\boldsymbol{w} - \boldsymbol{w}_t||^2/2 + \eta\, HL(\boldsymbol{w} \cdot \boldsymbol{x}_t, g^{-1}(y_t)) \right) \\
&= \boldsymbol{w}_t - \eta \left(\operatorname{sign}(\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t) \right) - y_t)\, \boldsymbol{x}_t \\
&\approx \boldsymbol{w}_t - \eta \left(\underbrace{\operatorname{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)}_{\hat{y}_t} - y_t \right) \boldsymbol{x}_t
\end{aligned}$$

Normalized Winnow

$$\begin{aligned}
&\boldsymbol{w}_{t+1} \\
&= \operatorname*{argmin}_{\boldsymbol{w}} \left(\sum_{i=1}^{n} w_i \ln \frac{w_i}{w_{t,i}} + \eta\, HL(\boldsymbol{w} \cdot \boldsymbol{x}_t, g^{-1}(y_t)) \right) \\
&= w_{t,i}\; e^{-\eta\, (\operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x}_t) - y_t)\, x_{t,i}} \,/\, \text{normalization} \\
&\approx w_{t,i}\; e^{-\eta\, (\underbrace{\operatorname{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)}_{\hat{y}_t} - y_t)\, x_{t,i}} \,/\, \text{normalization}
\end{aligned}$$

Trade-off between two divergences [KW]

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} (\underbrace{\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t)}_{\text{parameter divergence}} + \eta_t \underbrace{\Delta_H(\boldsymbol{w} \cdot \boldsymbol{x}_t, h^{-1}(y_t))}_{\text{matching loss divergence}}$$

Both divergences are convex in $\boldsymbol{w}$

$$\boldsymbol{w}_{t+1} = f^{-1} \left(f(\boldsymbol{w}_t) - \eta_t (h(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) - y_t) \boldsymbol{x}_t \right)$$

Generalization of the "delta"-rule

Duality

Special case:

$$\min_{\boldsymbol{w}} \Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) + \Delta_H(\boldsymbol{x}_t \cdot \boldsymbol{w}, h^{-1}(y_t))$$
$$= -\min_{\alpha} \Delta_{\mathcal{G}}(\alpha + y_t, h(0)) + \Delta_{\mathcal{F}}(f(\boldsymbol{w}_t) - \alpha \boldsymbol{x}_t, f(\mathbf{0})) + \text{const.}$$

General:

$$\min_{\boldsymbol{w}} \Delta_F(\boldsymbol{w} + \boldsymbol{\mu}, \underbrace{f^{-1}(\boldsymbol{\phi})}_{\boldsymbol{w}_t}) + \Delta_H(X\boldsymbol{w} + \boldsymbol{\nu}, h^{-1}(\boldsymbol{y}))$$
$$= -\min_{\boldsymbol{\alpha}} \Delta_{\mathcal{G}}(\boldsymbol{\alpha} + \boldsymbol{y}, h(\boldsymbol{\nu})) + \Delta_{\mathcal{F}}(\boldsymbol{\phi} - X^\top \boldsymbol{\alpha}, f(\boldsymbol{\mu})) + \text{const.}$$

where F and $\mathcal{F}$ are convex conjugate functions:

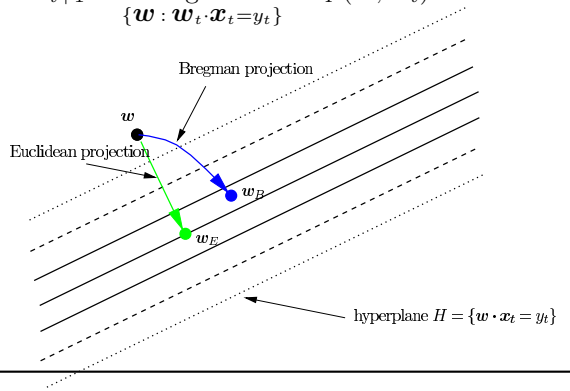
$$\mathcal{F}(\boldsymbol{x}) = \sup_{\boldsymbol{y}} \boldsymbol{x} \cdot \boldsymbol{y} - F(\boldsymbol{y}) = \boldsymbol{x} \cdot (\nabla F)^{-1}(\boldsymbol{x}) - F((\nabla F)^{-1}(\boldsymbol{x}))$$

Projections onto Hyperplanes

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} (\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) + \eta(\boldsymbol{w} \cdot \boldsymbol{x}_t - y_t)^2)$$

When η is large then $\boldsymbol{w}_{t+1}$ is projection of $\boldsymbol{w}_t$ onto plane $\boldsymbol{w} \cdot \boldsymbol{x}_t = y_t$

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\{\boldsymbol{w} \,:\, \boldsymbol{w}_t \cdot \boldsymbol{x}_t = y_t\}} \Delta_F(\boldsymbol{w}, \boldsymbol{w}_t)$$



Relation to Boosting

The AdaBoost update of the probability vector $\boldsymbol{w}_t$:

$$w_i^{(t+1)} = w_i^{(t)} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))$$

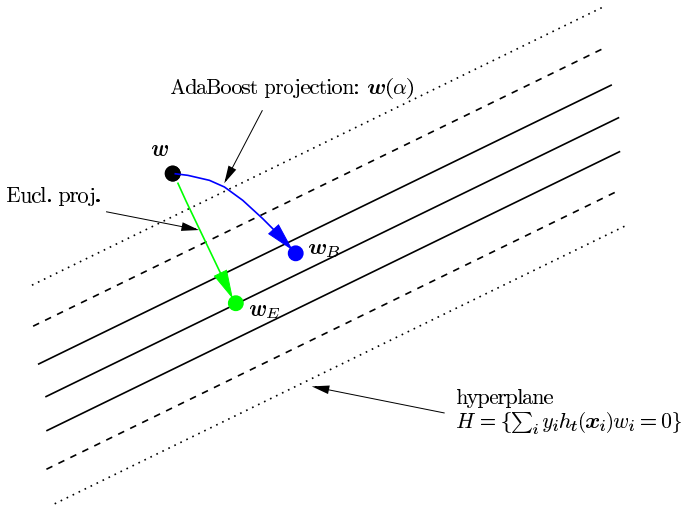
Is a projection w.r.t. divergence [CKW,La,KW,CSS]

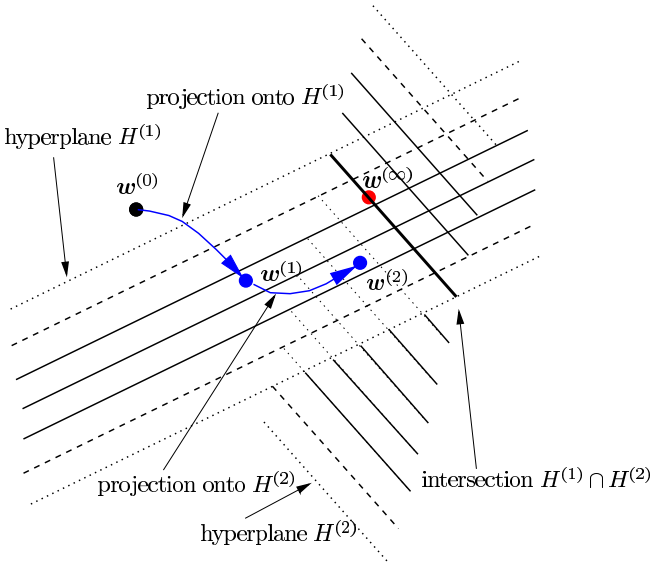
$$\Delta_F(\boldsymbol{w}, \boldsymbol{w}_t) = \sum_i w_i \ln \frac{w_i}{w_{t,i}}$$

Such that the weighted training error of h_t w.r.t. $\boldsymbol{w}^{(t+1)}$ is $\frac{1}{2}$

("diversification" of Boosting mentioned in Ron Meir's talk)

$$w_i(\alpha) = w_i^{(t)} \exp(-\alpha y_i h_t(\boldsymbol{x}_i)) \qquad \alpha = \operatorname*{argmin}_{\alpha'} \sum_i w_i(\alpha')$$





Bregman divergences and exponential families?

- Exponential family of distributions
- Inherent duality

$$\boldsymbol{w}_{t+1} = f^{-1}\left(f(\boldsymbol{w}_t) - \eta \nabla L_t(\boldsymbol{w}_t)\right)$$

primal param. dual param.

$$\begin{array}{ccc} \boldsymbol{w}_t & \xrightarrow{f} & f(\boldsymbol{w}_t) \\ \boldsymbol{w}_{t+1} & \xleftarrow{f^{-1}} & -\eta \nabla L_t(\boldsymbol{w}_t) \end{array}$$

Exponential Family of Distributions

- Parametric density functions

$$P_G(\boldsymbol{x}|\boldsymbol{\theta}) = e^{\boldsymbol{\theta}\cdot\boldsymbol{x} - G(\boldsymbol{\theta})}\, P_0(\boldsymbol{x})$$

- $\boldsymbol{\theta}$ and $\boldsymbol{x}$ vectors in R^d
- Cumulant function $G(\boldsymbol{\theta})$ assures normalization

$$G(\boldsymbol{\theta}) = \ln \int e^{\boldsymbol{\theta}\cdot\boldsymbol{x}}\, P_0(\boldsymbol{x})\, d\boldsymbol{x}$$

- $G(\boldsymbol{\theta})$ is convex function on convex set $\boldsymbol{\Theta} \subseteq R^d$
- G characterizes members of the family
- $\boldsymbol{\theta}$ is *natural* parameter

- Expectation parameter

$$\boldsymbol{\mu} = \int_{\boldsymbol{x}} \boldsymbol{x} P_G(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x} = E_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\boldsymbol{\theta})$$

  where $g(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} G(\boldsymbol{\theta})$

- Second convex function $F(\boldsymbol{\mu})$ on space $g(\boldsymbol{\Theta})$

$$F(\boldsymbol{\mu}) \;=\; \boldsymbol{\theta}\cdot\boldsymbol{\mu} - G(\boldsymbol{\theta})$$

- $G(\boldsymbol{\theta})$ and $F(\boldsymbol{\mu})$ are *convex conjugate* functions
- Let $f(\boldsymbol{\mu}) = \nabla_{\boldsymbol{\mu}} F(\boldsymbol{\mu})$
- $f(\boldsymbol{\mu}) = g^{-1}(\boldsymbol{\mu})$

Primal & Dual Parameters

natural paramater		expectation parameter
$\boldsymbol{\theta}$	$\xrightarrow{g}$ $\xleftarrow{f}$	$\boldsymbol{\mu}$
$G(\boldsymbol{\theta})$		$F(\boldsymbol{\mu})$

- $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ are dual parameters
- Parameter transformations
  $g(\boldsymbol{\theta}) = \boldsymbol{\mu}$ and $f(\boldsymbol{\mu}) = \boldsymbol{\theta}$ [A,BN]

Gaussian (unit variance)

$$\begin{aligned} P(\boldsymbol{x}|\boldsymbol{\theta}) &\sim e^{-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{x})^2} \\ &= e^{\boldsymbol{\theta}\cdot\boldsymbol{x}-\frac{1}{2}\boldsymbol{\theta}^2} e^{\frac{1}{2}\boldsymbol{x}^2} \end{aligned}$$

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

Parameter transformations:
$g(\boldsymbol{\theta}) = \boldsymbol{\theta} = \boldsymbol{\mu}$ and $f(\boldsymbol{\mu}) = \boldsymbol{\mu} = \boldsymbol{\theta}$

Dual convex function: $\begin{aligned} F(\boldsymbol{\mu}) &= \boldsymbol{\theta}\cdot\boldsymbol{\mu} - G(\boldsymbol{\theta}) \\ &= \frac{1}{2}\boldsymbol{\mu}^2 \end{aligned}$

Square loss: $L_t(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}_t - \boldsymbol{x}_t)^2$

Bernoulli

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

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

μ 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:

$$\mu = g(\theta) = \frac{e^\theta}{1+e^\theta} \text{ and } \theta = f(\mu) = \ln\frac{\mu}{1-\mu}$$

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

Log loss: $\begin{aligned} L_t(\theta) &= -x_t\theta + ln(1+e^\theta) \\ &= -x_t\ln\mu - (1-x_t)\ln(1-\mu) \end{aligned}$

Poisson

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

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

μ 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:

$$\mu = g(\theta) = e^\theta \text{ and } \theta = f(\mu) = \ln\mu$$

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

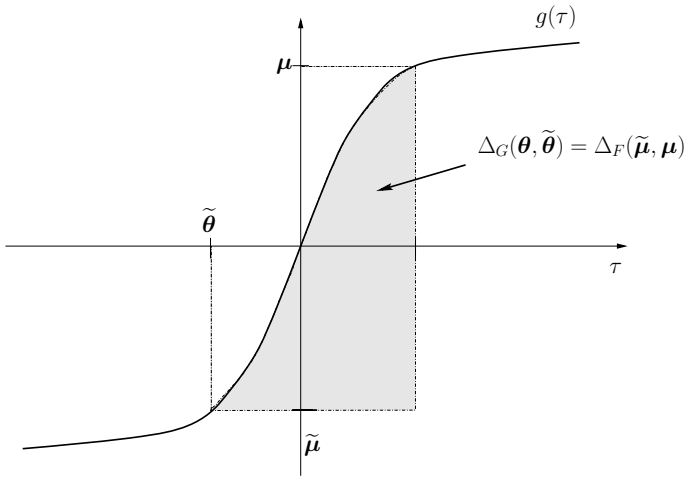
Loss: $\begin{aligned} L_t(\theta) &= -x_t\theta + e^\theta + \ln x_t! \\ &= -x_t\ln\mu + \mu + \ln x_t! \end{aligned}$

Bregman Div. as Rel. Ent. between Distributions

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

$$\begin{aligned}
\Delta_G(\widetilde{\boldsymbol{\theta}},\boldsymbol{\theta}) &= \int_{\boldsymbol{x}} P_G(\boldsymbol{x}|\boldsymbol{\theta})\ln\frac{P_G(\boldsymbol{x}|\boldsymbol{\theta})}{P_G(\boldsymbol{x}|\widetilde{\boldsymbol{\theta}})}d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}} P_G(\boldsymbol{x}|\boldsymbol{\theta})(\boldsymbol{\theta}\cdot\boldsymbol{x}-G(\theta)-\boldsymbol{\theta}\cdot\boldsymbol{x}+G(\widetilde{\boldsymbol{\theta}}))d\boldsymbol{x} \\
&= G(\widetilde{\boldsymbol{\theta}})-G(\boldsymbol{\theta})-(\widetilde{\boldsymbol{\theta}}-\boldsymbol{\theta})\cdot(\int_{\boldsymbol{x}} P_G(\boldsymbol{x}|\boldsymbol{\theta})\boldsymbol{x}d\boldsymbol{x}) \\
&= G(\widetilde{\boldsymbol{\theta}})-G(\boldsymbol{\theta})-(\widetilde{\boldsymbol{\theta}}-\boldsymbol{\theta})\cdot\boldsymbol{\mu} \\
\overset{F(\boldsymbol{\mu})=\boldsymbol{\theta}\cdot\boldsymbol{\mu}-G(\boldsymbol{\theta})}{=} & F(\boldsymbol{\mu})-F(\widetilde{\boldsymbol{\mu}})-(\boldsymbol{\mu}-\widetilde{\boldsymbol{\mu}})\cdot\widetilde{\boldsymbol{\theta}} \\
&= \Delta_F(\boldsymbol{\mu},\widetilde{\boldsymbol{\mu}}) \qquad [A,BN,AW]
\end{aligned}$$

Area unchanged When Slide Flipped



$$\begin{aligned}
\Delta_G(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}) &= G(\boldsymbol{\theta})-G(\widetilde{\boldsymbol{\theta}})-(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})\cdot g(\widetilde{\boldsymbol{\theta}}) \\
&= \int_{\widetilde{\boldsymbol{\theta}}}^{\boldsymbol{\theta}} (g(\boldsymbol{\tau})-g(\widetilde{\boldsymbol{\theta}}))\cdot d\boldsymbol{\tau} \\
\overset{\text{flip}}{=} & \int_{\boldsymbol{\mu}}^{\widetilde{\boldsymbol{\mu}}} (f(\boldsymbol{\sigma})-f(\boldsymbol{\mu}))\cdot d\boldsymbol{\sigma} \\
&= F(\widetilde{\boldsymbol{\mu}})-F(\boldsymbol{\mu})-(\widetilde{\boldsymbol{\mu}}-\boldsymbol{\mu})\cdot f(\boldsymbol{\mu}) \\
&= \Delta_F(\widetilde{\boldsymbol{\mu}},\boldsymbol{\mu})
\end{aligned}$$

Dual divergence for Bernoulli

$$G(\boldsymbol{\theta})=\ln(1+e^{\theta}) \qquad F(\mu)=\mu\ln\mu+(1-\mu)\ln(1-\mu)$$

$$g(\theta)=\tfrac{e^{\theta}}{1+e^{\theta}}=\mu \qquad f(\mu)=\ln\tfrac{\mu}{1-\mu}=\theta$$

$$\Delta_G(\widetilde{\theta},\theta)=\ln(1+e^{\widetilde{\theta}})-\ln(1+e^{\theta})-(\widetilde{\theta}-\theta)\frac{e^{\theta}}{1+e^{\theta}}$$

$$\Delta_F(\mu,\widetilde{\mu})=\mu\ln\frac{\mu}{\widetilde{\mu}}+(1-\mu)\ln\frac{1-\mu}{1-\widetilde{\mu}}$$

Binary relative entropy

Sum of binary relative entropies is parameter divergence for BEG

Dual divergence for Poisson

$$G(\boldsymbol{\theta}) = e^{\theta} \qquad F(\mu) = \mu \ln \mu - \mu$$

$$g(\theta) = e^{\theta} = \mu \quad f(\mu) = \ln \mu = \theta$$

$$\Delta_G(\widetilde{\theta}, \theta) = e^{\widetilde{\theta}} - e^{\theta} - (\widetilde{\theta} - \theta)e^{\theta}$$

$$\Delta_F(\mu, \widetilde{\mu}) = \mu \ln \frac{\mu}{\widetilde{\mu}} + \widetilde{\mu} - \mu$$

Unnormalized relative entropy

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

Dual matching loss for sigmoid transfer func.

$$H(z) = \ln(1 + e^z) \qquad K(r) = r \ln r + (1 - r) \ln(1 - r)$$
$$h(z) = \tfrac{e^z}{1+e^z} = r \qquad k(r) = \ln \tfrac{r}{1-r} = z$$

K dual to H and $k = h^{-1}$

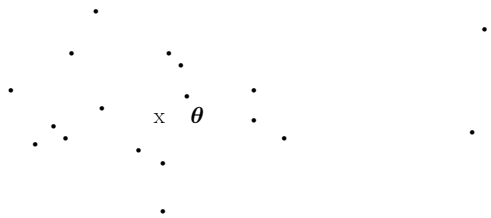
$$\Delta_H(\boldsymbol{w} \cdot \boldsymbol{x}, h^{-1}(y))$$
$$= \ln(1 + e^{\boldsymbol{w} \cdot \boldsymbol{x}}) - y \boldsymbol{w} \cdot \boldsymbol{x} + y \ln y + (1 - y) \ln(1 - y)$$

By duality logistic loss is same as entropic loss

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

Matching loss for logistic transfer function

Example: Gaussian density estimation



Off-line versus on-line

- Loss on example $\boldsymbol{x}_t$

$$L_t(\boldsymbol{\theta}) = -\ln P(\boldsymbol{x}_t|\boldsymbol{\theta}) \quad = \quad \frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\theta})^2$$

Derivation of Updates

- Want to bound

$$\sum_{t=1}^{T} L_t(\boldsymbol{\theta}_t) - \inf_{\boldsymbol{\theta}} L_{1..T}(\boldsymbol{\theta})$$

- Off-line algorithm has all T examples

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$$

- Setup for choosing best parameter setting

$$\boldsymbol{\theta}_B = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(\eta_B^{-1} \underbrace{\Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_1)}_{\text{divergence to initial}} + \underbrace{L_{1..T}(\boldsymbol{\theta})}_{\text{total loss}})$$

Here $\eta_B^{-1} > 0$ is a tradeoff parameter

On-line Algorithm [AW]

- In trial t, the first t examples

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$$

  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

$$\boldsymbol{\theta}_{t+1} = \underset{\boldsymbol{\theta}}{\text{argmin}} \; (\eta_1^{-1} \underbrace{\Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_1)}_{\substack{\text{divergence} \\ \text{to initial}}} + \underbrace{L_{1..t}(\boldsymbol{\theta})}_{\substack{\text{loss} \\ \text{so far}}})$$

  Tradeoff parameter $\eta_1^{-1} \geq 0$

Alternate Motivation of Same On-Line Update

$$\boldsymbol{\theta}_{t+1} = \underset{\boldsymbol{\theta}}{\text{argmin}} (\eta_t^{-1} \underbrace{\Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_t)}_{\text{divergence to last}} + \underbrace{L_t(\boldsymbol{\theta})}_{\text{current loss}})$$

where $\eta_t = \dfrac{1}{\eta_1^{-1} + t - 1}$

Parameter Updates

Off-line: $\boldsymbol{\mu}_B = \dfrac{\eta_B^{-1}\boldsymbol{\mu}_1 + \sum_{t=1}^{T} \boldsymbol{x}_t}{\eta_B^{-1} + T}$

On-Line in trial t: $\boldsymbol{\mu}_{t+1} = \dfrac{\eta_1^{-1}\boldsymbol{\mu}_1 + \sum_{q=1}^{t} \boldsymbol{x}_q}{\eta_1^{-1} + t} = \boldsymbol{\mu}_t - \eta_{t+1}(\boldsymbol{\mu}_t - \boldsymbol{x}_t)$

$$\boldsymbol{\theta}_{t+1} \quad = \quad g^{-1}\left(g(\boldsymbol{\theta}_t) - \eta_{t+1}(\boldsymbol{\mu}_t - \boldsymbol{x}_t)\right)$$

- On-line algorithm has freedom to use a tradeoff parameter η_1^{-1} that could be different from the off-line parameter η_B^{-1}

- Two choices for η_1^{-1}

  Case $\eta_1^{-1} = \eta_B^{-1}$:

  Incremental Off-Line Algorithm

  Case $\eta_1^{-1} = \eta_B^{-1} + 1$:

  Forward Algorithm [V]

Shrinkage Towards Initial

$$\boldsymbol{\mu}_B = \overline{\boldsymbol{x}_T} - \eta_B^{-1}(\eta_B^{-1} + T)^{-1}(\overline{\boldsymbol{x}_T} - \boldsymbol{\mu}_1)$$

where $\overline{\boldsymbol{x}_T} = \frac{\sum_{t=1}^{T} \boldsymbol{x}_t}{T}$

Shrinkage factor $\eta_B^{-1}(\eta_B^{-1} + T)^{-1}$

	Off-line	Forward on-line
Gauss $\boldsymbol{\mu}_1 = 0, \eta_B^{-1} = 0$	$\boldsymbol{\mu}_B = \frac{\sum_{t=1}^{T} \boldsymbol{x}_t}{T}$	$\boldsymbol{\mu}_t = \frac{\sum_{q=1}^{t-1} \boldsymbol{x}_q}{t-1+1}$
Bernoulli $\boldsymbol{\mu}_1 = \frac{1}{2}, \eta_B^{-1} = 0$	$\boldsymbol{\mu}_B = \frac{\sum_{t=1}^{T} \boldsymbol{x}_t}{T}$	$\boldsymbol{\mu}_t = \frac{\frac{1}{2}+\sum_{q=1}^{t-1} \boldsymbol{x}_q}{t-1+1}$ [KT]

Key Lemma [AW]

For any example $\boldsymbol{x}_t$ and any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

$$\underbrace{L_t(\boldsymbol{\theta}_t)}_{\text{loss of algorithm}} - \underbrace{L_t(\boldsymbol{\theta})}_{\text{loss of comparator } \boldsymbol{\theta}}$$

$$= \underbrace{\eta_t^{-1} \; \Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_t)}_{\text{divergence to last par.}} - \underbrace{\eta_{t+1}^{-1} \; \Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_{t+1})}_{\text{divergence to updated par.}}$$

$$+ \underbrace{\eta_{t+1}^{-1} \Delta_G(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1})}_{\text{cost of update}}$$

Main Theorem

For any sequence of examples and any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

$$\underbrace{\textstyle\sum_{t=1}^{T} L_t(\boldsymbol{\theta}_t)}_{\text{total loss of algorithm}} - \underbrace{L_{1..T}(\boldsymbol{\theta})}_{\text{total loss of comparator } \boldsymbol{\theta}}$$

$$= \underbrace{\eta_1^{-1} \; \Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_1)}_{\text{divergence to initial par.}} - \underbrace{\eta_{T+1}^{-1} \; \Delta_G(\boldsymbol{\theta}, \boldsymbol{\theta}_{T+1})}_{\text{divergence to last par.}}$$

$$+ \underbrace{\textstyle\sum_{t=1}^{T} \eta_{t+1}^{-1} \Delta_G(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1})}_{\text{cost of all updates}}$$

Proven by simply summing the Key Lemma

Bounds for the Forward Algorithm

$$\sum_{t=1}^{T} L_t(\boldsymbol{\theta}_t) - \inf_{\boldsymbol{\theta}} L_{1..T}(\boldsymbol{\theta}) \overset{\text{Gauss}}{=} \sum_{t=1}^{T} \eta_t \, \boldsymbol{x}_t^2/2 - \sum_{t=1}^{T-1} \eta_t \, \boldsymbol{\mu}_{t+1}^2/2 \qquad [AW]$$

$$\leq \frac{X^2}{2} \ln(1 + \frac{T}{\eta_1^{-1} - 1})$$

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

$$\overset{\text{lin. regr.}}{\leq} \frac{1}{2} Y^2 n \ln\left(1 + \frac{TX^2}{a}\right) \qquad [V, Fo, AW]$$

$X^2 = \max_{t=1}^{T} \boldsymbol{x}_t^2$, $Y = \max_{t=1}^{T} y_t$, $\boldsymbol{w}_t = \left(a\boldsymbol{I} + \sum_{q=1}^{t} \boldsymbol{x}_q \boldsymbol{x}_q'\right)^{-1} \sum_{q=1}^{t-1} \boldsymbol{x}_q y_q$

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

$$\inf_{\boldsymbol{\mu}_1}\sup_{\boldsymbol{x}_1}\inf_{\boldsymbol{\mu}_2}\sup_{\boldsymbol{x}_2}\inf_{\boldsymbol{\mu}_3}\sup_{\boldsymbol{x}_3}\ \ldots\ \inf_{\boldsymbol{\mu}_T}\sup_{\boldsymbol{x}_T}$$

$$\sum_{t=1}^{T}\tfrac{1}{2}(\boldsymbol{\mu}_t-\boldsymbol{x}_t)^2 \quad - \quad \inf_{\boldsymbol{\mu}}\sum_{t=1}^{T}\tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{x}_t)^2$$

total loss of on-line algorithm — total loss of off-line algorithm

Instances must be bounded: $||\boldsymbol{x}_t||_2 \le X$

Minimax algorithm usually intractable

Gaussian and Bernoulli are exceptions [TW,Sh]

Gaussian

Forward Alg. $\quad \boldsymbol{\mu}_t = \frac{\sum_{q=1}^{t-1}\boldsymbol{x}_q}{t}$

Bound $\quad \frac{1}{2}X^2(1+\ln T)$

Minimax Alg. $\quad \boldsymbol{\mu}_t = \frac{\sum_{q=1}^{t-1}\boldsymbol{x}_q}{t+\ln T-\ln(t+O(\ln T))}$

Bound $\quad \frac{1}{2}X^2(\ln T-\ln\ln T)+o(1)$

Minimax alg. needs to know T

Last-step Minimax

Assumes that current trial is last trial [Fo,TW]

$$\begin{aligned}\boldsymbol{\mu}_t &= \operatorname*{arginf}_{\boldsymbol{\mu}} \sup_{\boldsymbol{x}_t} \sum_{q=1}^{t} L_q(\boldsymbol{\mu}_q) - \inf_{\boldsymbol{\mu}} L_{1..t}(\boldsymbol{\mu}) \\ &= \operatorname*{arginf}_{\boldsymbol{\mu}} \sup_{\boldsymbol{x}_t} L_t(\boldsymbol{\mu}_t) - \inf_{\boldsymbol{\mu}} L_{1..t}(\boldsymbol{\mu})\end{aligned}$$

For Gaussian and linear regression
Last-step Minimax is same as Forward Alg.

Last-step Minimax: Bernoulli

Forward alg:

$$\boldsymbol{\mu}_t = \frac{s+\frac{1}{2}}{t-1+1}, \text{ where } s = \sum_{q=1}^{t-1} x_q$$

Last-step:

$$\boldsymbol{\mu}_t = \frac{(s+1)^{(}s+1)(t-s-1)^{t-s-1}}{s^s(t-s)^{t-s}+(s+1)^{s+1}(t-s-1)^{t-s-1}}$$

Worst-case regret bounds: $\ln(T+1)+\mathbf{c}$

Forward: $\mathbf{c}=\frac{\ln\pi}{\mathbf{2}}$ Last step: $\mathbf{c}=\frac{\mathbf{1}}{\mathbf{2}}$

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 $L : \mathbf{R}^2 \to \mathbf{R}$

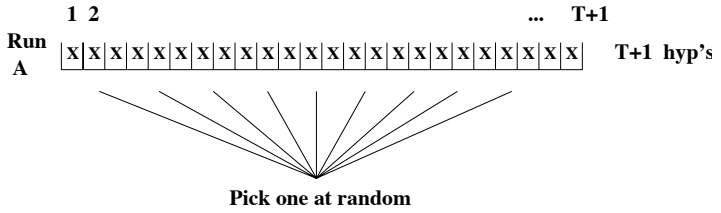
$S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T) \sim D^T$

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

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

$$\begin{aligned}
&\mathrm{E}_{S\sim D^T}(\text{TotLoss}(A,S)) \\
&= \mathrm{E}_{(e_1,\ldots,e_T)\sim D^T}\left(\sum_{t=1}^{T} L(A((e_1,\ldots,e_{t-1})),e_t)\right) \\
&= \sum_{t=1}^{T} \mathrm{E}_{(e_1,\ldots,e_{t-1})\sim D^{t-1}}\left(\mathrm{E}_{e\sim D}\left(L(A((e_1,\ldots,e_{t-1})),e)\right)\right) \\
&= \sum_{t=1}^{T} \mathrm{E}_{(e_1,\ldots,e_{t-1})\sim D^{t-1}}\left(\text{InstLoss}(A(e_1,\ldots,e_{t-1}),D)\right)
\end{aligned}$$

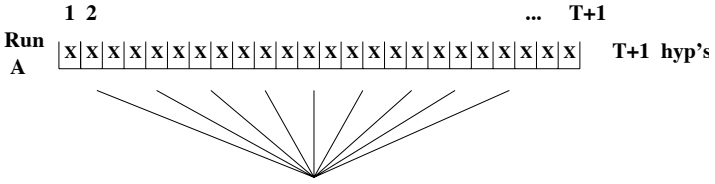
So expected total loss is total instaneous loss



- Choose h_i uniformly at random from the $T+1$ hypotheses
- On new instance $\boldsymbol{x}$ predict with $h_i(\boldsymbol{x})$
- Instanteneous loss of this algorithm is expected total loss of original algorithm over $T+1$
- Applied to the Perceptron Algorithm [FS]

Tail bound [CCG]

Convex Loss $L : \mathbf{R}^2 \to [0, L_{\max}]$



$$\overline{h}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x})$$

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

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

Application: Adaptive Channel Equalization

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

- Common approach:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta(y - \boldsymbol{w}_t \cdot \boldsymbol{x}_t)\boldsymbol{x}_t$$

- But: Many coefficients are zero, or close to zero [MSWJ]
  $\Rightarrow$ Use Unnormalized Exponentiated Gradient update or the approximate version

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t\,(1 - \eta(y - \boldsymbol{w}_t \cdot \boldsymbol{x}_t)\boldsymbol{x}_t)$$

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

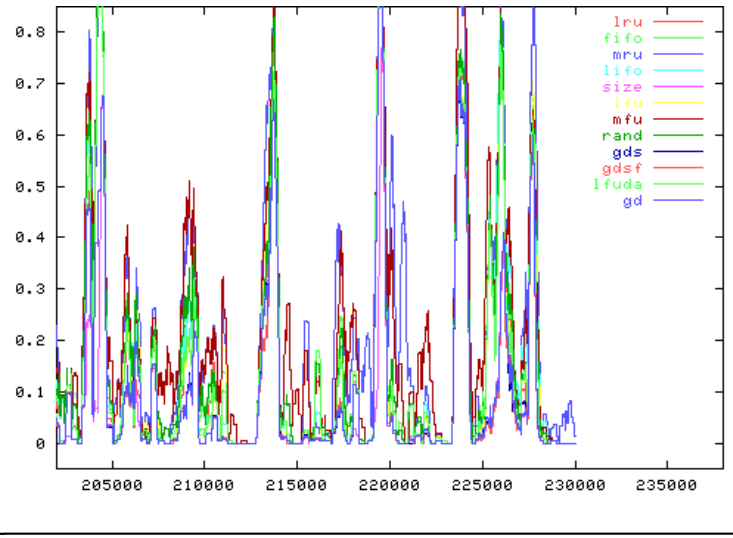
Caching Policies

- 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

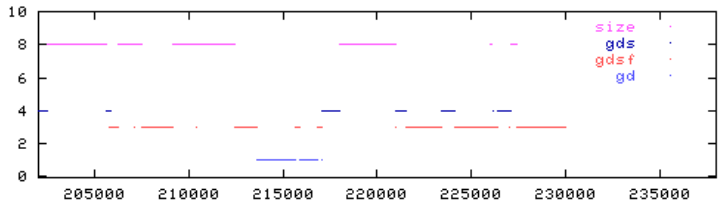
Which Policy to Choose?

- 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

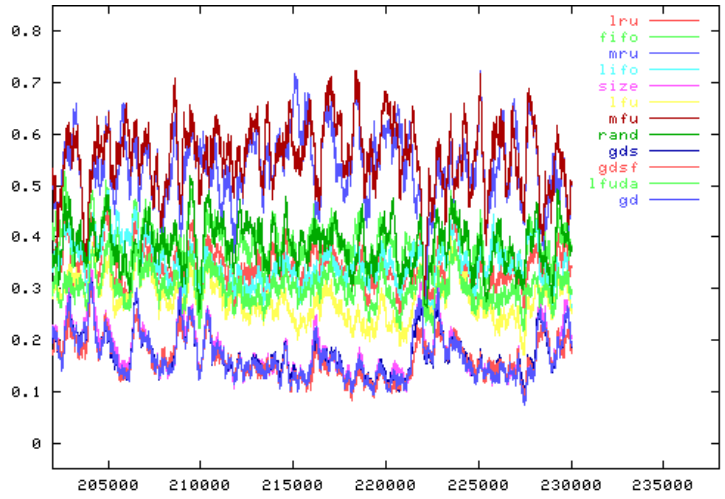
Characteristics Vary with Time



Best Policy Varies with time



Randomly Permuted Request Stream

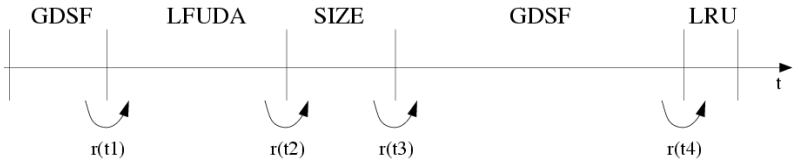


Want "Adaptive" Policy

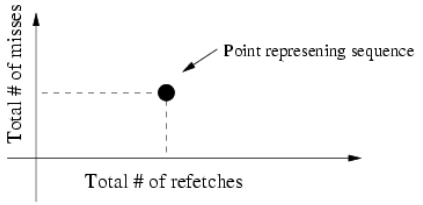
- Good compared to off-line comparator
  - **BestFixed**: a posteriori best of 12 policies on entire request stream
  - **BestRefetching**(R):
    minimum number of misses with at most R refetches in any sequence of switching policies

Refetches & Policy Switches

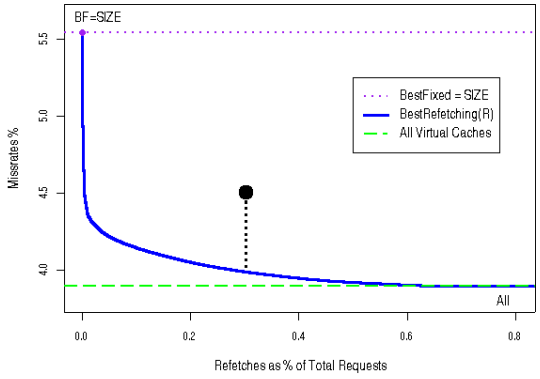
Off-line comparator: All sequences of the form



We plot miss rate v.s. refetches:



BestRefetching(R)

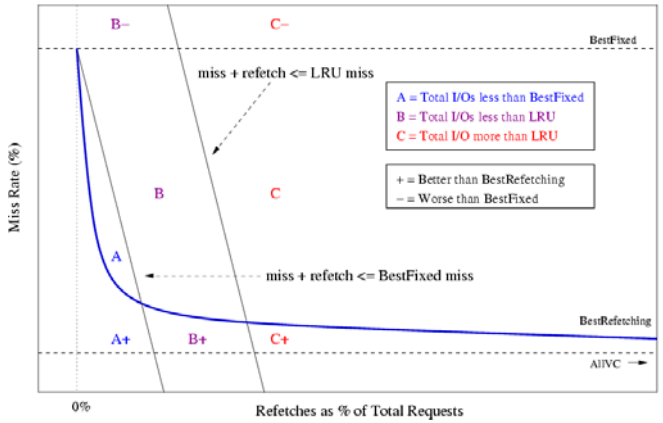


Dynamic programming in time $O(RN^2T)$

Goal for On-line policies

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

Score Card

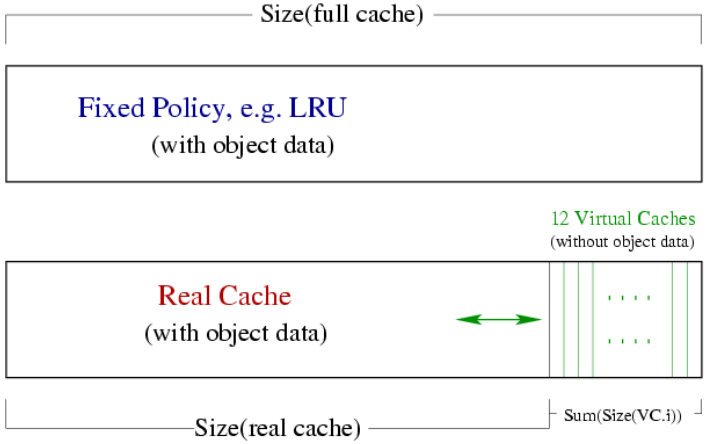


Key Idea: Virtual Caches

- 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:

$$\text{Size(real cache)} = \text{Size(full cache)} - \sum_{i=1}^{12} \text{Size(VC}_i\text{)}.$$

Virtual Caches



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

Fixed Share to Uniform Past

Loss Update:

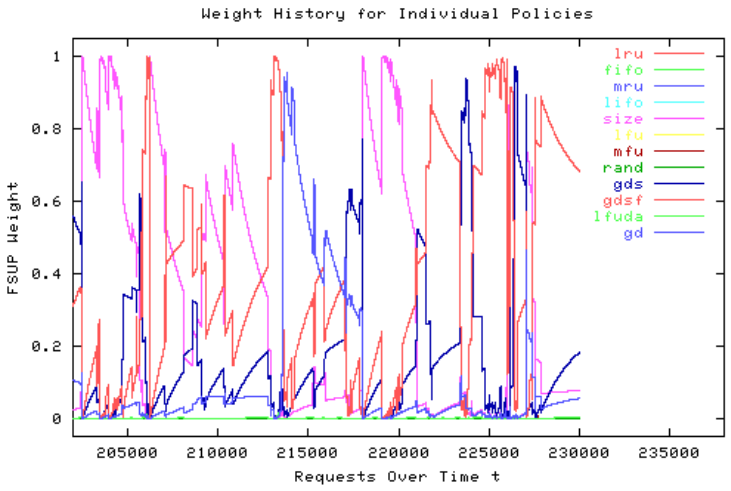
$$w'_{t,i} = \frac{w_{t,i}\beta^{\,\mathrm{miss}_{t,i}}}{\text{normaliz.}}, \quad \beta \in (0,1)$$

Share Update:

$$\mathbf{w}_{t+1} = (1-\alpha)\,\mathbf{w}'_t + \alpha\,\mathbf{r}_{t-1},$$
$$\text{where } \mathbf{r}_{t-1} = \sum_{q=1}^{t-1} \mathbf{w}'_q/(t-1)$$

- 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						
	E_1	E_2	E_3	E_n	prediction	*true label*	loss
day 1	1	1	0	0	0	1	1
day 2	1	0	1	0	1	0	1
day 3	0	1	1	1	1	1	0
day t	$x_{t,1}$	$x_{t,2}$	$x_{t,3}$	$x_{t,n}$	$\hat{y}_t$	y_t	$(y_t - \hat{y}_t)^2$

- Choose comparison class of predictors (experts)
- Master Algorithm combines predictions of experts
- $\mathbf{x}_t$ vector of expert's predictions

Protocol of Master Algorithm

Loop for each trial $t = 1, .., T$

Get next instance	$\mathbf{x}_t$
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

$$\forall S: \quad L_{1..T,A} \leq \min_i \left(L_{1..T,i} + \underbrace{c \log n}_{\text{bits}} \right)$$

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

$$\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$$

- The weights are updated according to the Loss Update

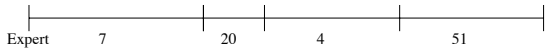
$$w_{t+1,i} := \frac{w_{t,i} \quad e^{-\eta \, L_{t,i}}}{\text{normaliz.}}, \quad e^{-\eta} = \beta$$

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 ?



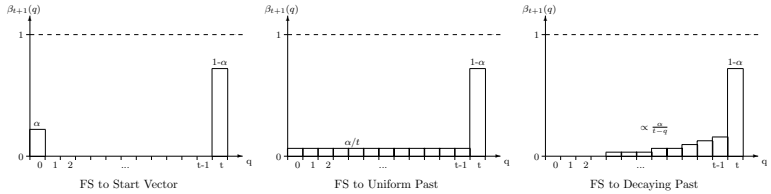
- Off-line algorithm partitions sequence into sections and chooses best expert in each section
- Goal:
  Do well compared to the best off-line partition
- Problem:
  Loss Update learns too well
  and does not recover fast enough

Mixing Update

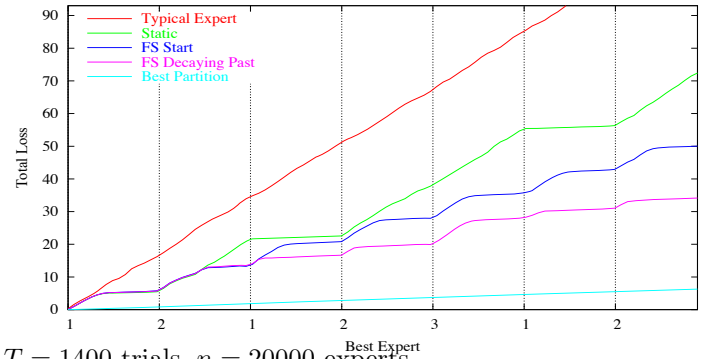
- Predict $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$
- Loss Update $w'_{t,i} = \frac{w_{t,i} e^{-\eta L_{t,i}}}{\text{normaliz.}}$
- Mixing Update

$$\mathbf{w}_{t+1} = \sum_{q=0}^{t} \beta_{t+1,q} \mathbf{w}'_q, \quad \text{where} \quad \sum_{q=0}^{t} \beta_{t+1} = 1$$

- Mixing schemes

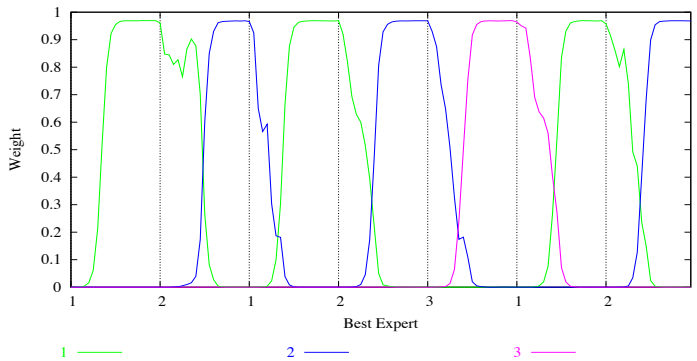


Total Loss Plots



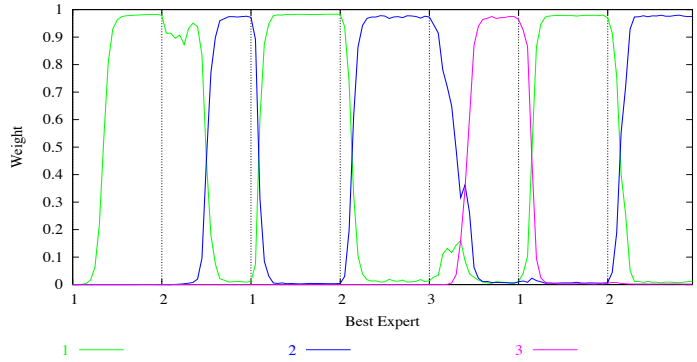
- $T = 1400$ trials, $n = 20000$ experts
- $k = 6$ shifts (every 200 trials)

Weights of Fixed Share to Start Vector Alg.



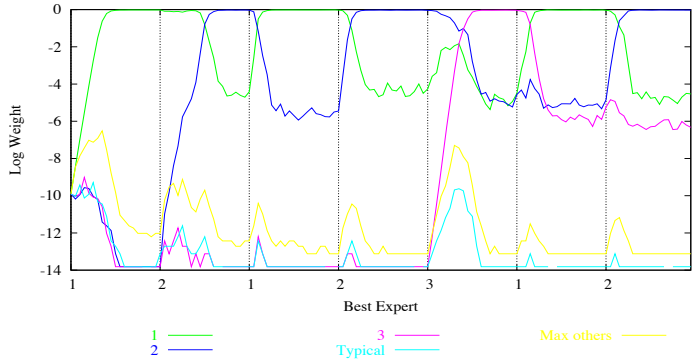
Weights of Fixed Share to Decaying Past Alg.

- Improved recovery when expert used before

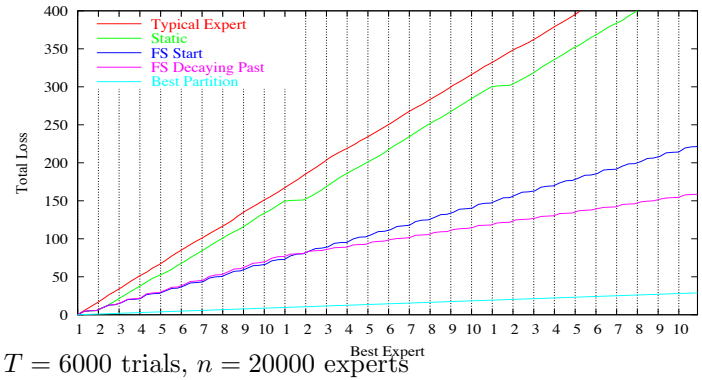


Fixed Share to Decaying Past - Log Weights

- Past good experts remain at higher level



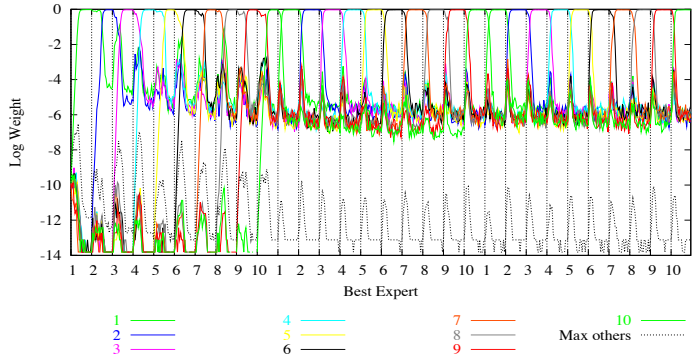
More Experts Remembered



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

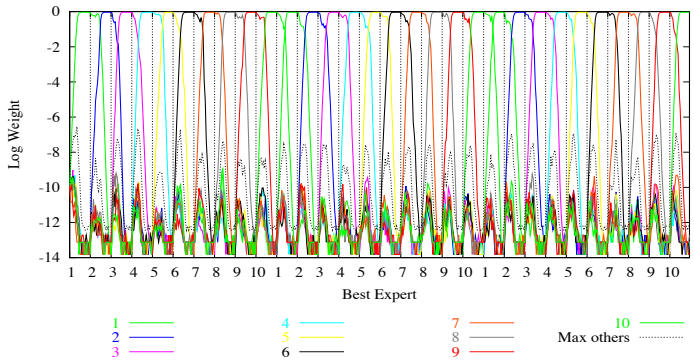
Fixed Share to Decaying Past - Log Weights

- Past good expert are cached



Fixed Share to Start Vector - Log Weights

- No memory



Relative Loss Bounds

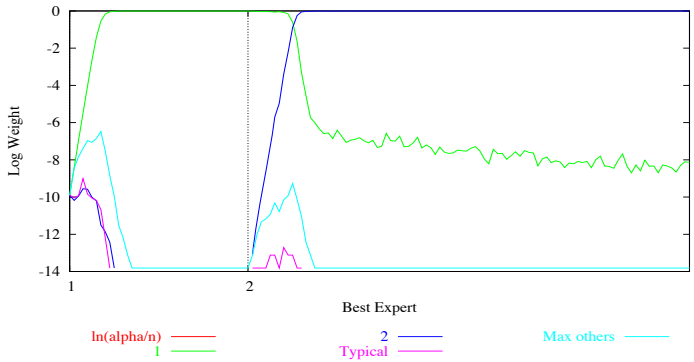
- Always have the form

$$L_{1..T,A} \leq \min_{P} \left(L_{1..T,P} + O(\#\text{ of bits for } P) \right)$$

→ Boundaries are encoded twice

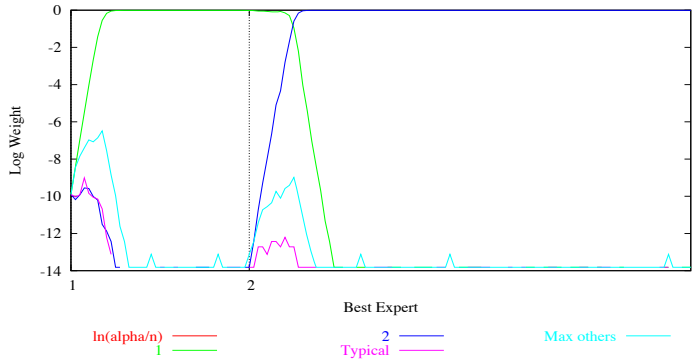
→ Off-line problem NP-complete

Fixed Share to Decaying Past - Log Weights



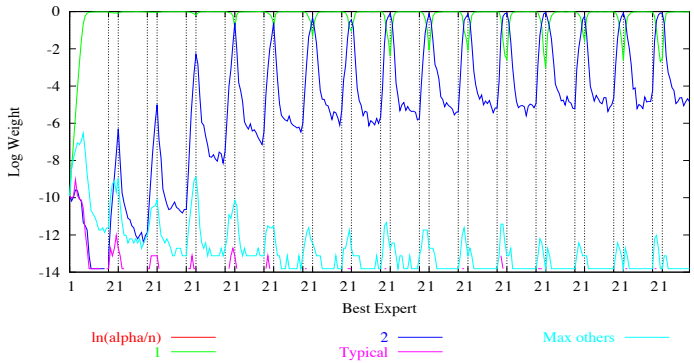
- Larger alpha gives better long-term memory

Fixed Share to Start Vector - Log Weights



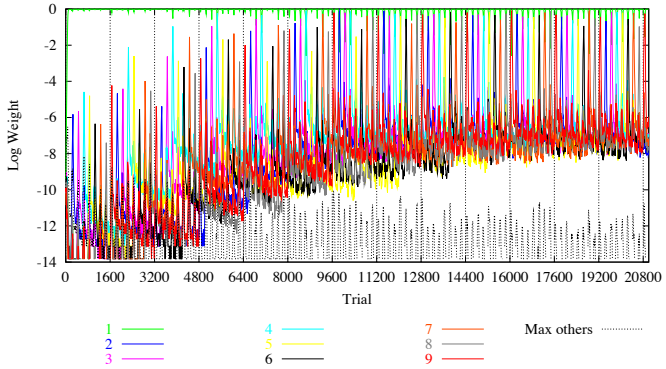
Memory from many short sections accumulates

- Fixed Share to Decaying Past - Log Weights



Bigger memory

- Cycling thru 10 different short sections



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 (α, β or η) not crucial
  - fix at

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

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:

object	o_{12}	o_7	o_2	o_{22}	o_3	o_6	o_2	o_{15}	o_9
priority	31.2	30.2	24.1	17.1	9.3	8	4.1	2.5	1.2
rank	9	8	7	6	5	4	3	2	1

- o_9 first discarded
- o_{12} last

Master Rank

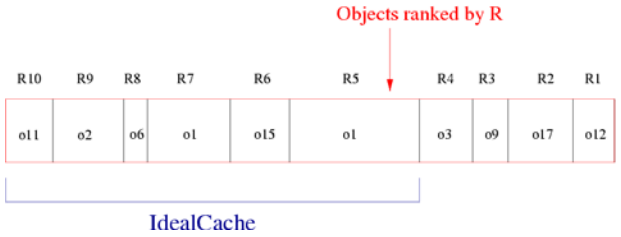
- Master priority P constructed from weights and ranks of virtual caches

$$P_o = \begin{cases} \sum_{n:o \in \mathrm{VC}_n} w_n r_{n,o} & \text{if } \exists n : o \in \mathrm{VC}_n \\ 0 & \text{if } \forall n : o \notin \mathrm{VC}_n \end{cases}$$

- R is corresponding master rank

Ideal Cache

- Highest ranked objects fill the ideal cache to capacity
- IdealCache =



Managing Real Cache: Instantaneous Rollover

- Keep RealCache = Ideal Cache
  i.e. Refetch all $o \in$ IdealCache $-$ RealCache
- Too much refetching

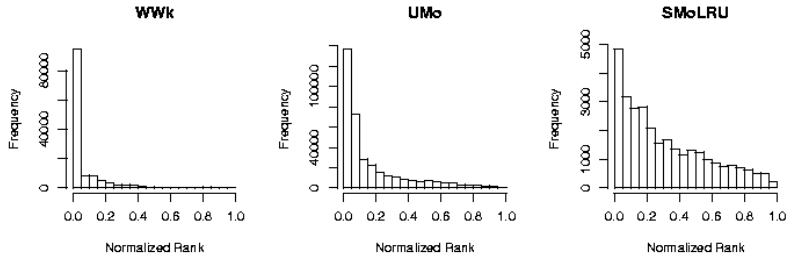
Demand Rollover

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

Compromise: Background Rollover

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

Smart Refetching



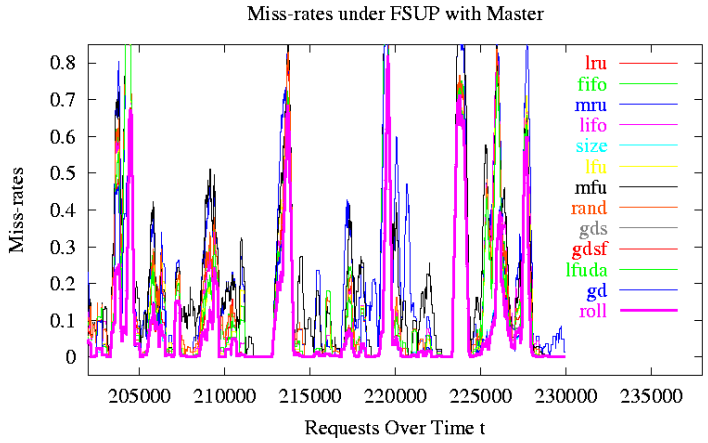
- Most hits in real cache have high R-rank
- Refetch only top 40-60% of R-ranked objects

Experimental Results: Filesystem Data

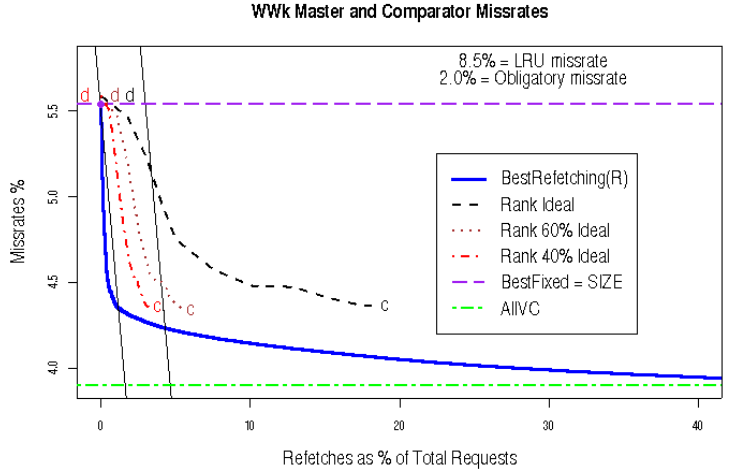
Dataset:	Work-Week (WWk)	User-Month (UMo)	Server-Month-LRU (SMoLRU)
#Requests	138k	382k	48k
Cache size	900KB	2MB	4MB
%Skipped	6.5%	12.8%	15.7%
# Compuls	0.020	0.015	0.152
LRU Miss Rate	0.166	0.076	0.870
BestFixed Pol / MR	SIZE 0.055	GDS 0.075	GDSF 0.399
%<LRU	36.8%	54.7%	54.2%

CMU DFStrace

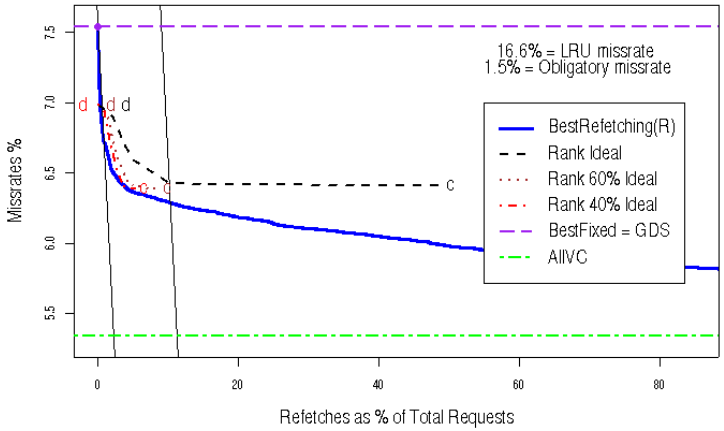
Demand Rollover "Tracks" best policy



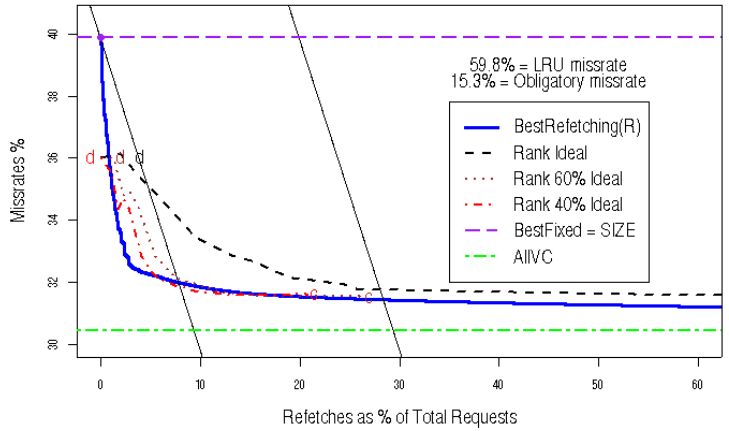
WWk



UMo



SMoLRU



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

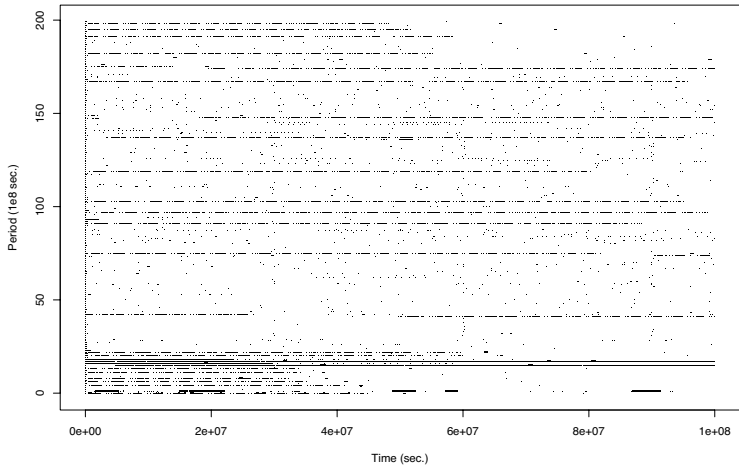
- Use existing caching strategies as experts
- Use set of fine-grained experts
  from which all existing caching policies are built
- Machine Learners will get interested if there are realistic benchmark data sets

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



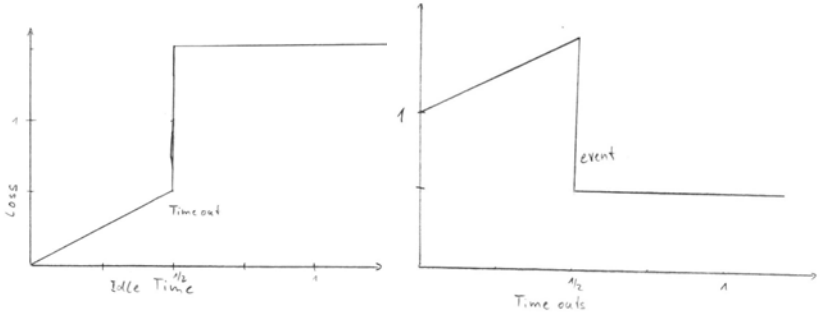
Idea

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

Which loss?

Costs for spinning up/down, running machine in idle mode, ...

L("idle-time", "time-out") $\sim$ 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 competitve ratios

Other Applications

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