Boosting as a Regularized Path

to a Maximum Margin Classifier

Saharon Rosset (IBM)
Ji Zhu (U. Michigan)
Trevor Hastie (Stanford)

Presented by
Charlie Olson
March 21, 2006
Outline

- Boosting as Gradient Descent
- Margins, Support Vector Machines, and Boosting
- Boosting as Approximate Incremental \( l_1 \) Constrained Fitting
- \( l_p \)-Constrained Classification Loss Functions
Boosting as Gradient Descent

One way of looking at Boosting:

\[ F_T(x) = \sum_{t=1}^{T} \alpha_t h_{j_t}(x) \]

\(x\) is the input vector.

\(h_{j_t}\) is the best hypothesis to use at step \(t\) (based on information gain, etc.)

\(\alpha_t\) is the corresponding weight of that hypothesis at step \(t\).
Boosting as Gradient Descent

A simpler perspective of Boosting:

\[ F_T(x) = \sum_{j=1}^{J} h_j(x) \cdot \beta_j^{(T)} \]
Boosting as Gradient Descent

A simpler perspective of Boosting:

\[ F_T(x) = \sum_{j=1}^{J} h_j(x) \cdot \beta_j^{(T)} \]

\( \beta \) is the coefficient vector
(each element \( \beta_j \) is the sum of all \( \alpha \)'s ever applied to the corresponding \( h_j \))

\( J \) is the total number of hypotheses in the dictionary
Boosting as Gradient Descent

So you could simply write:

\[ F(x) = \beta \cdot h(x) \]

where \( \beta \) is a normal vector, and \( h(x) \) is a vector in hypothesis-space (\( x \) mapped into hypothesis space).
Boosting as Gradient Descent

In other words...
F(x) = β • h(x) = the projection onto the normal, which is your classification prediction,

but you care about how correct you are, so find a β-hat that minimizes the loss function C(y,F) over all training examples:

\[ \hat{\beta}(c) = \arg \min_{\|\beta\|_1 \leq c} \sum_i C(y_i, h(x_i)'\beta) \]
Boosting as Gradient Descent

\[ \hat{\beta}(c) = \arg \min_{\|\beta\|_1 \leq c} \sum_i C(y_i, h(x_i)'\beta) \]

Limit possible values of \( \beta \) to 1-norms less than \( c \)...

Large values of \( c \) would send the loss toward zero if the training data was separated.

(could \( c > 1 \) then be considered a regularizer..?)
Boosting as Gradient Descent

The meat of Boosting as Gradient Descent:

Find a good value for the β vector (one that minimizes the total loss) using an iterative process:
1. Scan for the coordinate of β whose change has the best effect on the loss function.
2. Step in that direction.

Variations like line-search can also work (AdaBoost). If the dictionary is large, settle for an okay coordinate direction rather than “the best”.

Boosting as Gradient Descent

Coordinate descent is probably self-explanatory, but here is an algorithm:

Algorithm 1 Generic gradient-based boosting algorithm

1. Set $\beta^{(0)} = 0$.

2. For $t = 1 : T$,

   (a) Let $F_i = \beta^{(t-1)'h(x_i)}$, $i = 1, \ldots, n$ (the current fit).
   
   (b) Set $w_i = \frac{\partial C(y_i, F_i)}{\partial F_i}$, $i = 1, \ldots, n$.
   
   (c) Identify $j_t = \arg\max_j |\sum_i w_i h_j(x_i)|$.
   
   (d) Set $\beta_{j_t}^{(t)} = \beta_{j_t}^{(t-1)} - \alpha_t \text{sign}\left(\sum_i w_i h_{j_t}(x_i)\right)$ and $\beta_{k}^{(t)} = \beta_{k}^{(t-1)}$, $k \neq j_t$. 
Common Loss Functions

Exponential: \( C_e(y, F) = \exp(-yF); \)
Loglikelihood: \( C_l(y, F) = \log(1 + \exp(-yF)) \)
Margins, SVMs, and Boosting
Margins, SVMs, and Boosting

(in hypothesis space:)

Margin

Large Marge
Margins, SVMs, and Boosting

$l_p$ margin: $m_p(\beta) = \min_i \frac{y_i F(x_i)}{\|\beta\|_p}$

$F(x) = \sum_j h_j(x) \beta_j$

$\beta$ is the normal to the separating hyperplane, normalize it, then margin is the minimal projection onto the normal.
Margins, SVMs, and Boosting

$l_2$ Euclidean margin:  

$l_1$ max. margin:

(this picture could be more helpful)
Margins, SVMs, and Boosting

One intuition for why $m_1$ is vertical, and $m_2$ is diagonal:

\[
\frac{yF(x)}{\|\beta\|_1} = \frac{yF(x)}{\|\beta\|_2} \cdot \frac{\|\beta\|_2}{\|\beta\|_1}
\]

$m_1$ will be large when the $\beta$-ratio is large, which happens when $\beta$ is sparse.
Margins, SVMs, and Boosting

Or just notice that $m_1$ is larger if $\|\beta\|_1$ is smaller.

Keep $\|\beta\|_1$ small by staying as close to a single axis as possible... the more zeroes in $\beta$ the better.

(the “sparsity” effect)
Margins, SVMs, and Boosting

Coordinate descent attempts to separate in the $l_1$-margin sense.

By stepping along the best axis each iteration it tries to find $\beta$ with a minimal $1$-norm.

If it moves monotonically towards $\beta$

$$\beta_{j_t} \neq 0 \Rightarrow \text{sign}(\alpha_t) = \text{sign}(\beta_{j_t})$$

Then the sum of the steps, $\|\alpha\|_1$, is the same as $\|\beta\|_1$ (not profound, but used later)
Margins, SVMs, and Boosting

Margin-maximization leads to over-fitting...
Boosting as Approximate Incremental $l_1$ Constrained Fitting
Boosting as Approximate Incremental $l_1$ Constrained Fitting

If the optimal $l_1$ path is monotone, then traditional coordinate descent using infinitely-small steps will result in the same $l_1$-optimal solution path.

(identical)
Boosting as Approximate Incremental $l_1$ Constrained Fitting

If the solution path is non-monotone, then the similarity breaks down:
Boosting as Approximate Incremental $l_1$ Constrained Fitting

The points:

Boosting follows the optimal $l_1$ constrained path if the step size is infinitely small, and the optimal path is monotone, by moving in the locally optimal $l_1$ direction.

But, realistic step sizes only approximate the optimal path.

And.. it only works for monotone paths.
$l_p$-Constrained Classification Loss Function
$l_p$-Constrained Classification Loss Function

Authors prove that if there is a unique $l_p$-margin maximizing hyper-plane, then the normalized constrained solution converges to it.

Recall “normalized constrained” from earlier:

$$\|\beta\|_p \leq c_{\text{max}}$$

as in:

$$\hat{\beta}(c) = \arg \min_{\|\beta\|_1 \leq c} \sum_i C(y_i, h(x_i)'\beta)$$
$l_p$-Constrained Classification
Loss Functions

Can turn coordinate-descent into $l_2$ boosting, and maximize the $l_2$-margin, by choosing the coordinate that has the greatest proportional effect on $\beta$

Choose the coordinate to maximize:

$$\frac{|\sum_i w_i h_j(x_i)|}{|\beta_j|}$$

This has problems in reality, but is still cool.