MLSS 2010 Boosting Tutorial
Recent Advances in Boosting

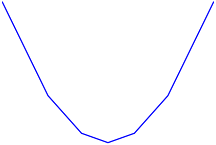
Part I: Entropy Regularized LPBoost
Part II: Boosting from an Optimization Perspective

Manfred K. Warmuth - UCSC
prepared jointly with S.V.N. Vishwanathan - Purdue
adapted from ICML 2009 tutorial

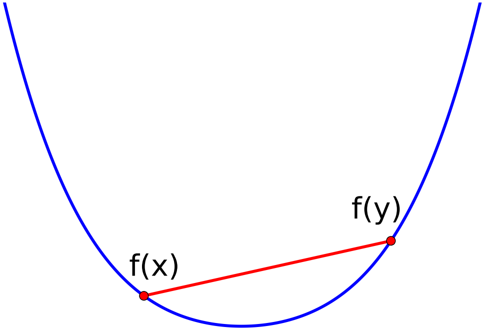
Updated September 26, 2010

Minimizing the Max Edge is a Convex Optimization Problem

$$\min_{\mathbf{d}\in\mathcal{S}^N} \underbrace{\max_{\mathbf{u}\in\mathcal{U}} \langle \mathbf{u}, \mathbf{d} \rangle}_{f(\mathbf{d})}$$



Convex Function - Common Definition

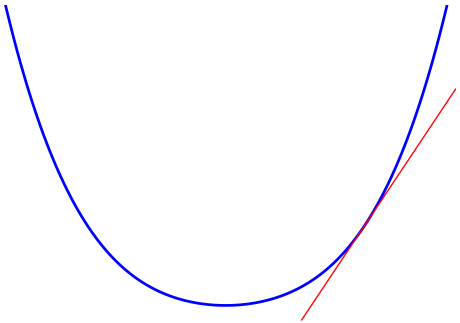


A function f is convex if, and only if, for all x, y and $\alpha \in (0,1)$ we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

A Key Property of Convex Functions

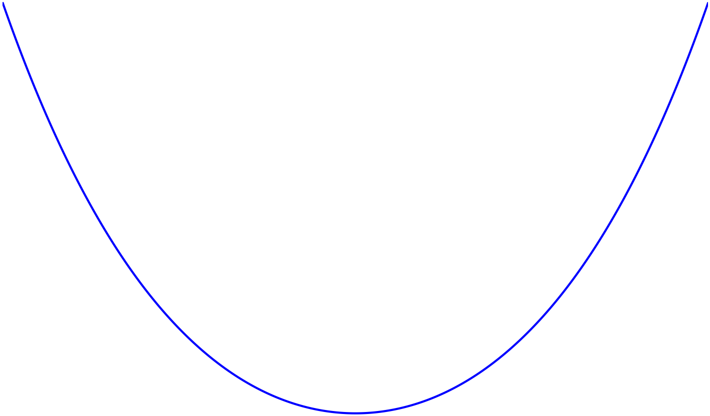
The First Order Taylor approximation always lower bounds the function



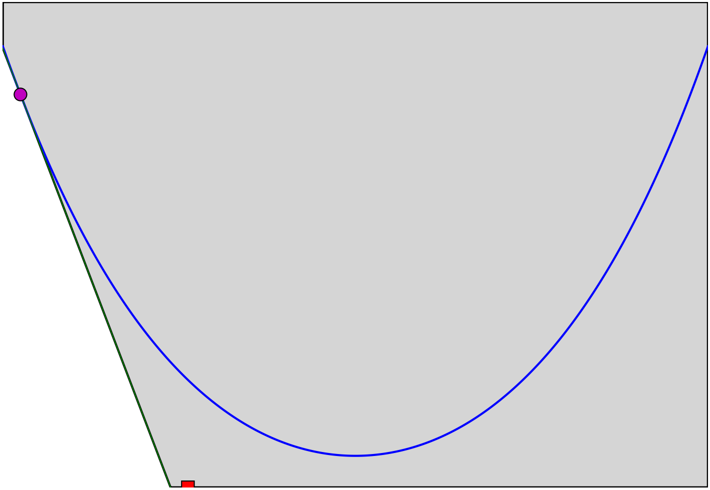
For any x and y we have

$$f(x) \geq f(y) + \langle x - y, \nabla f(y) \rangle$$

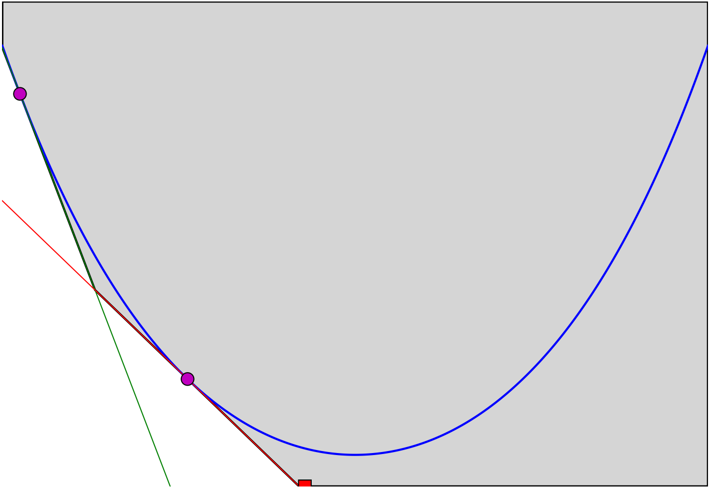
Cutting Plane Methods [Kelly 60]



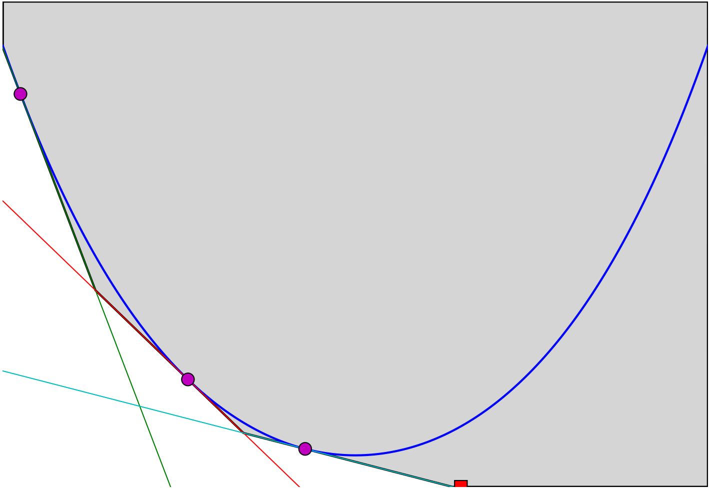
Cutting Plane Methods [Kelly 60]



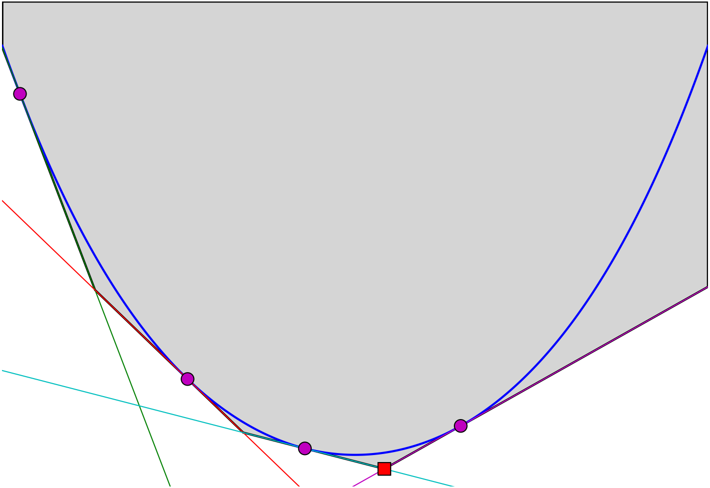
Cutting Plane Methods [Kelly 60]



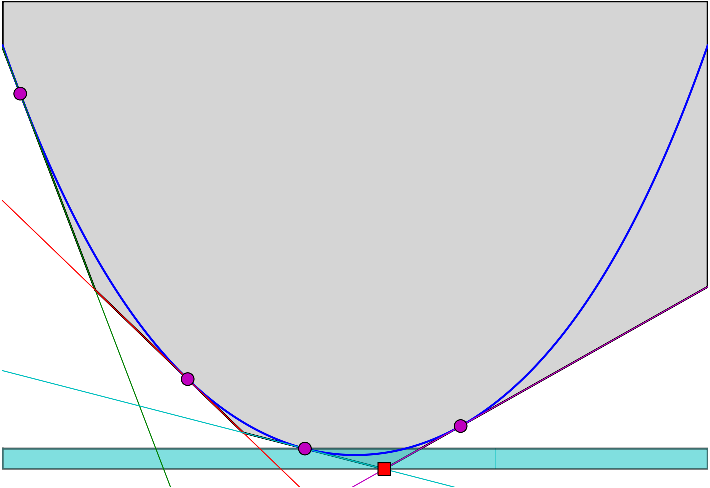
Cutting Plane Methods [Kelly 60]



Cutting Plane Methods [Kelly 60]



Monitoring Convergence



In a Nutshell

- Cutting Plane Methods work by forming the piecewise linear lower bound

$$f(x) \geq f_t^{\mathrm{CP}}(x) := \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

  where s_i denote gradients $\nabla f(x_{i-1})$.

- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_{x} f_t^{\mathrm{CP}}(x).$$

- Stop when the duality gap

$$\epsilon_t := \min_{0 \leq i \leq t} f(x_i) - f_t^{\mathrm{CP}}(x_t)$$

  falls below a pre-specified threshold ϵ.

In a Nutshell

- Cutting Plane Methods work by forming the piecewise linear lower bound

$$f(x) \geq f_t^{\mathrm{CP}}(x) := \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

  where s_i denote gradients $\nabla f(x_{i-1})$.
- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_x f_t^{\mathrm{CP}}(x).$$

- Stop when the duality gap

$$\epsilon_t := \min_{0 \leq i \leq t} f(x_i) - f_t^{\mathrm{CP}}(x_t)$$

  falls below a pre-specified threshold ϵ.

In a Nutshell

- Cutting Plane Methods work by forming the piecewise linear lower bound

$$f(x) \geq f_t^{\mathrm{CP}}(x) := \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

  where s_i denote gradients $\nabla f(x_{i-1})$.
- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

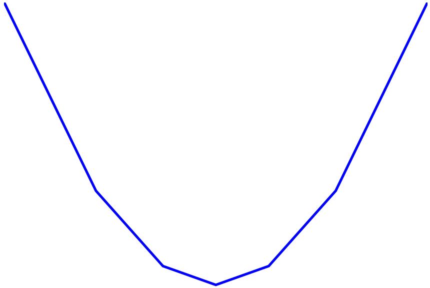
$$x_t := \operatorname*{argmin}_x f_t^{\mathrm{CP}}(x).$$

- Stop when the duality gap

$$\epsilon_t := \min_{0 \leq i \leq t} f(x_i) - f_t^{\mathrm{CP}}(x_t)$$

  falls below a pre-specified threshold ϵ.

What if the Function is NonSmooth?

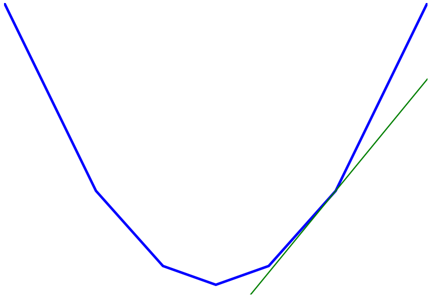


The piecewise linear function

$$f(x) := \max_i \langle u_i, x \rangle$$

is convex but not differentiable at the kinks!

Subgradients to the Rescue

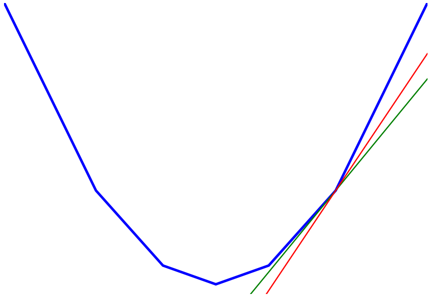


A subgradient at y is any vector μ which satisfies

$$f(x) \geq f(y) + \langle x - y, \mu \rangle \text{ for all } x$$

Set of all subgradients is denoted as $\partial f(y)$

Subgradients to the Rescue

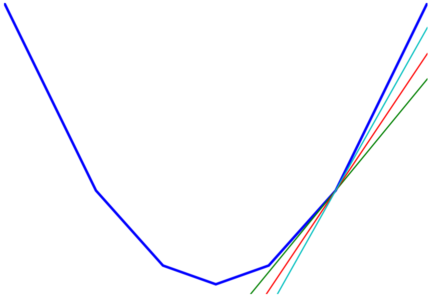


A subgradient at y is any vector μ which satisfies

$$f(x) \geq f(y) + \langle x - y, \mu \rangle \text{ for all } x$$

Set of all subgradients is denoted as $\partial f(y)$

Subgradients to the Rescue



A subgradient at y is any vector μ which satisfies

$$f(x) \geq f(y) + \langle x - y, \mu \rangle \text{ for all } x$$

Set of all subgradients is denoted as $\partial f(y)$

Good News!

Cutting Plane Methods work with subgradients

Just choose an arbitrary one

Boosting as an Optimization Problem

- Minimizing the maximum edge

$$\min_{\mathbf{d}\in\mathcal{S}^N} \underbrace{\max_{\mathbf{u}\in\mathcal{U}} \langle \mathbf{u}, \mathbf{d}\rangle}_{f(\mathbf{d})}$$

  is a convex optimization problem.

- Subgradient: $\partial f(\mathbf{d}) = \mathrm{argmax}_{\mathbf{u}\in\mathcal{U}} \langle \mathbf{u}, \mathbf{d}\rangle$

Computing Subgradient of $f(\mathbf{d})$ = Strong Oracle!

Boosting as an Optimization Problem

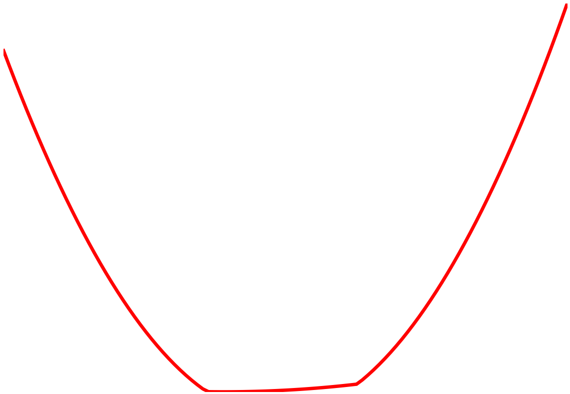
- Minimizing the maximum edge

$$\min_{\mathbf{d} \in \mathcal{S}^N} \underbrace{\max_{\mathbf{u} \in \mathcal{U}} \langle \mathbf{u}, \mathbf{d} \rangle}_{f(\mathbf{d})}$$

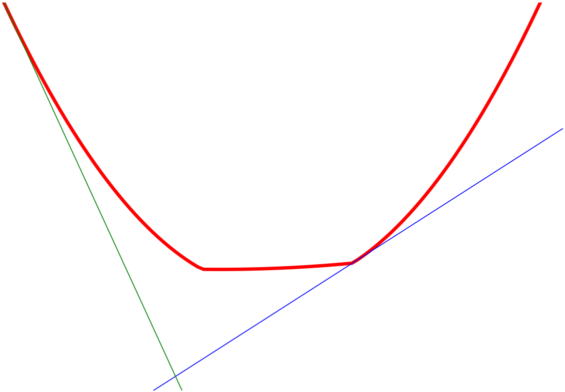
  is a convex optimization problem.
- Subgradient: $\partial f(\mathbf{d}) = \mathrm{argmax}_{\mathbf{u} \in \mathcal{U}} \langle \mathbf{u}, \mathbf{d} \rangle$

Computing Subgradient of $f(\mathbf{d})$ = Strong Oracle!

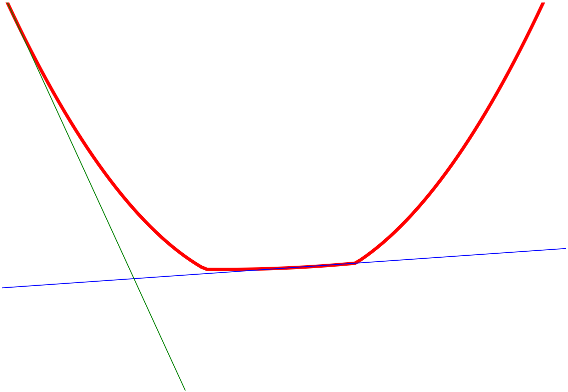
Subgradients and Stability



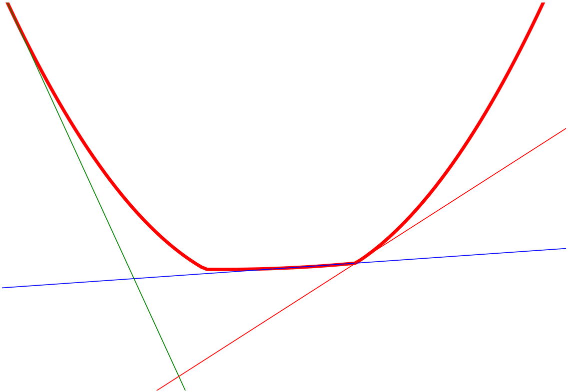
Subgradients and Stability



Subgradients and Stability

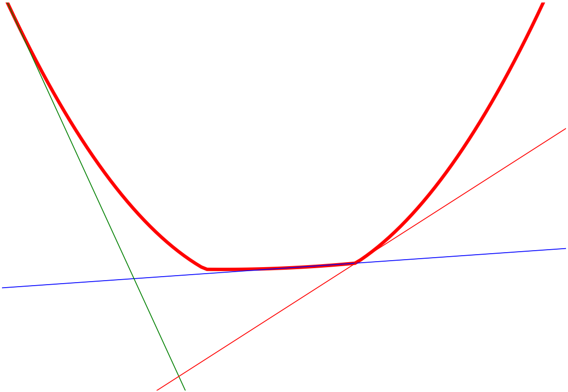


Subgradients and Stability



- Picking an arbitrary subgradient can cause stability issues

Subgradients and Stability



- Picking an arbitrary subgradient can cause stability issues

Back to Convex Analysis

Strong Convexity

A convex function f is strongly convex if, and only if, there exists a constant $\sigma > 0$ such that the function $f(x) - \frac{\sigma}{2}\|x\|^2$ is convex.

- If f is twice differentiable then the eigenvalues of the Hessian of f are lower bounded

$$\nabla^2 f(x) \succeq \sigma I.$$

- If f is strongly convex then

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle + \frac{\sigma}{2}\|x' - x\|^2 \quad \forall x, x' \text{ and } \mu \in \partial f(x).$$

Back to Convex Analysis

Strong Convexity

A convex function f is strongly convex if, and only if, there exists a constant $\sigma > 0$ such that the function $f(x) - \frac{\sigma}{2}\|x\|^2$ is convex.

- If f is twice differentiable then the eigenvalues of the Hessian of f are lower bounded

$$\nabla^2 f(x) \succeq \sigma I.$$

- If f is strongly convex then

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle + \frac{\sigma}{2}\|x' - x\|^2 \quad \forall x, x' \text{ and } \mu \in \partial f(x).$$

Back to Convex Analysis

Strong Convexity

A convex function f is strongly convex if, and only if, there exists a constant $\sigma > 0$ such that the function $f(x) - \frac{\sigma}{2}\|x\|^2$ is convex.

- If f is twice differentiable then the eigenvalues of the Hessian of f are lower bounded

$$\nabla^2 f(x) \succeq \sigma I.$$

- If f is strongly convex then

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle + \frac{\sigma}{2}\|x' - x\|^2 \quad \forall x, x' \text{ and } \mu \in \partial f(x).$$

Bundle Methods [HL93]

- Are stabilized cutting plane methods
- At every iteration they form the model

$$f_t(x) := \Omega(x) + \max_{1 \le i \le t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

  where Ω is an appropriately chosen strongly convex function, and s_i denotes a (sub)gradient of f evaluated at x_{i-1}.
- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_x f_t(x).$$

- Stop when

$$\epsilon_t := \min_{0 \le i \le t} \Omega(x_i) + f(x_i) - f_t(x_t)$$

  falls below a pre-specified threshold ϵ.

Bundle Methods [HL93]

- Are stabilized cutting plane methods
- At every iteration they form the model

$$f_t(x) := \Omega(x) + \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

  where Ω is an appropriately chosen strongly convex function, and s_i denotes a (sub)gradient of f evaluated at x_{i-1}.
- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_x f_t(x).$$

- Stop when

$$\epsilon_t := \min_{0 \leq i \leq t} \Omega(x_i) + f(x_i) - f_t(x_t)$$

  falls below a pre-specified threshold ϵ.

Bundle Methods [HL93]

- Are stabilized cutting plane methods
- At every iteration they form the model

$$f_t(x) := \Omega(x) + \max_{1 \le i \le t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}.$$

where Ω is an appropriately chosen strongly convex function, and s_i denotes a (sub)gradient of f evaluated at x_{i-1}.

- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_x f_t(x).$$

- Stop when

$$\epsilon_t := \min_{0 \le i \le t} \Omega(x_i) + f(x_i) - f_t(x_t)$$

falls below a pre-specified threshold ϵ.

Bundle Methods [HL93]

- Are stabilized cutting plane methods
- At every iteration they form the model

$$f_t(x) := \Omega(x) + \max_{1 \le i \le t} \{f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle\}.$$

where Ω is an appropriately chosen strongly convex function, and s_i denotes a (sub)gradient of f evaluated at x_{i-1}.

- At iteration t the set $\{x_i\}_{i=0}^{t-1}$ is augmented by

$$x_t := \operatorname*{argmin}_x f_t(x).$$

- Stop when

$$\epsilon_t := \min_{0 \le i \le t} \Omega(x_i) + f(x_i) - f_t(x_t)$$

falls below a pre-specified threshold ϵ.

Rates of Convergence [SVL08]

Nonsmooth Functions

- The number of iterations to reach ϵ precision is bounded by

$$t \leq \frac{c_1}{\epsilon \cdot \sigma} + c_2$$

where c_1 and c_2 are problem dependent constants, and σ is the modulus of strong convexity of Ω.

Proving Iteration Bounds for Boosting

Lemma

Suppose $0 \leq \Omega(\mathbf{d}) \leq \epsilon/2$ *for all* $\mathbf{d}$*, and let*

$$\min_{0 \leq i \leq t} \Omega(\mathbf{d}_i) + f(\mathbf{d}_i) - f_t(\mathbf{d}_t) \leq \epsilon/2.$$

Then

$$f(\mathbf{d}_t) \leq f(\mathbf{d}^*) + \epsilon$$

Proving Iteration Bounds Contd.

Entropy as a Regularizer

Suppose we set $\Omega(\mathbf{d}) = \frac{\epsilon}{2\log N}\sum_{i=1}^{N} d_i \log d_i$ then

- $\Omega(\mathbf{d}) \leq \epsilon/2$ for all $\mathbf{d}$
- Ω is strongly convex with modulus $\frac{\epsilon}{2\log N}$.

- Plugging this into rates of convergence of bundle methods shows that we need

$$t \leq 2\frac{c_1 \log N}{\epsilon^2} + c_2$$

iterations to obtain an ϵ accurate solution [SSS08].

- Can also prove similar iteration bounds for an oracle which instead of returning $\operatorname{argmax}_{\mathbf{u}\in\mathcal{U}} \langle \mathbf{u}, \mathbf{d}\rangle$ returns an $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{d}\rangle \geq g$ [WGV08].

Proving Iteration Bounds Contd.

Entropy as a Regularizer

Suppose we set $\Omega(\mathbf{d}) = \frac{\epsilon}{2 \log N} \sum_{i=1}^{N} d_i \log d_i$ then

- $\Omega(\mathbf{d}) \leq \epsilon/2$ for all $\mathbf{d}$
- Ω is strongly convex with modulus $\frac{\epsilon}{2 \log N}$.

- Plugging this into rates of convergence of bundle methods shows that we need

$$t \leq 2\frac{c_1 \log N}{\epsilon^2} + c_2$$

iterations to obtain an ϵ accurate solution [SSS08].

- Can also prove similar iteration bounds for an oracle which instead of returning $\operatorname{argmax}_{\mathbf{u} \in \mathcal{U}} \langle \mathbf{u}, \mathbf{d} \rangle$ returns an $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{d} \rangle \geq g$ [WGV08].

Proving Iteration Bounds Contd.

Entropy as a Regularizer

Suppose we set $\Omega(\mathbf{d}) = \frac{\epsilon}{2\log N}\sum_{i=1}^{N} d_i \log d_i$ then

- $\Omega(\mathbf{d}) \leq \epsilon/2$ for all $\mathbf{d}$
- Ω is strongly convex with modulus $\frac{\epsilon}{2\log N}$.

- Plugging this into rates of convergence of bundle methods shows that we need

$$t \leq 2\frac{c_1 \log N}{\epsilon^2} + c_2$$

  iterations to obtain an ϵ accurate solution [SSS08].
- Can also prove similar iteration bounds for an oracle which instead of returning $\mathrm{argmax}_{\mathbf{u}\in\mathcal{U}} \langle \mathbf{u}, \mathbf{d}\rangle$ returns an $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{d}\rangle \geq g$ [WGV08].

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

$\Omega(1/\epsilon^2)$ Iteration Bounds [NY78,BMN01]

Setting

- Let B be any black box boosting algorithm
- $\mathbf{d}^1, \ldots, \mathbf{d}^{t-1}$ are intermediate distributions
- $\mathbf{d}^t$ is the distribution produced by B after t iterations

The Adversary

- Produces a set of hypothesis $\mathbf{u}_1, \ldots, \mathbf{u}_t$ which defines a function

$$f(\mathbf{d}) := \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d} \rangle$$

- We will show that $f(\mathbf{d}^t) - \min_{\mathbf{d}} f(\mathbf{d}) \geq \frac{1}{\sqrt{t}}$
- To get ϵ accuracy B needs $O(1/\epsilon^2)$ iterations.

The Resisting Oracle

Hadamard Matrix

The Hadamard matrix is defined recursively:

$$\mathbf{H}_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad \mathbf{H}_{2N} = \begin{pmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{pmatrix}$$

Hypothesis Space

- The oracle chooses $\mathbf{u}^q$ from the columns of the following matrix:

$$\mathbf{U} = \begin{pmatrix} \mathbf{H}_{N/2} & -\mathbf{H}_{N/2} \\ -\mathbf{H}_{N/2} & \mathbf{H}_{N/2} \end{pmatrix}.$$

- Because of symmetry, for any $t < N/2$ we have

$$f(\mathbf{d}^t) = \max_{\mathbf{u} \in \mathbf{U}} \langle \mathbf{u}, \mathbf{d}^t \rangle \geq 0.$$

The Resisting Oracle

Hadamard Matrix

The Hadamard matrix is defined recursively:

$$\mathbf{H}_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad \mathbf{H}_{2N} = \begin{pmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{pmatrix}$$

Hypothesis Space

- The oracle chooses $\mathbf{u}^q$ from the columns of the following matrix:

$$\mathbf{U} = \begin{pmatrix} \mathbf{H}_{N/2} & -\mathbf{H}_{N/2} \\ -\mathbf{H}_{N/2} & \mathbf{H}_{N/2} \end{pmatrix}.$$

- Because of symmetry, for any $t < N/2$ we have

$$f(\mathbf{d}^t) = \max_{\mathbf{u} \in \mathbf{U}} \langle \mathbf{u}, \mathbf{d}^t \rangle \geq 0.$$

The Resisting Oracle

Hadamard Matrix

The Hadamard matrix is defined recursively:

$$\mathbf{H}_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad \mathbf{H}_{2N} = \begin{pmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{pmatrix}$$

Hypothesis Space

- The oracle chooses $\mathbf{u}^q$ from the columns of the following matrix:

$$\mathbf{U} = \begin{pmatrix} \mathbf{H}_{N/2} & -\mathbf{H}_{N/2} \\ -\mathbf{H}_{N/2} & \mathbf{H}_{N/2} \end{pmatrix}.$$

- Because of symmetry, for any $t < N/2$ we have

$$f(\mathbf{d}^t) = \max_{\mathbf{u} \in \mathbf{U}} \langle \mathbf{u}, \mathbf{d}^t \rangle \geq 0.$$

The Resisting Oracle Contd.

Upper Bound on $\min_{\mathbf{d}} f(\mathbf{d})$

$$\begin{aligned}
\min_{\mathbf{d}\in\mathcal{S}^N} f(\mathbf{d}) &= \min_{\mathbf{d}\in\mathcal{S}^n}\ \max_{q=1,\ldots,t} \langle \mathbf{u}^q, \mathbf{d}\rangle \\
\text{(Mixed strategy)} &= \min_{\mathbf{d}\in\mathcal{S}^N} \max_{\mathbf{w}\in\mathcal{S}^t} \mathbf{d}^\top \mathbf{U}_t\, \mathbf{w} \\
\text{(Convexity)} &= \max_{\mathbf{w}\in\mathcal{S}^t} \min_{\mathbf{d}\in\mathcal{S}^N} \mathbf{d}^\top \mathbf{U}_t\, \mathbf{w} \\
\text{(Pure Strategy)} &= \max_{\mathbf{w}\in\mathcal{S}^t}\ \min_{j=1,\ldots,n} [\mathbf{U}_t\, \mathbf{w}]_j.
\end{aligned}$$

The Resisting Oracle Contd.

$\mathbf{U}_t$ has vertical Symmetry

Since columns of $\mathbf{U}_t$ are chosen from

$$U = \begin{pmatrix} \mathbf{H}_{n/2} & -\mathbf{H}_{n/2} \\ -\mathbf{H}_{n/2} & \mathbf{H}_{n/2} \end{pmatrix} \text{ we can write } \mathbf{U}_t = \begin{pmatrix} \widehat{\mathbf{U}}_t \\ -\widehat{\mathbf{U}}_t \end{pmatrix}.$$

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{S}^N} f(\mathbf{d}) &= \max_{\mathbf{w}\in\mathcal{S}^t} \min_{j=1,\ldots,n} [\mathbf{U}_t\,\mathbf{w}]_j \\ &= \max_{\mathbf{w}\in\mathcal{S}^t} - \left(\max_{j=1,\ldots,n} [\widehat{\mathbf{U}}_t\,\mathbf{w}]_j \right) \\ &= -\min_{\mathbf{w}\in\mathcal{S}^t} \|\widehat{\mathbf{U}}_t\,\mathbf{w}\|_\infty \end{aligned}$$

The Resisting Oracle Contd.

$$-\|\cdot\|_\infty \leq -\frac{1}{\sqrt{n}}\|\cdot\|_2$$

$$\begin{aligned}
\min_{\mathbf{d}\in\mathcal{S}^N} f(\mathbf{d}) &= -\min_{\mathbf{w}\in\mathcal{S}^t} \|\widehat{\mathbf{U}}_t\,\mathbf{w}\,\|_\infty \\
&\leq -\frac{1}{\sqrt{n/2}} \min_{\mathbf{w}\in S^t} \|\widehat{\mathbf{U}}_t\,\mathbf{w}\,\|_2 \\
&= -\frac{1}{\sqrt{n/2}} \min_{\mathbf{w}\in S^t} \sqrt{\mathbf{w}^\top \underbrace{\widehat{\mathbf{U}}_t^\top \widehat{\mathbf{U}}_t}_{n/2\cdot\mathbf{I}_t} \mathbf{w}} \\
&= -\min_{\mathbf{w}\in S^t} \sqrt{\mathbf{w}^\top\,\mathbf{w}} \\
&= -\frac{1}{\sqrt{t}}
\end{aligned}$$

The Optimization Problem

Primal Problem

$$\min_{\substack{\mathbf{d}\cdot\boldsymbol{I}=1\\ \mathbf{d}\le\frac{1}{\nu}\boldsymbol{I}}} \quad \underbrace{\frac{1}{\eta}\Delta(\mathbf{d},\mathbf{d}^0)+\max_{q=1,2,\ldots,t}\langle\mathbf{u}^q,\mathbf{d}\rangle}_{:=P^t(\mathbf{d})}.$$

Dual Problem

$$\max_{\substack{\mathbf{w}\ge\mathbf{0}\\ \langle\mathbf{w},\mathbf{e}\rangle=1\\ \psi\ge 0}} \quad -\frac{1}{\eta}\log\sum_{n=1}^{N} d_n^0\exp(-\eta\sum_{q=1}^{t}u_n^q w_q-\eta\psi_n)-\frac{1}{\nu}\sum_{n=1}^{N}\psi_n$$

The Optimization Problem

Primal Problem

$$\min_{\substack{\mathbf{d}\cdot\boldsymbol{I}=1\\ \mathbf{d}\le\frac{1}{\nu}\boldsymbol{I}}} \quad \underbrace{\frac{1}{\eta}\Delta(\mathbf{d},\mathbf{d}^0) + \max_{q=1,2,\ldots,t}\langle\mathbf{u}^q,\mathbf{d}\rangle}_{:=P^t(\mathbf{d})}.$$

Dual Problem

$$\max_{\substack{\mathbf{w}\ge\mathbf{0}\\ \langle\mathbf{w},\mathbf{e}\rangle=1\\ \boldsymbol{\psi}\ge 0}} \quad -\frac{1}{\eta}\log\sum_{n=1}^{N} d_n^0 \exp\Big(-\eta\sum_{q=1}^{t} u_n^q w_q - \eta\psi_n\Big) - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n$$

Gradient Descent

Unconstrained

- Suppose you want to minimize

$$\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w})$$

- Gradient descent produces a sequence of iterates

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

- The step-size η found by
  - Solving $\min_\eta f(\mathbf{w}_t - \eta\nabla f(\mathbf{w}_t))$ or
  - Decay schedule such as $\eta_t = \frac{1}{t}$

Gradient Descent

Unconstrained

- Suppose you want to minimize

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w})$$

- Gradient descent produces a sequence of iterates

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

- The step-size η found by
  - Solving $\min_\eta f(\mathbf{w}_t - \eta \nabla f(\mathbf{w}_t))$ or
  - Decay schedule such as $\eta_t = \frac{1}{t}$

Gradient Descent

Unconstrained

- Suppose you want to minimize

$$\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w})$$

- Gradient descent produces a sequence of iterates

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

- The step-size η found by
  - Solving $\min_\eta f(\mathbf{w}_t - \eta\nabla f(\mathbf{w}_t))$ or
  - Decay schedule such as $\eta_t = \frac{1}{t}$

Projected Gradient Descent

Constrained

- Suppose you want to minimize

$$\min_{\mathbf{w}\in\Gamma} f(\mathbf{w})$$

- Projected Gradient descent produces a sequence of iterates

$$\mathbf{w}_{t+1} = P_\Gamma(\mathbf{w}_t - \eta \nabla f(\mathbf{w}))$$

where $P_\Gamma(\mathbf{z}) := \mathrm{argmin}_{x\in\Omega} \|\mathbf{z} - x\|^2$.

Projected Gradient Descent

Constrained

- Suppose you want to minimize

$$\min_{\mathbf{w}\in\Gamma} f(\mathbf{w})$$

- Projected Gradient descent produces a sequence of iterates

$$\mathbf{w}_{t+1} = P_\Gamma(\mathbf{w}_t - \eta\nabla f(\mathbf{w}))$$

where $P_\Gamma(\mathbf{z}) := \mathrm{argmin}_{x\in\Omega} \| \mathbf{z} - x\|^2$.

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t / b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w})\rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1\lambda, \sigma_2\lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$ $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1\lambda, \sigma_2\lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1\lambda, \sigma_2\lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \le \max_{0 \le j \le M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$ $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1\lambda, \sigma_2\lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \le 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t / b_t\}\}$

Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w}) \rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1\lambda, \sigma_2\lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t/b_t\}\}$

Spectral Projected Gradient Method [BMR00]

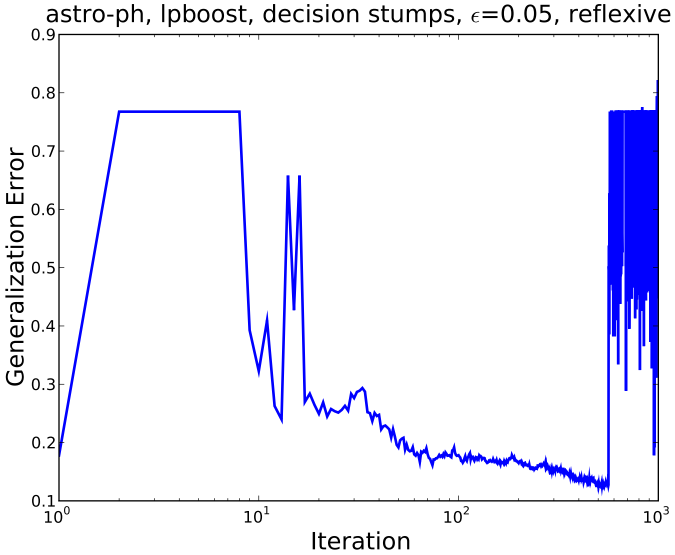
- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $\mathbf{d}_t = P_\Omega(\mathbf{w}_t - \alpha_t \nabla f(\mathbf{w})) - \mathbf{w}_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $\mathbf{w}_+ = \mathbf{w}_t + \lambda\, \mathbf{d}_t$
  - **Step 2.3:** If $f(\mathbf{w}_+) \leq \max_{0 \leq j \leq M} f(\mathbf{w}_{t-j}) + \gamma\lambda \langle \mathbf{d}_t, \nabla f(\mathbf{w})\rangle$
    $\lambda_t = \lambda$, $\mathbf{w}_{t+1} = \mathbf{w}_+$, $\mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t$, $\mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t)$
  - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle \mathbf{s}_t, \mathbf{y}_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\max}$
  - Else compute $a_t = \langle \mathbf{y}_t, \mathbf{y}_t \rangle$ and set
    $\alpha_{t+1} = \min\{\alpha_{\max}, \max\{\alpha_{\min}, a_t / b_t\}\}$

Dataset Properties

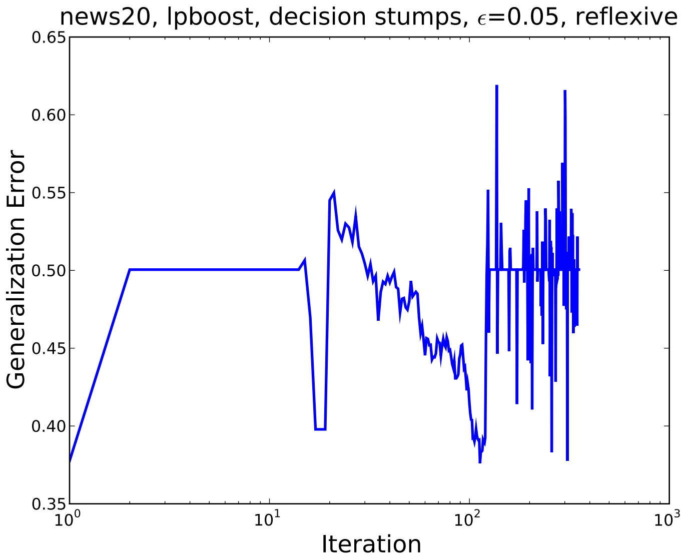
Name	Size	Dimension	Density
Astro-ph	94,856	99,757	0.008
News20	19,954	1,355,191	0.03
Real-Sim	72,201	20,958	0.25
rcv1	677,399	47,236	0.15

- Combined test and train. 60% randomly chosen for train, 20% for test and 20% for validation.
- Plot for generating the splits available.

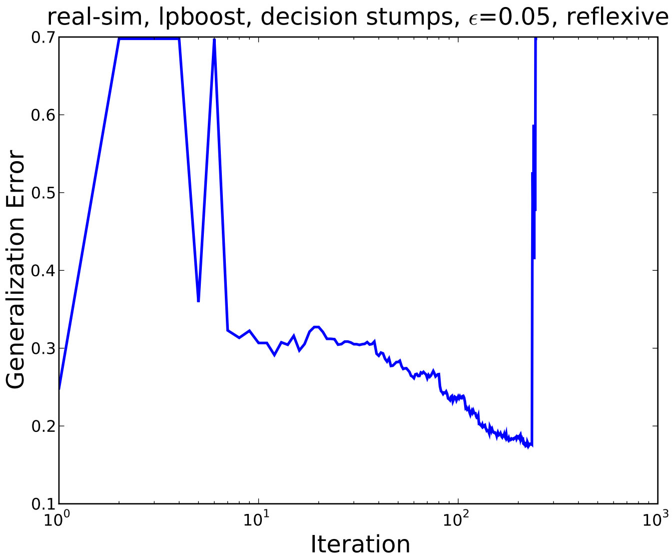
LPBoost is Brittle



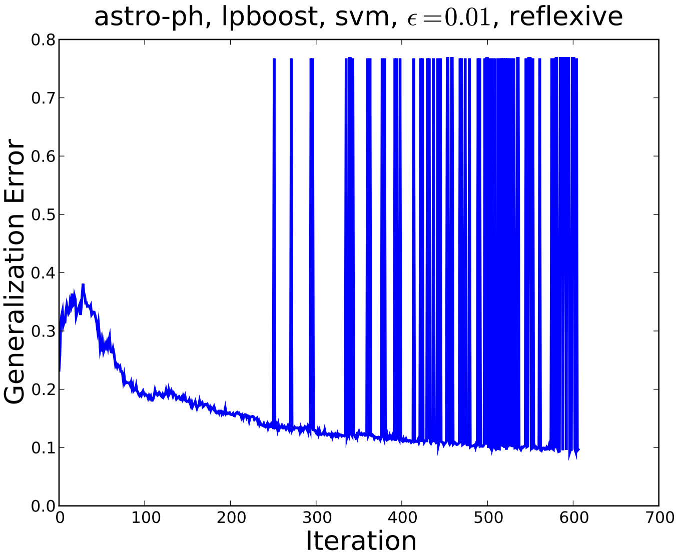
LPBoost is Brittle



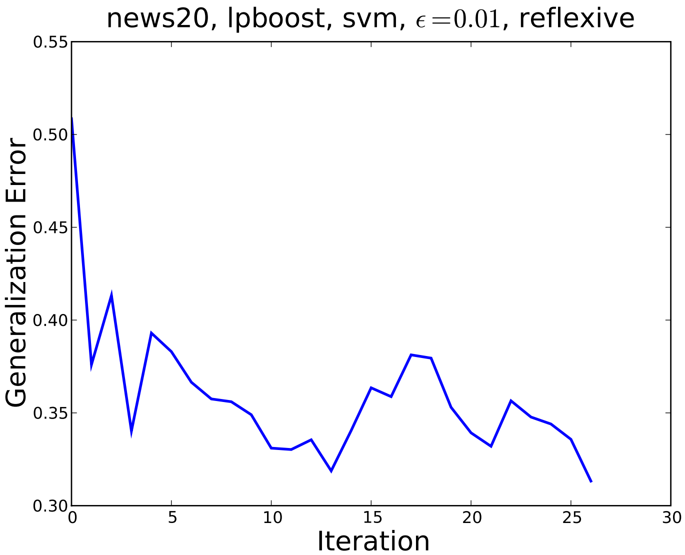
LPBoost is Brittle



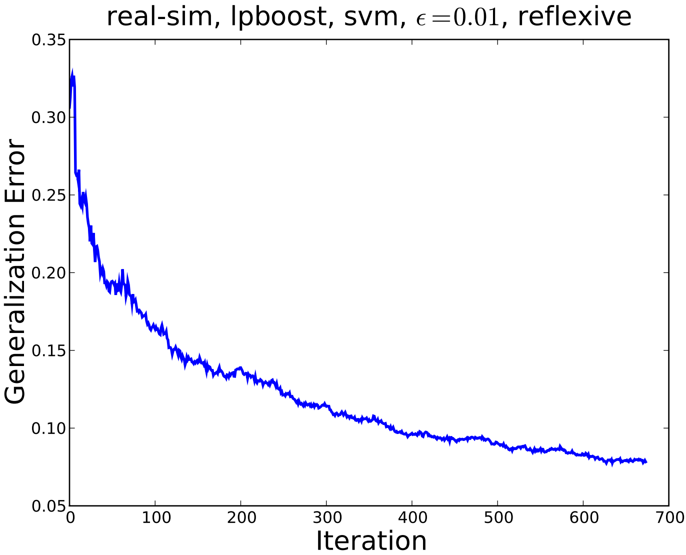
LPBoost is Brittle



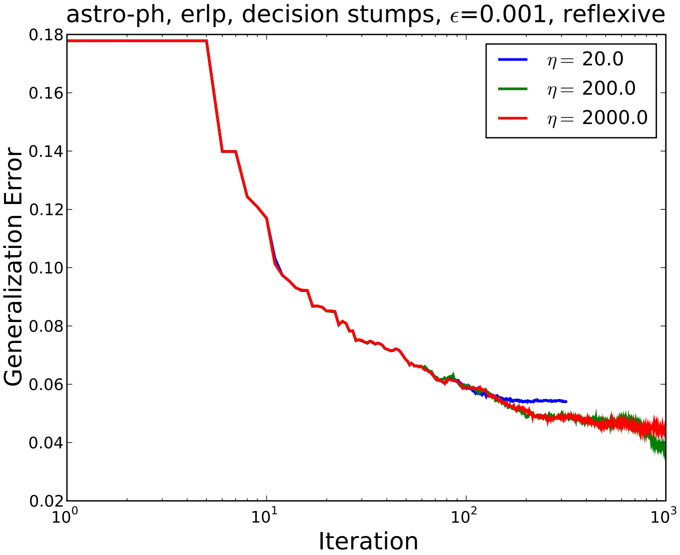
LPBoost is Brittle



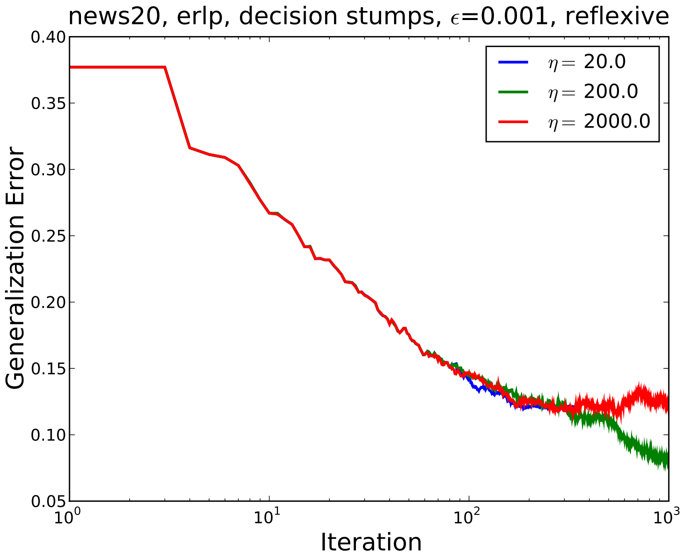
LPBoost is Brittle



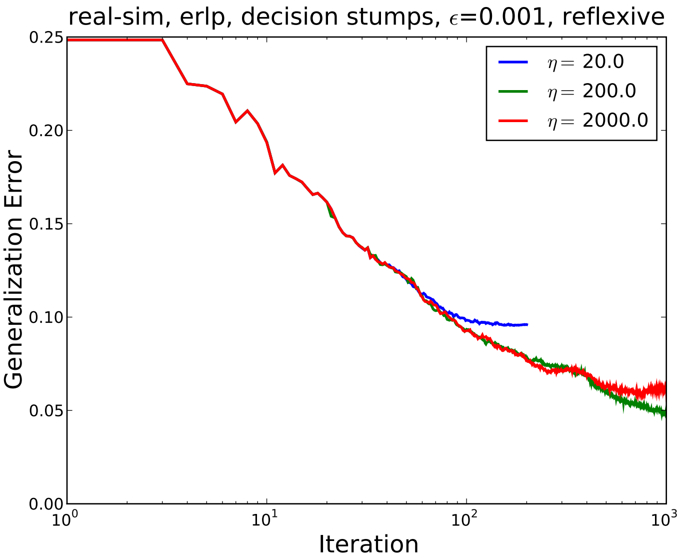
ERLPBoost fixes the problem



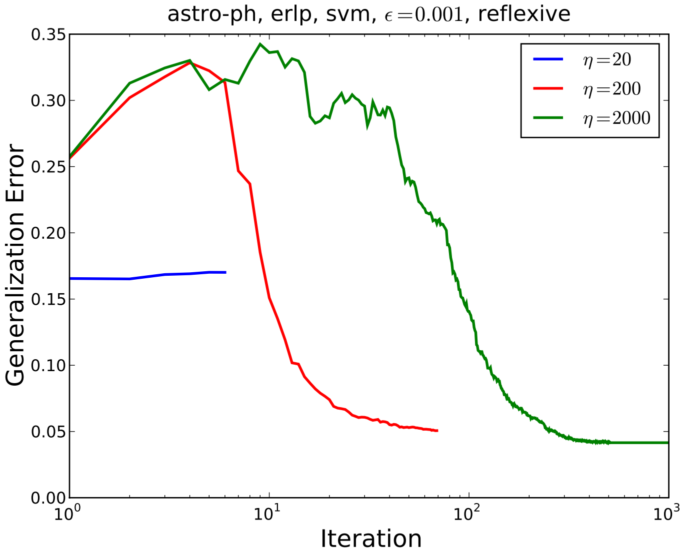
ERLPBoost fixes the problem



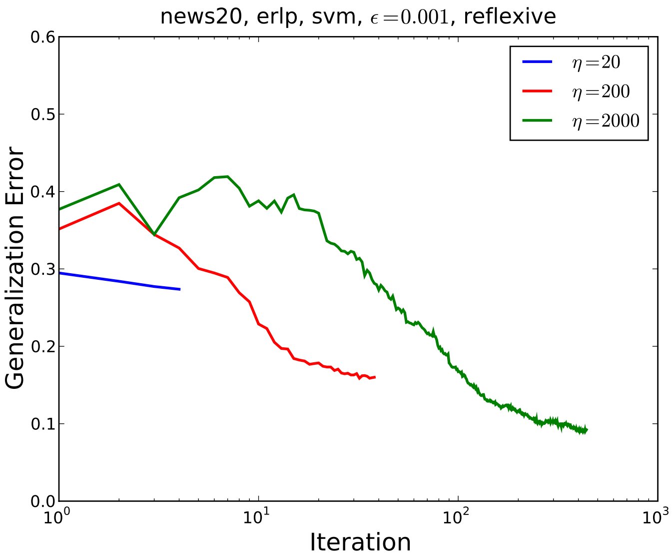
ERLPBoost fixes the problem



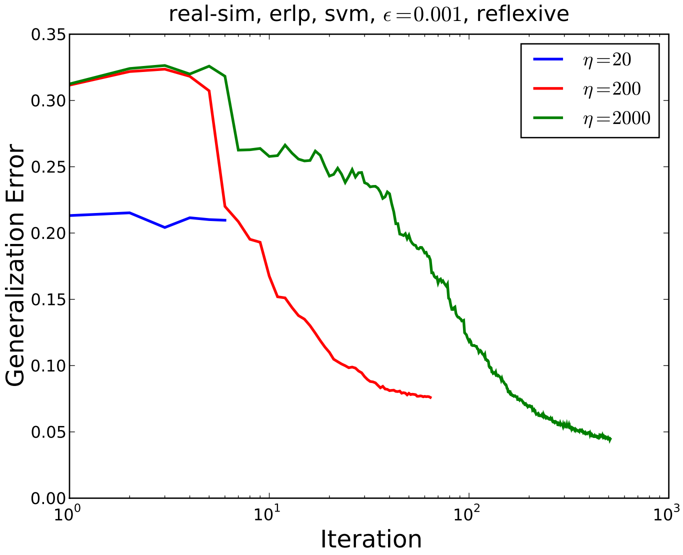
ERLPBoost fixes the problem



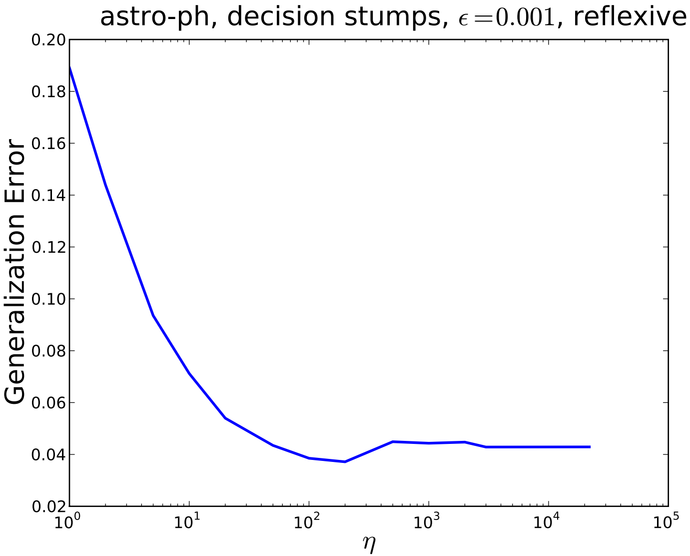
ERLPBoost fixes the problem



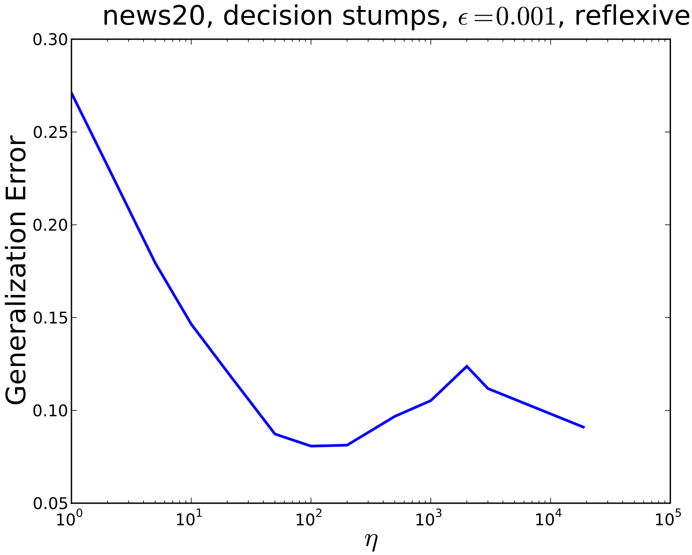
ERLPBoost fixes the problem



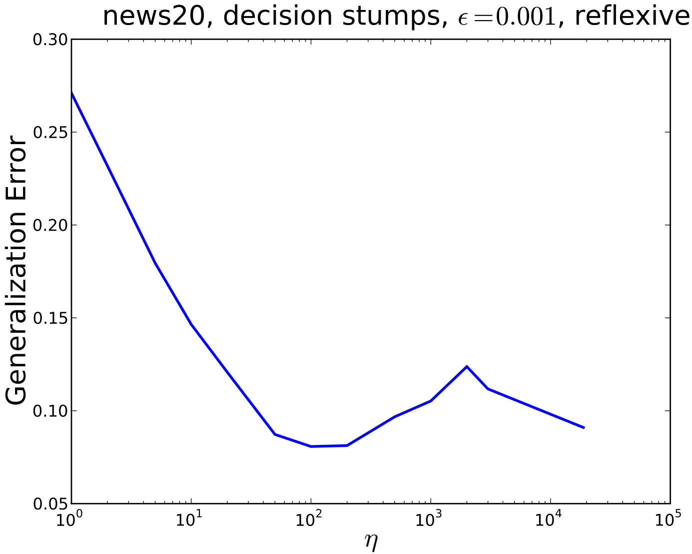
Generalization as a function of η



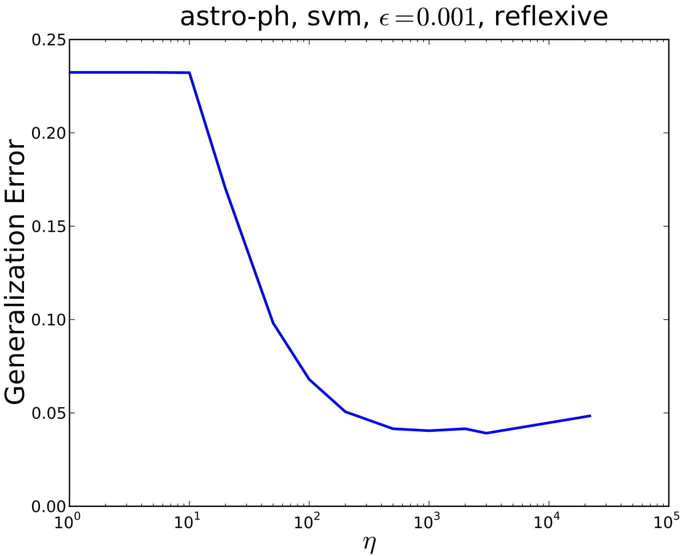
Generalization as a function of η



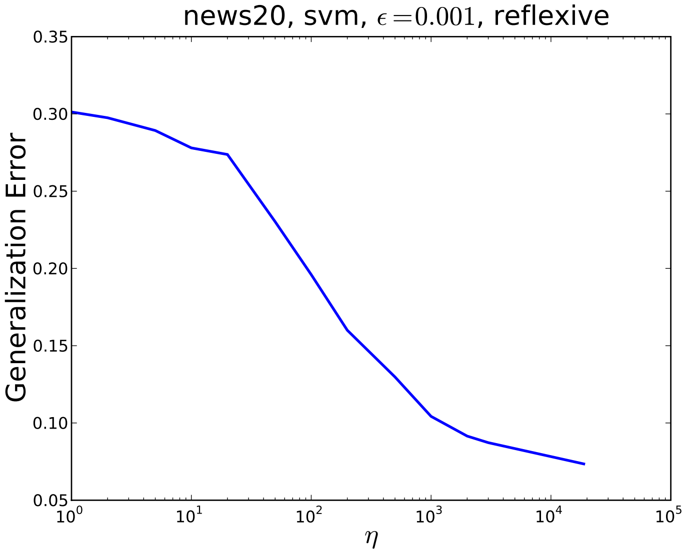
Generalization as a function of η



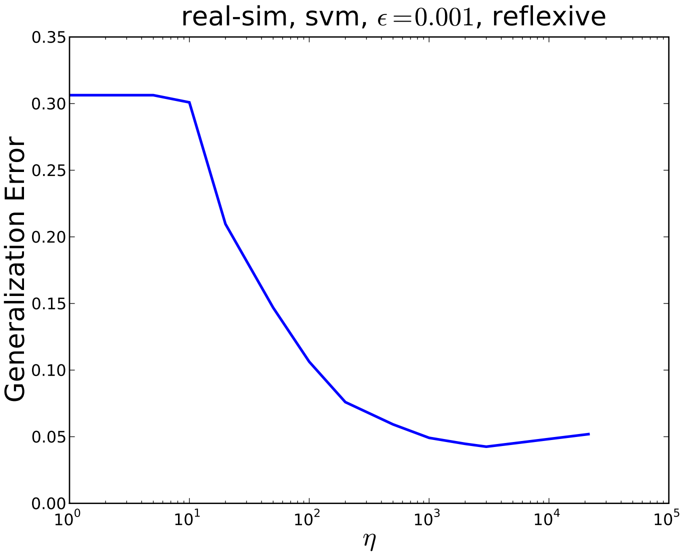
Generalization as a function of η



Generalization as a function of η

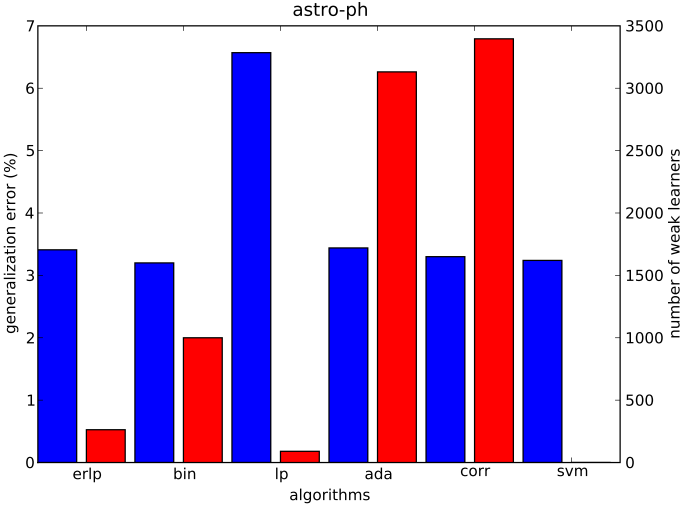


Generalization as a function of η



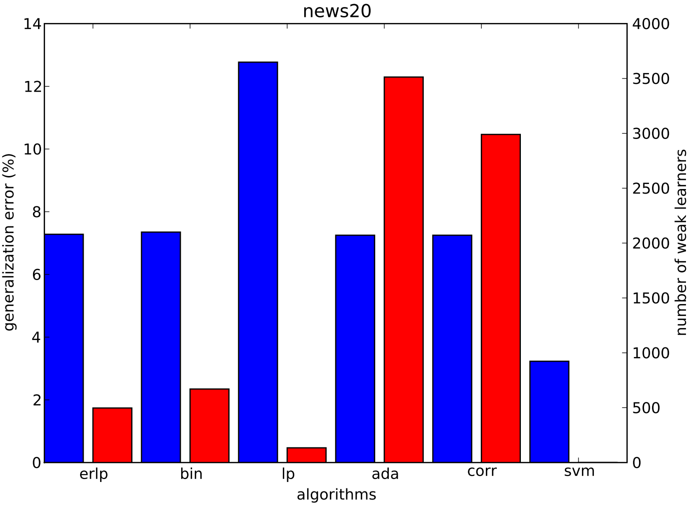
Generalization Error and # of Weak Hypothesis

Parameters tuned for best generalization performance



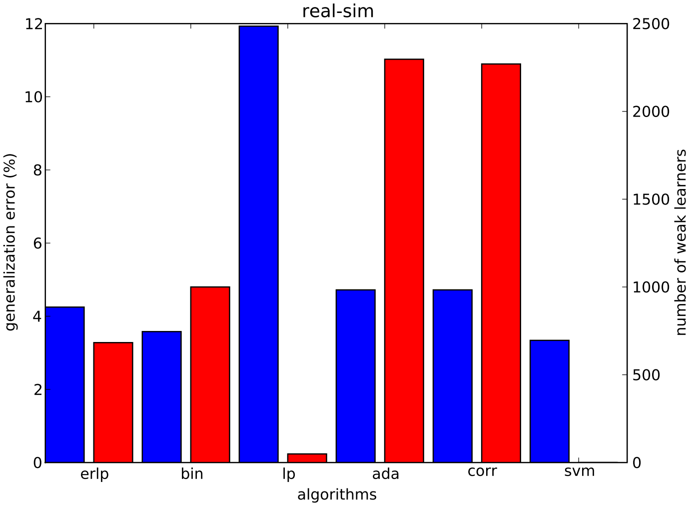
Generalization Error and # of Weak Hypothesis

Parameters tuned for best generalization performance

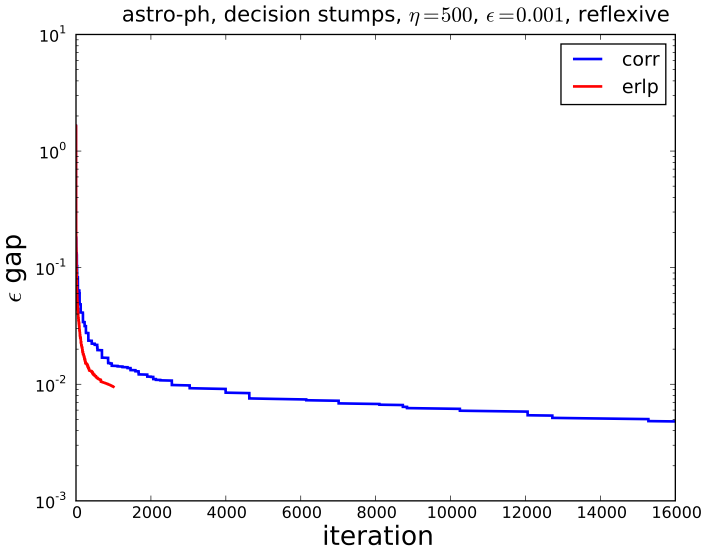


Generalization Error and # of Weak Hypothesis

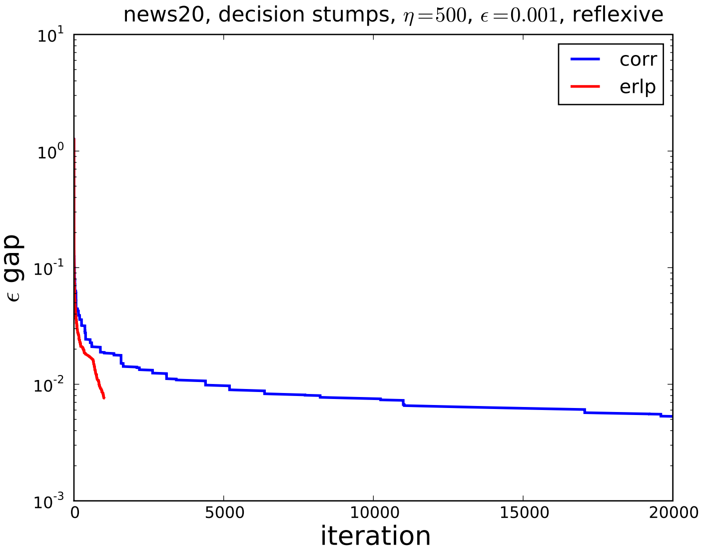
Parameters tuned for best generalization performance



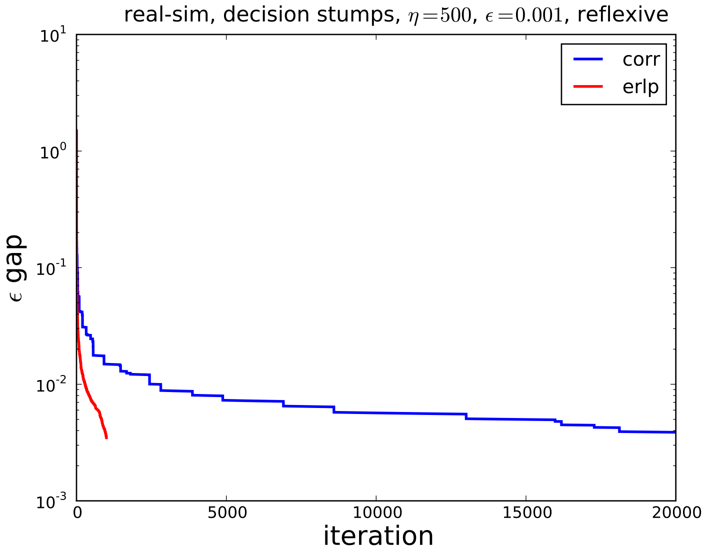
Corrective vs Totally Corrective



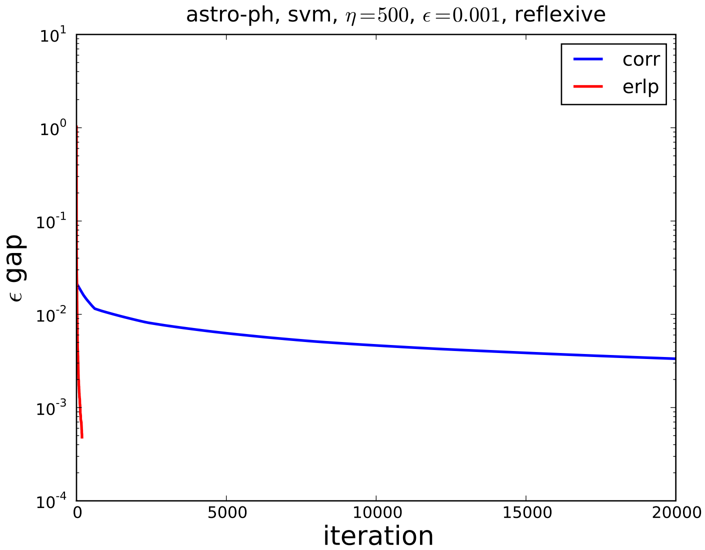
Corrective vs Totally Corrective



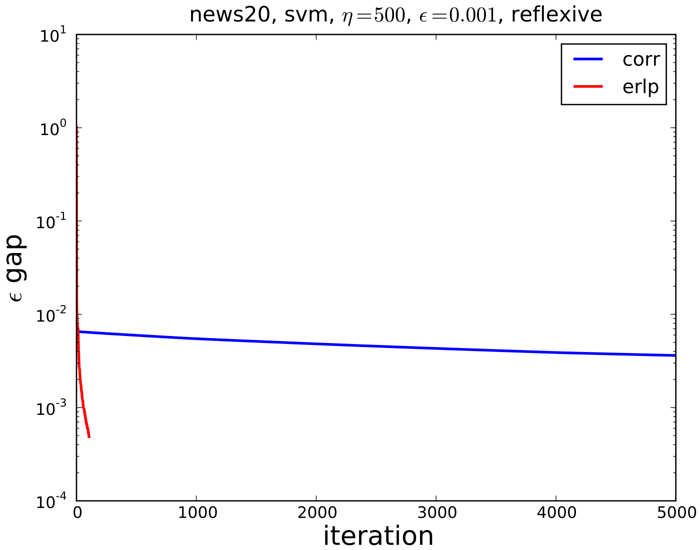
Corrective vs Totally Corrective



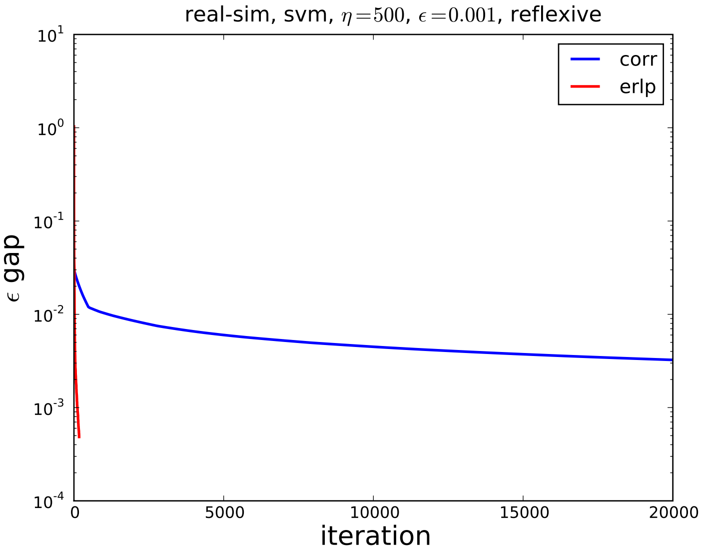
Corrective vs Totally Corrective



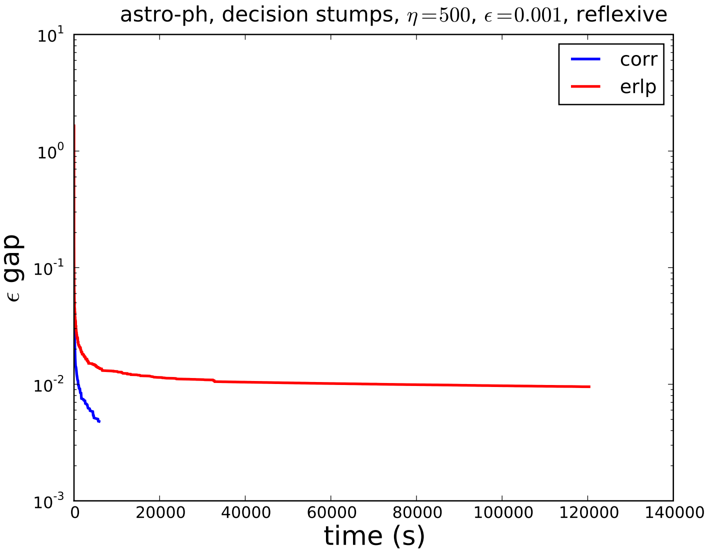
Corrective vs Totally Corrective



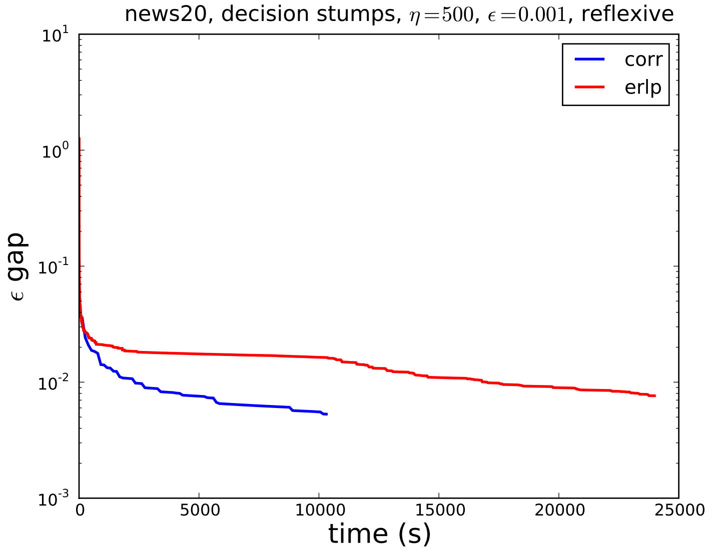
Corrective vs Totally Corrective



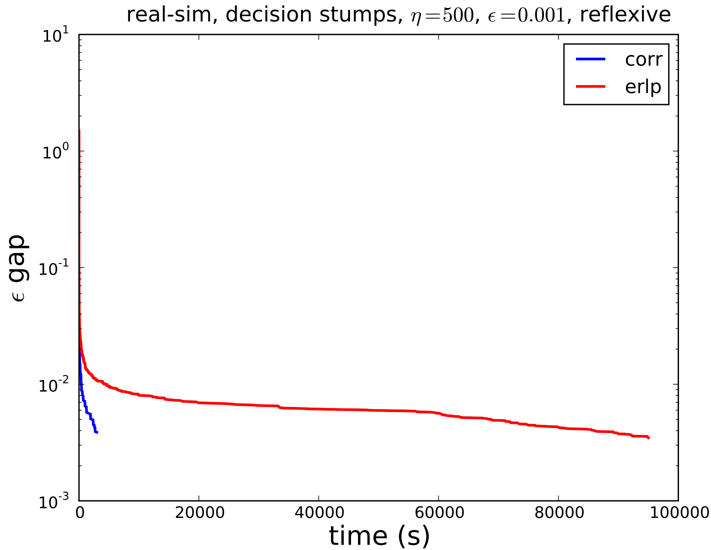
Corrective vs Totally Corrective contd.



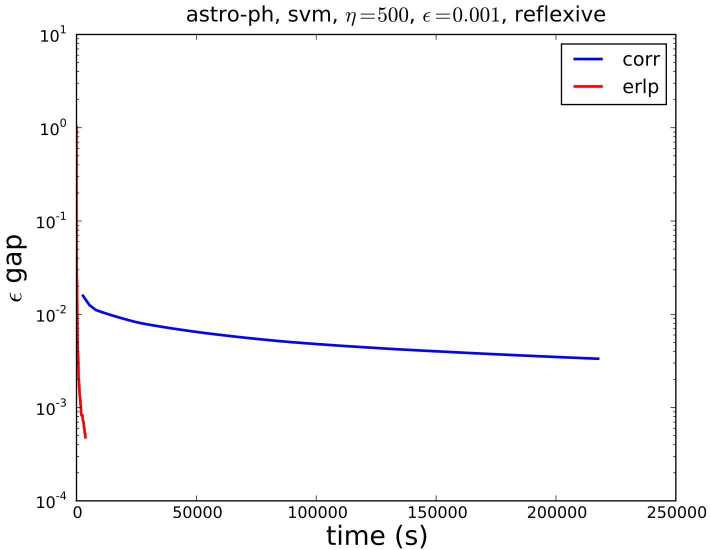
Corrective vs Totally Corrective contd.



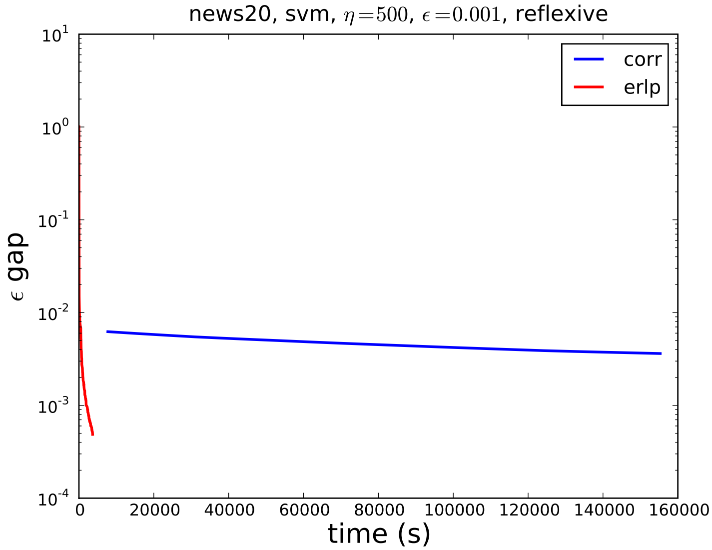
Corrective vs Totally Corrective contd.



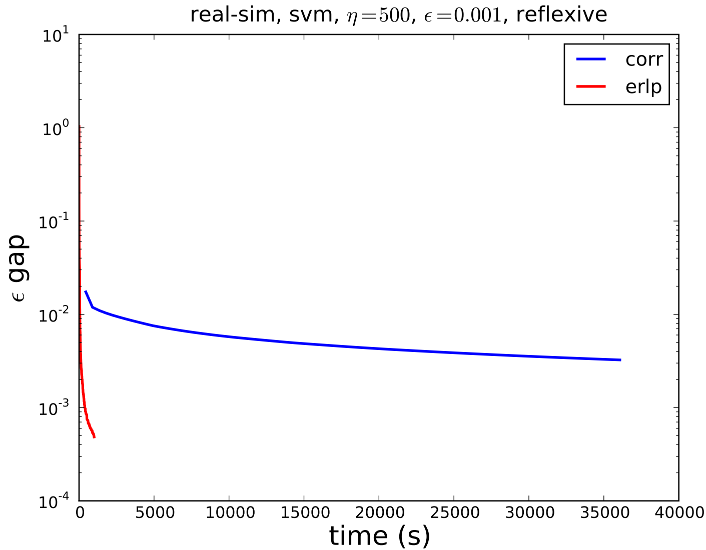
Corrective vs Totally Corrective contd.



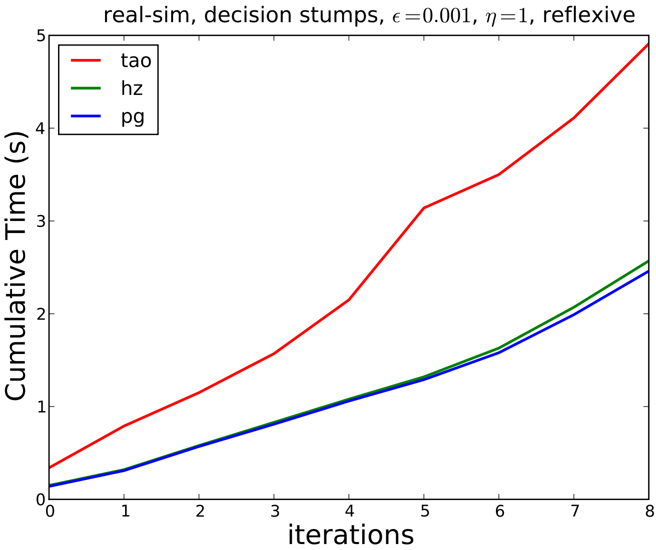
Corrective vs Totally Corrective contd.



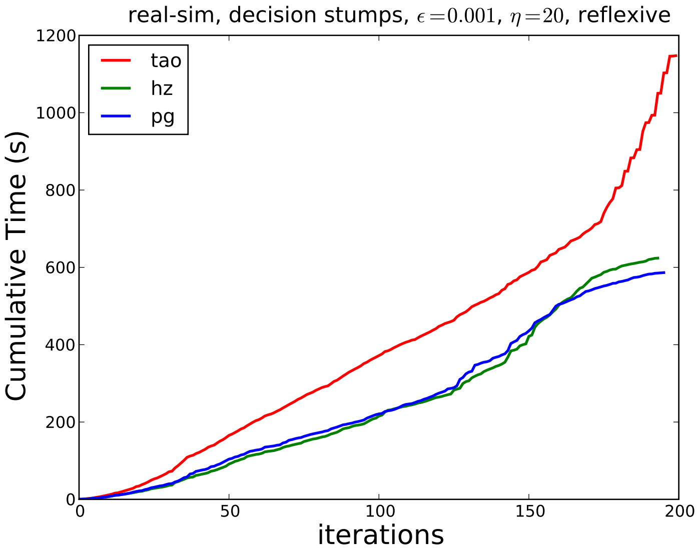
Corrective vs Totally Corrective contd.



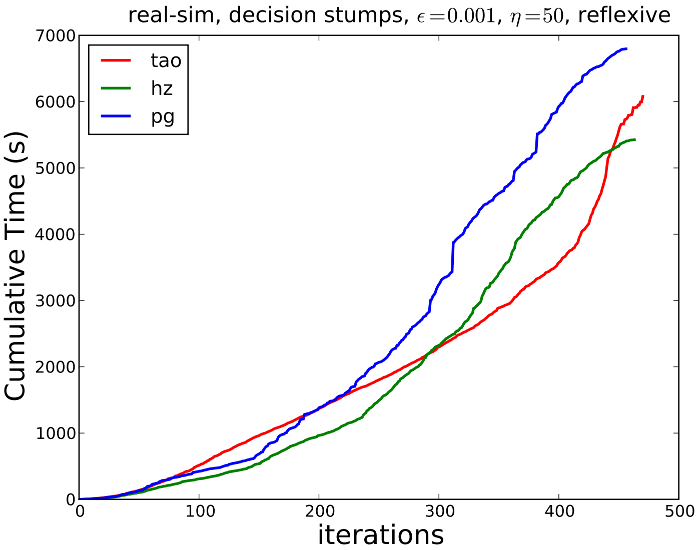
Comparing Different Optimizers



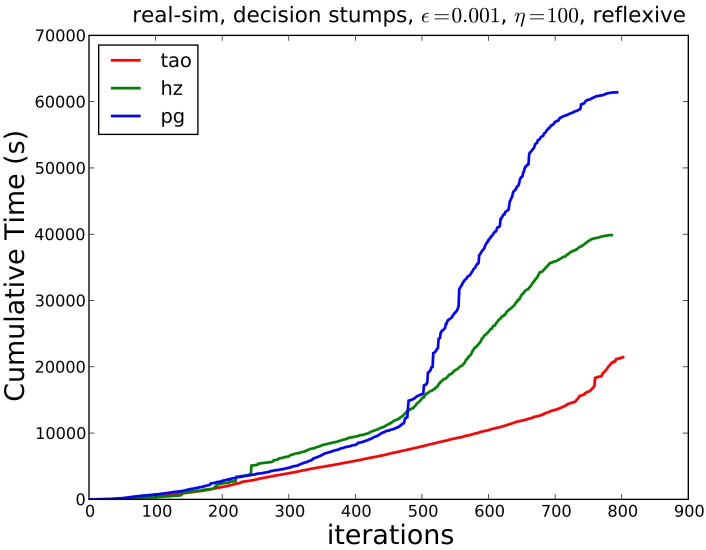
Comparing Different Optimizers



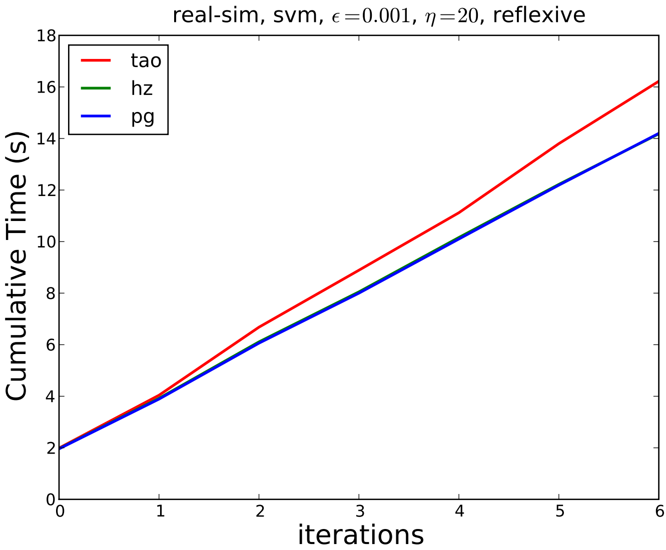
Comparing Different Optimizers



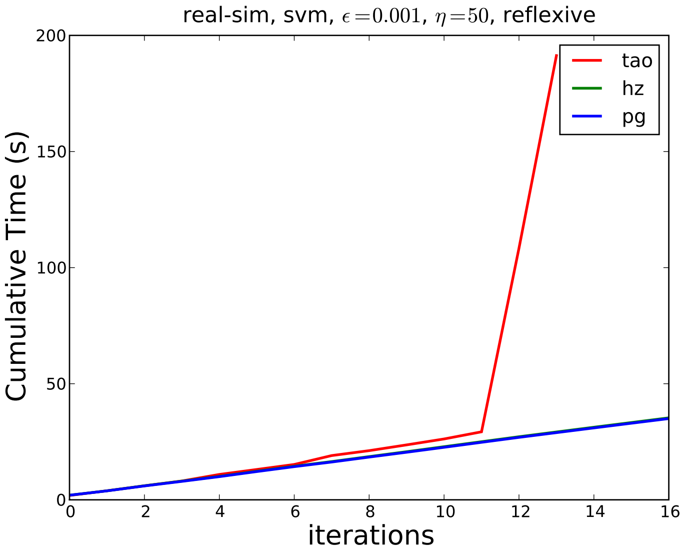
Comparing Different Optimizers



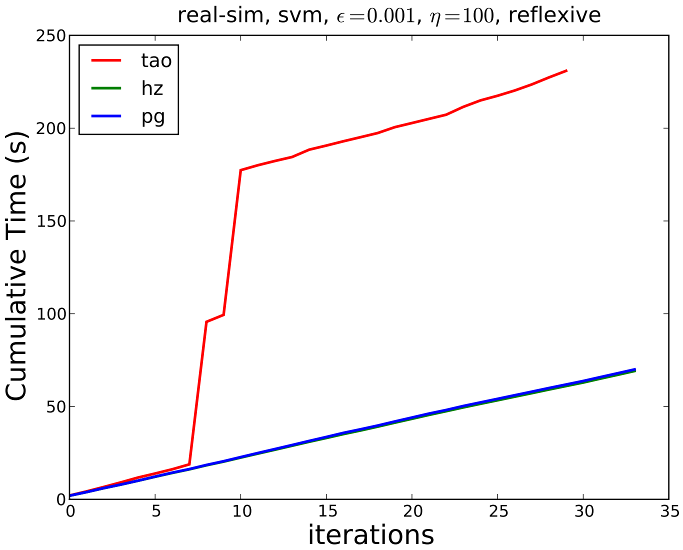
Comparing Different Optimizers



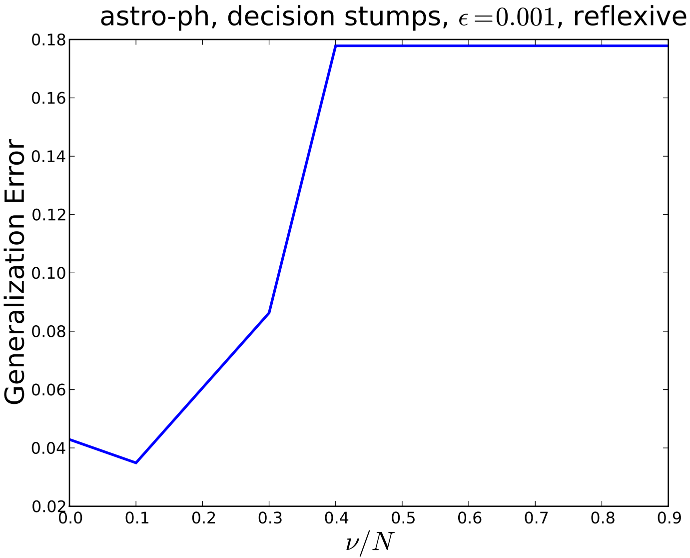
Comparing Different Optimizers



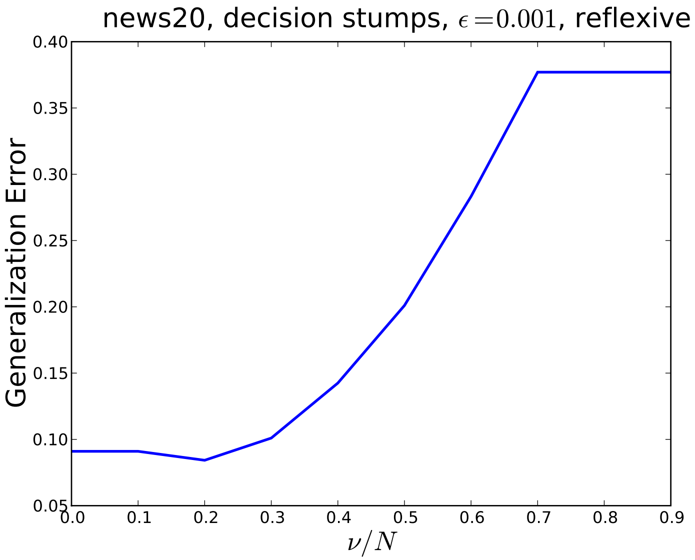
Comparing Different Optimizers



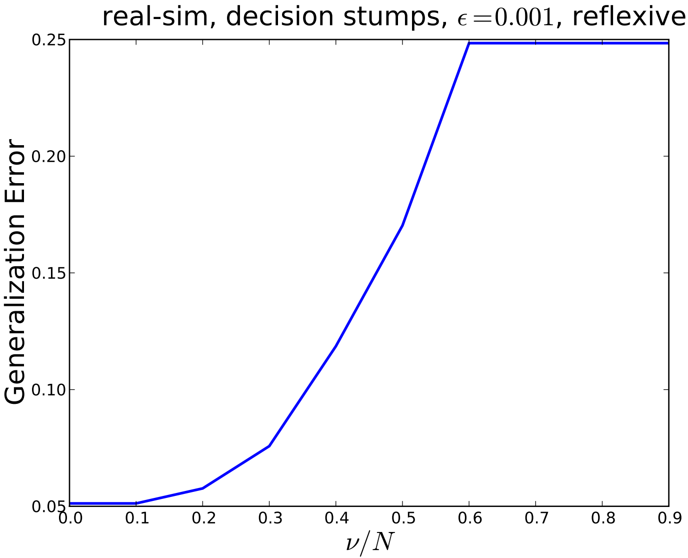
Effect of ν


astro-ph, decision stumps, $\epsilon = 0.001$, reflexive
Generalization Error
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

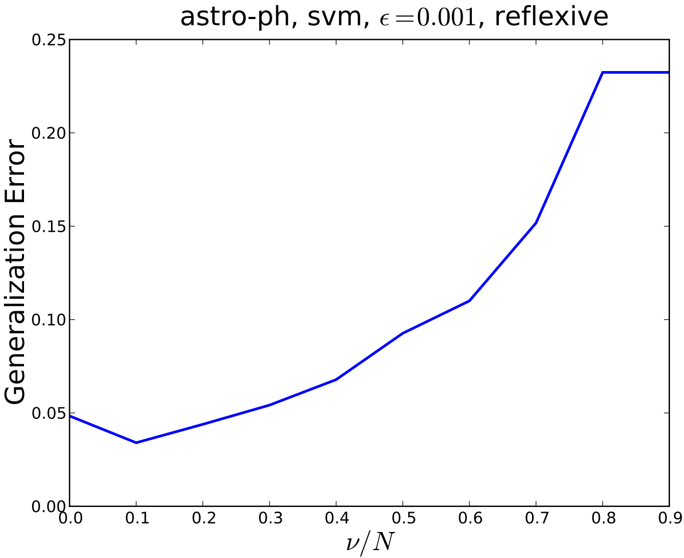
Effect of ν


news20, decision stumps, $\epsilon = 0.001$, reflexive
Generalization Error
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

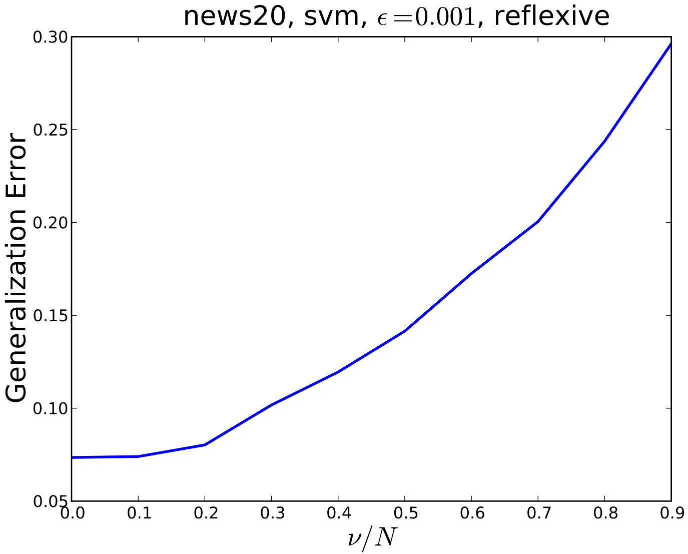
Effect of ν


real-sim, decision stumps, $\epsilon = 0.001$, reflexive
Generalization Error
0.05
0.10
0.15
0.20
0.25
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

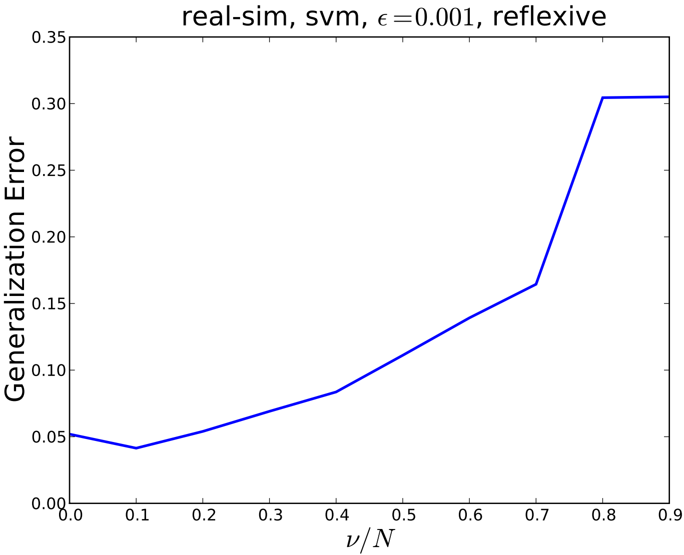
Effect of ν


astro-ph, svm, $\epsilon = 0.001$, reflexive
Generalization Error
0.25
0.20
0.15
0.10
0.05
0.00
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

Effect of ν


news20, svm, $\epsilon = 0.001$, reflexive
Generalization Error
0.05
0.10
0.15
0.20
0.25
0.30
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

Effect of ν



real-sim, svm, $\epsilon = 0.001$, reflexive
Generalization Error
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ν/N

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines . . .

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines ...

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines ...

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines ...

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines . . .

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines . . .

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines . . .

SVMs vs Boosting

SVMs

- Simple Quadratic Optimization Problem
- A decade of experience
- Can handle $\approx 10^6$ points

Boosting

- Softmax problem (log, exp, and simplex constraints)
- Just starting out
- Can handle $\approx 10^4$ points

Our Code

- Freely available under the MPL
- Scaling to large datasets (lots of room for improvement)
- Ideas on how to prune weak learners
- Still looking for large datasets where boosting shines ...

Conclusion

- Lots of exciting connections between boosting and optimization (we are only scratching the surface)
- Bring entropic regularization algorithms up to par with squared Euclidean distance regularization
- Look for datasets that exploit merits of new algorithms
- Find artificial datasets that highlight advantages of different families of algorithms
- Better lower bounds (the case of the missing log n)

Acknowledgements

- Manfred Warmuth for teaching me about boosting
- Karen Glocer for helping with the plots and code
- Ankan Saha, Choon Hui Teo, Jin Yu, and Xinhua Zhang for technical discussions