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

\[
\min_{d \in S^N} \max_{u \in U} \langle u, d \rangle
\]

\[f(d)\]
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]
Monitoring Convergence
Cutting Plane Methods work by forming the piecewise linear lower bound

\[ f(x) \geq f^\text{CP}_t(x) := \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}. \]

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

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

\[ x_t := \arg\min_x f^\text{CP}_t(x). \]

Stop when the duality gap

\[ \epsilon_t := \min_{0 \leq i \leq t} f(x_i) - f^\text{CP}_t(x_t) \]

falls below a pre-specified threshold \( \epsilon \).
In a Nutshell

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

\[ f(x) \geq f^\text{CP}_t(x) := \max_{1 \leq i \leq t} \{ f(x_{i-1}) + \langle x - x_{i-1}, s_i \rangle \}. \]

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

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

\[ x_t := \arg\min_x f^\text{CP}_t(x). \]

- Stop when the duality gap

\[ \epsilon_t := \min_{0 \leq i \leq t} f(x_i) - f^\text{CP}_t(x_t) \]

falls below a pre-specified threshold \( \epsilon \).
Cutting Plane Methods work by forming the piecewise linear lower bound

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

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

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

$$x_t := \arg\min_x f^\text{CP}_t(x).$$

Stop when the duality gap

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

falls below a pre-specified threshold $\epsilon$. 

In a Nutshell
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 $\mu$ which satisfies

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

Set of all subgradients is denoted as $\partial f(y)$
Subgradients to the Rescue

A subgradient at $y$ is any vector $\mu$ which satisfies

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

for all $x$

Set of all subgradients is denoted as $\partial f(y)$
A subgradient at $y$ is any vector $\mu$ 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

\[
\min_{d \in S^N} \max_{u \in U} \langle u, d \rangle
\]

is a convex optimization problem.

- Subgradient: \( \partial f(d) = \arg\max_{u \in U} \langle u, d \rangle \)

Computing Subgradient of \( f(d) \) = Strong Oracle!
Boosting as an Optimization Problem

- Minimizing the maximum edge

\[
\min_{d \in \mathbb{S}^N} \max_{u \in U} \langle u, d \rangle
\]

is a convex optimization problem.

- Subgradient: \( \partial f(d) = \arg\max_{u \in U} \langle u, d \rangle \)

Computing Subgradient of \( f(d) \) = Strong Oracle!
Subgradients and Stability
Subgradients and Stability
Subgradients and Stability
Picking an arbitrary subgradient can cause stability issues
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

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

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

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

where \( \Omega \) 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

\[ x_t := \arg\min_x f_t(x). \]

- 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 \( \epsilon \).
Bundle Methods [HL93]

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

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

where \( \Omega \) 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

\[ x_t := \arg\min_x f_t(x). \]

- 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 \( \epsilon \).
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 \( \Omega \) 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

\[
x_t := \arg\min_x f_t(x).
\]

- 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 \( \epsilon \).
**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 \( \Omega \) 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

\[ x_t := \arg\min_x f_t(x). \]

- 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 \( \epsilon \).
Rates of Convergence [SVL08]

Nonsmooth Functions

The number of iterations to reach $\epsilon$ 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 $\sigma$ is the modulus of strong convexity of $\Omega$. 
Lemma

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

$$\min_{0 \leq i \leq t} \Omega(d_i) + f(d_i) - f_t(d_t) \leq \epsilon/2.$$  

Then

$$f(d_t) \leq f(d^*) + \epsilon$$
Entropy as a Regularizer

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

- $\Omega(d) \leq \epsilon/2$ for all $d$
- $\Omega$ 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 $\epsilon$ accurate solution [SSS08].

Can also prove similar iteration bounds for an oracle which instead of returning $\arg\max_{u \in U} \langle u, d \rangle$ returns an $u$ such that $\langle u, d \rangle \geq g$ [WGV08].
Bundle Methods

Proving Iteration Bounds Contd.

Entropy as a Regularizer

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

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

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 $\epsilon$ accurate solution [SSS08].

Can also prove similar iteration bounds for an oracle which instead of returning $\arg\max_{u \in U} \langle u, d \rangle$ returns an $u$ such that $\langle u, d \rangle \geq g$ [WGV08].
Suppose we set $\Omega(d) = \frac{\epsilon}{2\log N} \sum_{i=1}^{N} d_i \log d_i$ then

- $\Omega(d) \leq \epsilon/2$ for all $d$
- $\Omega$ 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 $\epsilon$ accurate solution [SSS08].

Can also prove similar iteration bounds for an oracle which instead of returning $\arg\max_{u \in U} \langle u, d \rangle$ returns an $u$ such that $\langle u, d \rangle \geq g$ [WGV08].
### Setting
- Let $B$ be any black box boosting algorithm
- $d^1, \ldots, d^{t-1}$ are intermediate distributions
- $d^t$ is the distribution produced by $B$ after $t$ iterations

### The Adversary
- Produces a set of hypothesis $u_1, \ldots, u_t$ which defines a function

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

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

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

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

**The Adversary**
- Produces a set of hypothesis $u_1, \ldots, u_t$ which defines a function
  \[ f(d) := \max_{q=1,\ldots,t} \langle u^q, d \rangle \]
- We will show that $f(d^t) - \min_d f(d) \geq \frac{1}{\sqrt{t}}$
- To get $\epsilon$ accuracy $B$ needs $O(1/\epsilon^2)$ iterations.
Lower Bounds

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

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

The Adversary
- Produces a set of hypothesis $u_1, \ldots, u_t$ which defines a function

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

- We will show that $f(d^t) - \min_d f(d) \geq \frac{1}{\sqrt{t}}$
- To get $\epsilon$ accuracy $B$ needs $O(1/\epsilon^2)$ iterations.
Ω(1/\epsilon^2) Iteration Bounds [NY78,BMN01]

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

The Adversary
- Produces a set of hypothesis $u_1, \ldots, u_t$ which defines a function
  \[
  f(d) := \max_{q=1,\ldots,t} \langle u^q, d \rangle
  \]
- We will show that $f(d^t) - \min_d f(d) \geq \frac{1}{\sqrt{t}}$
- To get $\epsilon$ accuracy $B$ needs $O(1/\epsilon^2)$ iterations.
**Lower Bounds**

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

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

### The Adversary
- Produces a set of hypothesis \( u_1, \ldots, u_t \) which defines a function

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

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

**Ω(1/ε²) Iteration Bounds** [NY78,BMN01]

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

### The Adversary
- Produces a set of hypothesis $u_1, \ldots, u_t$ which defines a function
  \[ f(d) := \max_{q=1,\ldots,t} \langle u^q, d \rangle \]
- We will show that $f(d^t) - \min_d f(d) \geq \frac{1}{\sqrt{t}}$
- To get $\epsilon$ accuracy $B$ needs $O(1/\epsilon^2)$ iterations.
The Resisting Oracle

Hadamard Matrix

The Hadamard matrix is defined recursively:

\[
H_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad \quad \quad H_{2N} = \begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix}
\]

Hypothesis Space

The oracle chooses \(u^q\) from the columns of the following matrix:

\[
U = \begin{pmatrix} H_{N/2} & -H_{N/2} \\ -H_{N/2} & H_{N/2} \end{pmatrix}
\]

Because of symmetry, for any \(t < N/2\) we have

\[
f(d^t) = \max_{u \in U} \langle u, d^t \rangle \geq 0.
\]
The Resisting Oracle

Hadamard Matrix

The Hadamard matrix is defined recursively:

\[ H_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad H_{2N} = \begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix} \]

Hypothesis Space

- The oracle chooses \( u^q \) from the columns of the following matrix:

\[ U = \begin{pmatrix} H_{N/2} & -H_{N/2} \\ -H_{N/2} & H_{N/2} \end{pmatrix}. \]

- Because of symmetry, for any \( t < N/2 \) we have

\[ f(d^t) = \max_{u \in U} \langle u, d^t \rangle \geq 0. \]
The Resisting Oracle

The Hadamard Matrix

The Hadamard matrix is defined recursively:

\[
H_2 = \begin{pmatrix}
+1 & +1 \\
+1 & -1
\end{pmatrix}
\]

\[
H_{2N} = \begin{pmatrix}
H_N & H_N \\
H_N & -H_N
\end{pmatrix}
\]

Hypothesis Space

- The oracle chooses \( u^q \) from the columns of the following matrix:

\[
U = \begin{pmatrix}
H_{N/2} & -H_{N/2} \\
-H_{N/2} & H_{N/2}
\end{pmatrix}
\]

- Because of symmetry, for any \( t < N/2 \) we have

\[
f(d^t) = \max_{u \in U} \langle u, d^t \rangle \geq 0.
\]
Upper Bound on \( \min_d f(d) \)

\[
\begin{align*}
\min_{d \in S^N} f(d) &= \min_{d \in S^n} \max_{q=1,\ldots,t} \langle u^q, d \rangle \\
(Mixed \ strategy) &= \min_{d \in S^N} \max_{w \in S^t} d^\top U_t w \\
(Convexity) &= \max_{w \in S^t} \min_{d \in S^N} d^\top U_t w \\
(Pure \ Strategy) &= \max_{w \in S^t} \min_{j=1,\ldots,n} [U_t w]_j.
\end{align*}
\]
The Resisting Oracle Contd.

**Uₜ has vertical Symmetry**

Since columns of \( U_t \) are chosen from

\[
U = \begin{pmatrix}
H_{n/2} & -H_{n/2} \\
-H_{n/2} & H_{n/2}
\end{pmatrix}
\]

we can write \( U_t = \begin{pmatrix}
\hat{U}_t \\
-\hat{U}_t
\end{pmatrix} \).

\[
\min_{d \in S^N} f(d) = \max_{w \in S^t} \min_{j=1,\ldots,n} [U_t w]_j \\
= \max_{w \in S^t} - \left( \max_{j=1,\ldots,n} [\hat{U}_t w]_j \right) \\
= -\min_{w \in S^t} \|\hat{U}_t w\|_\infty
\]
The Resisting Oracle Contd.

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

\[
\min_{d \in S^N} f(d) = - \min_{w \in S^t} \|\hat{U}_t w\|_\infty \\
\leq - \frac{1}{\sqrt{n/2}} \min_{w \in S^t} \|\hat{U}_t w\|_2 \\
= - \frac{1}{\sqrt{n/2}} \min_{w \in S^t} \sqrt{w^\top \hat{U}_t^\top \hat{U}_t w} \\
= - \min_{w \in S^t} \sqrt{w^\top w} \\
= - \frac{1}{\sqrt{t}}
\]
The Optimization Problem

**Primal Problem**

\[
\begin{align*}
\min_{d \cdot I = 1} & \quad \frac{1}{\eta} \Delta(d, d^0) + \max_{q=1,2,\ldots,t} \langle u^q, d \rangle \\
\text{s.t.} & \quad d \leq \frac{1}{\nu} I
\end{align*}
\]

\( : = P^t(d) \)

**Dual Problem**

\[
\begin{align*}
\max_{w \geq 0} & \quad -\frac{1}{\eta} \log \sum_{n=1}^{N} d^0_n \exp(-\eta \sum_{q=1}^{t} u^q_n w_q - \eta \psi_n) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n \\
\text{s.t.} & \quad \langle w, e \rangle = 1
\end{align*}
\]
The Optimization Problem

**Primal Problem**

\[\begin{align*}
\min_{d \cdot I = 1} \quad & \frac{1}{\eta} \Delta(d, d^0) + \max_{q=1,2,\ldots,t} \langle u^q, d \rangle \\
\text{s.t.} \quad & d \leq \frac{1}{\nu} I \\
& d \leq \frac{1}{\nu} I \\
\end{align*}\]

\[:= P_t(d)\]

**Dual Problem**

\[\begin{align*}
\max_{w \geq 0} \quad & -\frac{1}{\eta} \log \sum_{n=1}^{N} d_n^0 \exp\left(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta \psi_n\right) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n \\
\text{s.t.} \quad & \langle w, e \rangle = 1 \\
& \psi \geq 0
\end{align*}\]
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 $\eta$ 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 \( \eta \) 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 $\eta$ 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

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 \( P_\Gamma(\mathbf{z}) := \arg\min_{x \in \Omega} \| \mathbf{z} - x \|^2 \).
Projected Gradient Descent

Constrained

Suppose you want to minimize

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

Projected Gradient descent produces a sequence of iterates

$$w_{t+1} = P_{\Gamma}(w_t - \eta \nabla f(w))$$

where $P_{\Gamma}(z) := \arg\min_{x \in \Omega} \| z - x \|^2$. 
Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - Step 2.1: Compute $d_t = P_\Omega(w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$
  - Step 2.2: Set $w_+ = w_t + \lambda d_t$
  - Step 2.3: If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
    
    $\lambda_t = \lambda$, $w_{t+1} = w_+$, $s_k = w_{t+1} - w_t$, $y_t = \nabla f(w_{t+1}) - \nabla f(w_t)$
    
    Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle s_t, y_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$
  - Else compute $a_t = \langle y_t, y_t \rangle$ and set
    
    $\alpha_{t+1} = \min\{\alpha_{\text{max}}, \max\{\alpha_{\text{min}}, a_t/b_t\}\}$
**Spectral Projected Gradient Method [BMR00]**

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $d_t = P_{\Omega}(w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $w_+ = w_t + \lambda d_t$
  - **Step 2.3:** If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
    - $\lambda_t = \lambda$, $w_{t+1} = w_+$, $s_k = w_{t+1} - w_t$, $y_t = \nabla f(w_{t+1}) - \nabla f(w_t)$
    - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle s_t, y_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$
  - Else compute $a_t = \langle y_t, y_t \rangle$ and set $\alpha_{t+1} = \min\{\alpha_{\text{max}}, \max\{\alpha_{\text{min}}, a_t/b_t\}\}$
Spectral Projected Gradient Method [BMR00]

- **Step 1:** Detect if current point is stationary
- **Step 2:** Backtracking
  - **Step 2.1:** Compute $d_t = P_\Omega(w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $w_+ = w_t + \lambda d_t$
  - **Step 2.3:** If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
    - $\lambda_t = \lambda$, $w_{t+1} = w_+$, $s_k = w_{t+1} - w_t$, $y_t = \nabla f(w_{t+1}) - \nabla f(w_t)$
    - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle s_t, y_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$
  - Else compute $a_t = \langle y_t, y_t \rangle$ and set
    $\alpha_{t+1} = \min\{ \alpha_{\text{max}}, \max\{ \alpha_{\text{min}}, a_t / b_t \} \}$
### Spectral Projected Gradient Method \([\text{BMR00}]\)

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

Step 2: Backtracking

Step 2.1: Compute $d_t = P_\Omega (w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$

Step 2.2: Set $w_+ = w_t + \lambda d_t$

Step 2.3: If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
$$\lambda_t = \lambda, w_{t+1} = w_+, s_k = w_{t+1} - w_t, y_t = \nabla f(w_{t+1}) - \nabla f(w_t)$$
Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2

Step 3: Compute $b_t = \langle s_t, y_t \rangle$.

If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$

Else compute $a_t = \langle y_t, y_t \rangle$ and set
$$\alpha_{t+1} = \min \{\alpha_{\text{max}}, \max \{\alpha_{\text{min}}, a_t/b_t\}\}$$
**Step 1:** Detect if current point is stationary

**Step 2:** Backtracking
- **Step 2.1:** Compute $d_t = P_\Omega (w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$
- **Step 2.2:** Set $w_+ = w_t + \lambda d_t$
- **Step 2.3:** If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
  - $\lambda_t = \lambda$, $w_{t+1} = w_+$, $s_k = w_{t+1} - w_t$, $y_t = \nabla f(w_{t+1}) - \nabla f(w_t)$
  - Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2

**Step 3:** Compute $b_t = \langle s_t, y_t \rangle$.
- If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$
- Else compute $a_t = \langle y_t, y_t \rangle$ and set $\alpha_{t+1} = \min \{\alpha_{\text{max}}, \max \{\alpha_{\text{min}}, a_t/b_t\}\}$
Optimization in Focus

Spectral Projected Gradient Method [BMR00]

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

  \[ \lambda_t = \lambda, \quad \mathbf{w}_{t+1} = \mathbf{w}^+, \quad \mathbf{s}_k = \mathbf{w}_{t+1} - \mathbf{w}_t, \quad \mathbf{y}_t = \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t) \]

  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 $d_t = P_{\Omega}(w_t - \alpha_t \nabla f(w)) - w_t$. Set $\lambda = 1$
  - **Step 2.2:** Set $w_+ = w_t + \lambda d_t$
  - **Step 2.3:** If $f(w_+) \leq \max_{0 \leq j \leq M} f(w_{t-j}) + \gamma \lambda \langle d_t, \nabla f(w) \rangle$
    \[ \lambda_t = \lambda, \quad w_{t+1} = w_+, \quad s_k = w_{t+1} - w_t, \quad y_t = \nabla f(w_{t+1}) - \nabla f(w_t) \]
    Else define $\lambda \in [\sigma_1 \lambda, \sigma_2 \lambda]$ and go to Step 2.2
- **Step 3:** Compute $b_t = \langle s_t, y_t \rangle$.
  - If $b_t \leq 0$ set $\alpha_{t+1} = \alpha_{\text{max}}$
  - Else compute $a_t = \langle y_t, y_t \rangle$ and set
    \[ \alpha_{t+1} = \min\{\alpha_{\text{max}}, \max\{\alpha_{\text{min}}, a_t/b_t\}\} \]
## Dataset Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Dimension</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astro-ph</td>
<td>94,856</td>
<td>99,757</td>
<td>0.008</td>
</tr>
<tr>
<td>News20</td>
<td>19,954</td>
<td>1,355,191</td>
<td>0.03</td>
</tr>
<tr>
<td>Real-Sim</td>
<td>72,201</td>
<td>20,958</td>
<td>0.25</td>
</tr>
<tr>
<td>rcv1</td>
<td>677,399</td>
<td>47,236</td>
<td>0.15</td>
</tr>
</tbody>
</table>

- 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

astro-ph, lpboost, decision stumps, $\epsilon=0.05$, reflexive
LPBoost is Brittle

/news20, lpboost, decision stumps, $\epsilon=0.05$, reflexive
LPBoost is Brittle

real-sim, lpboost, decision stumps, $\epsilon=0.05$, reflexive

Generalization Error vs Iteration
LPBoost is Brittle

astro-ph, lpboost, svm, $\epsilon = 0.01$, reflexive

Warmuth (UCSC)
LPBoost is Brittle

![Graph showing generalization error over iterations for news20, lpboost, svm, $\epsilon = 0.01$, reflexive]

- Generalization Error
- Iteration
- News20, lpboost, svm, $\epsilon = 0.01$, reflexive
LPBoost is Brittle

real-sim, lpboost, svm, \( \epsilon = 0.01 \), reflexive

![Graph showing generalization error over iterations for real-sim, lpboost, svm with \( \epsilon = 0.01 \), and reflexive.](image)
ERLPBoost fixes the problem

astro-ph, erlp, decision stumps, $\epsilon=0.001$, reflexive

Generalization Error

- $\eta=20.0$
- $\eta=200.0$
- $\eta=2000.0$
ERLPBoost fixes the problem

news20, erlp, decision stumps, $\epsilon=0.001$, reflexive

Generalization Error

Iteration

Warmuth (UCSC) MLSS 2010, ANU
ERLPBoost fixes the problem

real-sim, erlp, decision stumps, $\epsilon=0.001$, reflexive

Generalization Error

Iteration

$\eta = 20.0$
$\eta = 200.0$
$\eta = 2000.0$
ERLPBoost fixes the problem

The diagram illustrates the generalization error over iterations for different learning rates ($\eta$). The generalization error decreases with increasing iterations for all learning rates, but the rate of decrease varies. The graph shows the results for $\eta = 20$, $\eta = 200$, and $\eta = 2000$, with generalization error values plotted on a logarithmic scale.

Key points:
- Generalization error decreases as iterations increase.
- Different learning rates affect the rate of decrease.
- The graph indicates that higher learning rates may lead to faster convergence but could also cause overfitting in certain cases.

Parameters used:
- $\epsilon = 0.001$
- reflexive

Experiment conducted by Warmuth (UCSC) at MLSS 2010, ANU.
ERLPBoost fixes the problem

```
news20, erlp, svm, \( \epsilon = 0.001 \), reflexive

Iteration

0.0  0.1  0.2  0.3  0.4  0.5  0.6

Generalization Error

\( \eta = 20 \)
\( \eta = 200 \)
\( \eta = 2000 \)
```
ERLPBoost fixes the problem

![Graph showing generalization error over iteration for real-sim, erlp, svm, $\epsilon=0.001$, reflexive. The graph plots generalization error on the y-axis and iteration on the x-axis. There are three curves: blue for $\eta=20$, red for $\eta=200$, and green for $\eta=2000$. The error decreases with increasing iteration.]
Generalization as a function of $\eta$

astro-ph, decision stumps, $\epsilon = 0.001$, reflexive
Generalization as a function of $\eta$

- Experiment: news20, decision stumps, $\epsilon = 0.001$, reflexive

- Graph shows generalization error as a function of $\eta$.
Generalization as a function of $\eta$

![Graph showing generalization error as a function of $\eta$ for news20, decision stumps, $\epsilon=0.001$, reflexive.

Warmuth (UCSC)  MLSS 2010, ANU
Generalization as a function of $\eta$
Generalization as a function of $\eta$

![Graph showing the generalization error for news20, svm, $\epsilon = 0.001$, reflexive, as a function of $\eta$.]
Generalization as a function of $\eta$

![Graph showing generalization error as a function of $\eta$. The x-axis represents $\eta$ on a logarithmic scale from $10^0$ to $10^5$, and the y-axis represents generalization error from 0.35 down to 0.00. The graph demonstrates that the generalization error decreases as $\eta$ increases. The label 'real-sim, svm, $\epsilon = 0.001$, reflexive' is also present on the graph.]
Generalization Error and # of Weak Hypothesis

Parameters tuned for best generalization performance

Warmuth (UCSC)  MLSS 2010, ANU
Parameters tuned for best generalization performance
Experimental Evaluation

Generalization Error and # of Weak Hypothesis

Parameters tuned for best generalization performance

![Graph showing generalization error and number of weak learners for different algorithms.](image)
Corrective vs Totally Corrective

Experimental Evaluation

astro-ph, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive

iteration

$\epsilon$ gap

$10^{-3}$ $10^{-2}$ $10^{-1}$ $10^0$ $10^1$

0 2000 4000 6000 8000 10000 12000 14000 16000

$\epsilon$ gap vs iteration

Warmuth (UCSC)
MLSS 2010, ANU
Corrective vs Totally Corrective

news20, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive
corr
erlp
Corrective vs Totally Corrective

real-sim, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive
Experimental Evaluation

Corrective vs Totally Corrective

astro-ph, svm, $\eta=500$, $\epsilon=0.001$, reflexive

Warmuth (UCSC)
Corrective vs Totally Corrective

news20, svm, $\eta=500$, $\epsilon=0.001$, reflexive

Warmuth (UCSC)  MLSS 2010, ANU
Corrective vs Totally Corrective

real-sim, svm, $\eta=500$, $\epsilon=0.001$, reflexive

$\epsilon$ gap vs iteration

$\eta=500$, $\epsilon=0.001$, reflexive
Corrective vs Totally Corrective contd.

astro-ph, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive
Corrective vs Totally Corrective contd.

news20, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive
Corrective vs Totally Corrective contd.

real-sim, decision stumps, $\eta=500$, $\epsilon=0.001$, reflexive
corr
erlp
Corrective vs Totally Corrective contd.

astro-ph, svm, $\eta=500$, $\epsilon=0.001$, reflexive

time (s)

$\epsilon$ gap

Warmuth (UCSC)  MLSS 2010, ANU
Corrective vs Totally Corrective contd.

news20, svm, $\eta=500$, $\epsilon=0.001$, reflexive
Experimental Evaluation

Corrective vs Totally Corrective contd.

real-sim, svm, \( \eta = 500, \epsilon = 0.001 \), reflexive

\[\epsilon_{\text{gap}}\]

\[\text{time (s)}\]

Warmuth (UCSC)  MLSS 2010, ANU
Comparing Different Optimizers

real-sim, decision stumps, $\epsilon = 0.001$, $\eta = 1$, reflexive
Comparing Different Optimizers

real-sim, decision stumps, $\epsilon = 0.001$, $\eta = 20$, reflexive
Comparing Different Optimizers

real-sim, decision stumps, $\epsilon = 0.001$, $\eta = 50$, reflexive

- tao
- hz
- pg
Comparing Different Optimizers

real-sim, decision stumps, $\epsilon=0.001$, $\eta=100$, reflexive

- red: tao
- green: hz
- blue: pg
Comparing Different Optimizers

real-sim, svm, $\epsilon = 0.001$, $\eta = 20$, reflexive

- tao
- hz
- pg

Cumulative Time (s)

iterations
Comparing Different Optimizers

real-sim, svm, $\epsilon = 0.001$, $\eta = 50$, reflexive

Cumulative Time (s)

iterations

Warmuth (UCSC) MLSS 2010, ANU
Comparing Different Optimizers

Experimental Evaluation

real-sim, svm, $\epsilon=0.001$, $\eta=100$, reflexive

Cumulative Time (s) vs. Iterations

- tao
- hz
- pg
Effect of $\nu$

Generalization Error vs. $\nu/N$

astro-ph, decision stumps, $\epsilon=0.001$, reflexive
Effect of $\nu$

News20, decision stumps, $\epsilon = 0.001$, reflexive
Effect of $\nu$

real-sim, decision stumps, $\epsilon = 0.001$, reflexive
Effect of $\nu$
Effect of $\nu$

![Graph showing the effect of $\nu/N$ on generalization error. The x-axis represents $\nu/N$ ranging from 0.0 to 0.9, and the y-axis represents generalization error ranging from 0.05 to 0.30. The graph is labeled with the dataset 'news20', the SVM method, $\epsilon=0.001$, and is reflexive. The data is sourced from Warmuth (UCSC) at MLSS 2010, ANU.]
Effect of $\nu$

![Graph of Generalization Error vs. $\nu/N$]

**real-sim, svm, $\epsilon = 0.001$, reflexive**
### SVMs vs Boosting

<table>
<thead>
<tr>
<th>SVMs</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Simple Quadratic Optimization Problem</td>
<td>- Softmax problem (log, exp, and simplex constraints)</td>
</tr>
<tr>
<td>- A decade of experience</td>
<td>- Just starting out</td>
</tr>
<tr>
<td>- Can handle $\approx 10^6$ points</td>
<td>- Can handle $\approx 10^4$ points</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>SVMs</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Quadratic</td>
<td>Softmax problem (log, exp, and simplex constraints)</td>
</tr>
<tr>
<td>Optimization Problem</td>
<td>Just starting out</td>
</tr>
<tr>
<td>A decade of experience</td>
<td>Can handle $\approx 10^4$ points</td>
</tr>
<tr>
<td>Can handle $\approx 10^6$ points</td>
<td></td>
</tr>
</tbody>
</table>

### 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