Machine Learning for Intelligent Systems

Lecture 24: Boosting

Reading: UML 10-10.3
Optional Readings: Schapire’s survey and tutorial

Instructors: Nika Haghtalab (this time) and Thorsten Joachims
I want a learning algorithm that for any distribution $P$ learns an 
**excellent** classifier $h_{strong}$ such that $err_P(h_{strong}) \leq 0.01$.

I’m given a learning algorithm $A$ that for any distribution $D$ returns 
a **not-too-terrible** classifier $h_{weak}$ such that $err_D(h_{weak}) \leq 0.49$.

Can I use this algorithm $A$ to find $h_{strong}$, 

$$err_P(h_{strong}) \leq 0.01?$$
**Strong versus Weak Learning**

**Strong Learner**

A learning algorithm for PAC learning.

For every distribution $P$ and every $\epsilon$, a strong learner can return a classifier $h$ such that $err_P(h) \leq \epsilon$. 

With probability $1 - \delta$

Error of random guessing: For any distribution $P$, ignore $P$ and

- for each $x$ predict $+1$ or $-1$, with probability 50-50.
- What’s the error?
- Exactly 0.5

**Weak Learner**

Better than random guessing.

For every distribution $P$ and some $\gamma > 0$, a weak Learner returns a classifier $h$ such that $err_P(h) \leq \frac{1}{2} - \gamma$. 

With probability $1 - \delta$
Is there a boosting algorithm that turns a weak learner into a strong learner?

Yes!

There is boosting algorithm that uses a weak learner on an adaptively designed polynomial-size sequence of distributions and strong learns.

Weak Learning = Strong Learning
Warmup

Suppose our weak learner knows when it doesn’t know!

- \( h: x \rightarrow \{+1, -1, \text{Not sure}\}. \)
- On at most \( 1 - \epsilon' \) fraction of the data, it can say “Not sure”.
- On the fraction of the data that it is sure, it makes \( \epsilon \) error.
- Leads to a weak learner, if “Not sure” \( \Rightarrow \) randomly guess:

\[
err_{P(h)} \leq \frac{1}{2} (1 - \epsilon') + \epsilon \epsilon' \leq \frac{1}{2} - \gamma \quad \text{for } \gamma = \epsilon' \left( \frac{1}{2} - \epsilon \right).
\]

Boosting:
- Start with a weak learner.
- Boost by focusing the distribution on instances the previous learner wasn’t sure about.
Warmup Analysis

Boost by a decision list:

• Train $h_i$ on $P_i$. Let $P_{i+1} \leftarrow P_i | \{x : h_i(x) = "Not sure"\}$.
• Repeat until the total prob. of the “Not sure” region is $\epsilon$.
• Total error at most $2\epsilon$.
• It only takes $T = \frac{1}{\epsilon'} \ln\left(\frac{1}{\epsilon}\right)$ rounds: $(1 - \epsilon')^T \leq \exp(-\epsilon'T) \leq \epsilon$.

Error on the sample it’s sure about: $\leq \epsilon$

Added after class: reason for the above. Conditioned on being sure, we are wrong with prob. $\leq \epsilon$. So, the total probability is $\leq \epsilon$.

Another way to see this is, prob. of error after each round: $\sum_{t=1}^{T} \epsilon \times \epsilon'(1 - \epsilon')^{t-1} \leq \epsilon$.

$\Pr_x[h_t(x) \text{ is wrong} | h_t(x) \text{ is sure}]$
**A Recipe for Boosting**

**Input:** \((x_1, y_1), \ldots, (x_m, y_m)\) and a weak learning algorithm.

Let \(P_1(x_i) = \frac{1}{m}\) for all \(i\), i.e., uniform distribution over samples.

For \(t = 1, \ldots, T\)
- Learn a weak classifier \(h_t \in H\) on distribution \(P_t\).
- Construct \(P_{t+1}\) that has **higher weight** compared to \(P\) on instance where \(h_1, \ldots, h_t\) didn’t perform well.

Output the final hypothesis

\[
h_{\text{final}}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t \, h_t(x) \right)
\]
Constructing $P_{t+1}$

Increase the weight of $x_i$ if $h_t$ made a mistake on it. Decrease the weight if $h_t$ was correct.

- Don’t want to cut the weight to 0
  - $h_{t+1}$ could be *arbitrarily bad* on where $h_t$ was good.
  - The majority vote could be bad.
- Change the weights, so that $h_t$ would have head error exactly 0.5

Use error $h_t$ on $P_t$

Change the weights, without normalizing

Normalize

$P_{t+1}$
Constructing $P_{t+1}$

Let $\epsilon_t = \Pr_{x_i \sim P_t} [h_t(x_i) \neq y_i]$ and let $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$. Let

$$P_{t+1}(x_i) = \frac{P_t(x_i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where $Z_t = \sum_i P_t(x_i) \exp(-\alpha_t y_i h_t(x_i))$ is the normalization factor.

$$P_{t+1}(x_i) = \begin{cases} 
\frac{P_t(x_i)}{Z_t} \exp(-\alpha_t) & \text{if } y_i = h_t(x_i) \\
\frac{P_t(x_i)}{Z_t} \exp(+\alpha_t) & \text{if } y_i \neq h_t(x_i) 
\end{cases}$$

Weight of $P_t$ on correct points:

$$\text{Weight on } h_t(x_i) = y_i: \quad \frac{1}{Z_t} (1 - \epsilon_t) \exp \left( -\frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) \right) = \frac{1}{Z_t} (1 - \epsilon_t) \left( \frac{\epsilon_t}{1-\epsilon_t} \right)^{1/2} = \frac{\sqrt{\epsilon_t (1-\epsilon_t)}}{Z_t}$$

Weight of $P_t$ on incorrect points:

$$\text{Weight on } h_t(x_i) \neq y_i: \quad \frac{1}{Z_t} \epsilon_t \exp \left( \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) \right) = \frac{1}{Z_t} \epsilon_t \left( \frac{1-\epsilon_t}{\epsilon_t} \right)^{1/2} = \frac{\sqrt{\epsilon_t (1-\epsilon_t)}}{Z_t}$$
Adaptive Boosting

**AdaBoost Algorithm**

Input: \((x_1, y_1), \ldots, (x_m, y_m)\) and a weak learning algorithm.

Let \(P_1(x_i) = \frac{1}{m}\) for all \(i\). i.e., uniform distribution over samples.

For \(t = 1, \ldots, T\)
- Learn a weak classifier \(h_t \in H\) on distribution \(P_t\).
- Let \(\epsilon_t = \Pr_{x_i \sim P_t} [h_t(x_i) \neq y_i]\) and let \(\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)\).
- \(P_{t+1}(x_i) = \frac{P_t(x_i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\)

Output the final hypothesis

\[
h_{final}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Assume that the weak learner return vertical or horizontal half-spaces (that’s the $H$).
Round 1

\[ h_1 \]

\[ \varepsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]
Round 3

$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$
The combined classifier

\[ h_{final} = \text{sign} \left( \begin{array}{c}
0.42 \\
+0.65 \\
+0.92
\end{array} \right) \]
Bounding the Sample Error

Theorem: AdaBoost’s training error

Let $\gamma_t = \frac{1}{2} - \epsilon_t$. For any $T$, $h_{final}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$ has training error

$$err_S(h_{final}) \leq \exp \left( -2 \sum_{t=1}^{T} \gamma_t^2 \right)$$

So, for weak learners where $\gamma_t > \gamma$, and $T = O \left( \frac{1}{\gamma^2} \ln \left( \frac{1}{\epsilon} \right) \right)$ we have

$$err_S(h_{final}) \leq \epsilon.$$

**Ada(ptive)Boost:**

- Adaptive: We don’t need to know $\gamma$ or $T$ before we start.
- Can adapt to $\gamma_t$.
- Automatically better when $\gamma_t \gg \gamma$.
- Practical algorithm.
Generalization Error

We gave a guarantee that the sample error is at most $err_S(H) \leq \epsilon$. What about generalization?

- $h_{final}$ is a combination of $T$ hypothesis $h_1, \ldots, h_T \in H$.
- $h_{final} \notin H$ possibly, but it’s still structured.
- Recall from Homework 3
  → Combination of $T$ hypothesis from $H$ has a bounded Growth function.
  → **Roughly speaking:** This means $h_{final}$ comes from a class of with VC dimension $\tilde{O}(T \text{ VCDim}(H))$.

---

**Theorem: AdaBoost’s true error**

When $S$ has $\tilde{\Omega}\left(\frac{\text{VCDim}(H)}{\gamma^2 \epsilon}\right)$ many samples, then $err_P(h_{final}) \leq \epsilon$. 
Better Generalization Guarantee

Last slide: VC dimension $\tilde{O}(T \text{ VCDim}(H))$
→ Keep $T$ small. As $T$ increases there is a chance of overfitting.

Our first guess!

Actual run of AdaBoost.

Cool theory for proving why AdaBoost doesn’t overfit.
Schapire and Freund also gave online learning algorithms (last lecture).

Connection between boosting and regret minimization

<table>
<thead>
<tr>
<th></th>
<th>$x_1$, $x_2$, $x_3$, ..., $x_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td></td>
</tr>
<tr>
<td>$h_2$</td>
<td></td>
</tr>
<tr>
<td>$h_3$</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$h_{</td>
<td>H</td>
</tr>
</tbody>
</table>

For every distribution $P$ over the columns, there is a row with expected payoff $\geq \frac{1}{2} + \gamma$.

- **Boosting**: Distribution $Q$ over $h_1$, $h_2$, ... that is $\geq \frac{1}{2} + \gamma$ for every $x_i$.
- **Regret minimization** against an adversary who is best responding results in the sequence $h_1$, $h_2$, ...

Optional Material
Ensemble Methods

Meta learning algorithms that call multiple algorithms to improve learning performance.

\[ h_{\text{ensemble}}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

**Boosting:** Take one sample set \( S \), learn \( h_t \) for different weight on these samples. Take \( \alpha_t \)-weighted majority vote.

\( \rightarrow \) Improve training error of the weak classifiers \( h_t \)’s.
Bagging

Even if the training error is already good (bias), can we decrease the variance?

\[ h_{bagging}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

Bagging (Bootstrap Aggregating)

Input: \( S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \) and any learning algorithm.

For \( t = 1, \ldots, T \)

- \( S_t = \) sample with replacement from \( S \).
- \( h_t = \) train on the sample set \( S_t \).

Return \( \text{sign}(\sum_{t=1}^{T} h_t(x)) \)
Enjoy the

Happy Thanksgiving!