Boosting

CSE 250B

Weak learners

It is often easy to come up with a weak classifier, one that is only slightly better than random guessing.

\[ \Pr(h(X) \neq Y) = \frac{1}{2} - \epsilon \]

A learning algorithm that can consistently generate such classifiers is called a weak learner.

Is it possible to systematically boost the quality of a weak learner?

Blueprint:

- Think of the data set \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\) as a distribution \(D\) on \(X \times Y\)
- Repeat for \(t = 1, 2, \ldots\):
  - Give \(D\) to the weak learner, get back a weak classifier \(h_t\)
  - Reweight \(D\) to put more emphasis on points that \(h_t\) gets wrong
  - Combine all these \(h_t\)'s somehow

Adaboost

Assume \(Y = \{-1, 1\}. \) Given: \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in X \times Y.\)

1. Initialize \(D_t(i) = 1/n\) for all \(i = 1, 2, \ldots, n\)
2. For \(t = 1, 2, \ldots, T:\)
   - Give \(D_t\) to weak learner, get back some \(h_t : X \rightarrow [-1, 1]\)
   - Update distribution:
     \[ r_t = \sum_{i=1}^{n} D_t(i) y^{(i)} h_t(x^{(i)}) \in [-1, 1] \text{ (} h_t\text{'s margin of success)} \]
     \[ \alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t} \in \mathbb{R} \text{ (strength of update)} \]
     \[ D_{t+1}(i) \propto D_t(i) \exp \left( -\alpha_t y^{(i)} h_t(x^{(i)}) \right) \]
3. Final classifier:
   \[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

Example

(From Freund and Schapire’s tutorial.)

Training set:

Weak classifiers: single-coordinate thresholds, popularly known as “decision stumps” (in this case, horizontal and vertical lines)
Example, cont’d

<table>
<thead>
<tr>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ + -</td>
<td>+ + -</td>
<td>+ + -</td>
</tr>
<tr>
<td>+ - -</td>
<td>+ - -</td>
<td>+ - -</td>
</tr>
<tr>
<td>h₁</td>
<td>h₂</td>
<td>h₃</td>
</tr>
<tr>
<td>r₁ = 0.40, α₁ = 0.42</td>
<td>r₂ = 0.58, α₂ = 0.65</td>
<td>r₃ = 0.72, α₃ = 0.92</td>
</tr>
</tbody>
</table>

Training error dropoff

The surprising power of a weak learner:

**Theorem.** Suppose that on each round $t$, the weak learner returns a classifier $h_t$ that differs from random by some margin $γ$:

$$\left| \sum_{i=1}^{n} D_t(i) y^{(i)} h_t(x^{(i)}) \right| \geq γ.$$ 

Then after $T$ rounds the training error is at most $e^{-γ^2 T/2}$.

Boosting decision stumps and trees

Final classifier:

$$\text{sign}(0.42h_1(x) + 0.65h_2(x) + 0.92h_3(x))$$
Boosting decision stumps and trees

Freund and Schapire: results on 27 benchmark data sets from the UCI repository.

Boosting versus decision tree

Freund and Schapire: results on 27 benchmark data sets from the UCI repository.

Overfitting?

Freund and Schapire: boosting decision trees for “letter” dataset.

- Test error doesn’t increase, even after 1000 rounds (over 2 million nodes total)
- Test error keeps dropping even after training error has gone to zero:

<table>
<thead>
<tr>
<th># rounds</th>
<th>train error</th>
<th>test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0</td>
<td>8.4</td>
</tr>
<tr>
<td>100</td>
<td>0.0</td>
<td>3.3</td>
</tr>
<tr>
<td>1000</td>
<td>0.0</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Looking at the margin

Recall the final classifier (with weights normalized to sum to 1):

\[ H(x) = \text{sign} \left( \frac{\sum \alpha_t h_t(x)}{\sum |\alpha|} \right) \]

The margin of this classifier on a particular \((x, y) \in \mathcal{X} \times \{-1, 1\}\) is:

\[ (\text{fraction of votes correct}) - (\text{fraction incorrect}) = yf(x) \in [-1, 1]. \]

Did \(H\) just barely get it right? Or definitively?

- Intuitively: the larger a classifier’s margins on the training data, the better its generalization – that is, the lower its true error.
- There are mathematical results that make this precise.
- Adaboost seems to increase the margins on the training points even after training error has gone to zero.
Example revisited

Look at cumulative distribution of margins of training examples:

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
<tr>
<td>% margins ≤ 0.5</td>
<td>7.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>minimum margin</td>
<td>0.14</td>
<td>0.52</td>
<td>0.55</td>
</tr>
</tbody>
</table>

Another view of boosting

Let $\mathcal{H}$ denote the set of base classifiers $\mathcal{X} \rightarrow \{-1, 1\}$.
For instance, $\mathcal{H} = \{\text{decision stumps}\}$.

Imagine a representation of $x \in \mathcal{X}$ in which each $h \in \mathcal{H}$ corresponds to a feature:

$$ \Phi(x) = (h(x) : h \in \mathcal{H}) $$

The final classifier found by boosting,

$$ H(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right) $$

call this $f(x)$.

is a linear classifier in this enhanced space.

What kind of linear classifier does boosting return? Is it optimizing some loss function?

Minimizing exponential loss

It can be shown that boosting is looking for the linear classifier that minimizes the exponential loss:

$$ \frac{1}{n} \sum_{i=1}^{n} e^{-y^{(i)} f(x^{(i)})} $$

This loss function is yet another convex upper bound on 0-1 loss:

Boosting is a coordinate descent procedure for minimizing the loss.