Introduction to Machine Learning
236756

Prof. Nir Ailon
Based on original presentation by Prof. Nati Srebro-Bartom

Lecture 8: Boosting
“Weak” vs “Strong” Learning

• Recall definition of (realizable) PAC learning of $\mathcal{H}$ using rule $A(\cdot)$:
  For any $\mathcal{D}$ s.t. $\inf_{h \in \mathcal{H}} L_D(h) = 0$, and any $\epsilon, \delta > 0$, using $m(\epsilon, \delta)$ sample,
  $\forall_{S \sim \mathcal{D}(m(\epsilon, \delta))} L_D(A(S)) < \epsilon$

• $A(\cdot)$ is a **weak learner** for $\mathcal{H}$ if:
  There exists $\epsilon < \frac{1}{2}, \delta < 1, m$, s.t. for any $\mathcal{D}$ with $\inf_{h \in \mathcal{H}} L_D(h) = 0$,
  $\forall_{S \sim \mathcal{D}(m)} L_D(A(S)) < \epsilon$
  (e.g. $\epsilon = 0.49$ and $1 - \delta = 0.01$)

• If $\mathcal{H}$ is weakly learnable, is it also strongly learnable?
  • Yes: $\mathcal{H}$ is weakly learnable $\Rightarrow$ VCdim($\mathcal{H}$)$<\infty$ $\Rightarrow$ $\mathcal{H}$ is (strongly) learnable

• If we have access to an (efficient) weak learner $A(\cdot)$, can we use it to build an (efficient) strong learner?
Example: Weak Learning with a Weak Class

• $\mathcal{X} = \mathbb{R}^2, \mathcal{H} =$ axis aligned rectangles

• Decision stumps: $\mathcal{B} = \left\{ \left[ s \cdot x[i] < \theta \right] | i = 1, 2, \ s = \pm 1, \ \theta \in \mathbb{R} \right\}$

• Claim: For any $\mathcal{D}$, if $\exists h \in \mathcal{H} L_D(h) = 0 \implies \exists h \in \mathcal{B} L_D(h) \leq \frac{3}{7} < 0.429$

• Since $\text{VCdim}(\mathcal{B}) = 3$, with $m = m_{VC}(D = 3, \epsilon = 0.001, \delta = 0.9)$:

  w.p. $\geq 0.1$ over $S \sim \mathcal{D}^m$: $L_D(\text{ERM}_B(S)) < 0.43$

• Conclusion:

  $\text{ERM}_B(\cdot)$ is a weak learner for $\mathcal{H}$ with $\epsilon = 0.43 < 0.5$ and $\delta = 0.9 < 1$
The Boosting Problem

• Boosting the Confidence:
  If the learning algorithm works only with some very small fixed probability \(1 - \delta_0\) (e.g. \(1 - \delta_0 = 0.01\)), can we construct a new algorithm that works with arbitrarily high probability \(1 - \delta\) (for any \(\delta > 0\))?

• Boosting the error:
  If the learning algorithm only returns a predictor that is guaranteed to be slightly better than chance, i.e. has error \(\epsilon_0 = \frac{1}{2} - \gamma < \frac{1}{2}\) (for some fixed \(\gamma > 0\)), can we construct a new algorithm that achieves arbitrarily low error \(\epsilon\)?
The Boosting Problem

• Boosting the Confidence:
  If the learning algorithm works only with some very small fixed probability \(1 - \delta_0\) (e.g. \(1 - \delta_0 = 0.01\)), can we construct a new algorithm that works with arbitrarily high probability \(1 - \delta\) (for any \(\delta > 0\))?
  • For any \(\epsilon > 0\), with \(m_0(\epsilon)\) samples, \(\forall S \sim D^{m_0(\epsilon)} L_D(A(S)) < \epsilon\)

• Boosting the error:
  If the learning algorithm only returns a predictor that is guaranteed to be slightly better than chance, i.e. has error \(\epsilon_0 = \frac{1}{2} - \gamma < \frac{1}{2}\) (for some fixed \(\gamma > 0\)), can we construct a new algorithm that achieves arbitrarily low error \(\epsilon\)?
Boosting the Confidence

• For any $\delta$:

1. For $i=1..k$: 
   \[ k = \frac{\log 2/\delta}{\log 1/\delta_0} \]
   Collect $m_0$ independent samples $S_i$
   \[ h_i = A(S_i) \]

2. Collect $m_{val} = \frac{4 \log(4k/\delta)}{\epsilon^2}$ additional independent samples $S_{val}$

3. Return \( \hat{h} = \arg \min_{h_1,...,h_k} L_{S_{val}}(h_i) \)

• Claim: w.p. $\geq 1 - \delta$, $L(\hat{h}) \leq \epsilon_0 + \epsilon$

• Total samples used: $O \left( m_0(\epsilon_0) \cdot \log \frac{1}{\delta} + \frac{\log \frac{1}{\delta}}{\epsilon^2} \right)$

• E.g. if $\epsilon_0 = \frac{1}{2} - \gamma < 1/2$, take $\epsilon = \frac{\gamma}{2}$ so that $\epsilon_0 + \epsilon = \frac{1}{2} - \frac{\gamma}{2} < \frac{1}{2}$
Boosting the Error

If a learning algorithm only returns a predictor that is guaranteed to be slightly better than chance, i.e. has error $\epsilon_0 = \frac{1}{2} - \gamma < \frac{1}{2}$ (for some $\gamma > 0$), can we construct a new algorithm that achieves arbitrarily low error $\epsilon$?

- Posed (as a theoretical question) by Valiant and Kearns, Harvard 1988
- Solved by MIT student Robert Schapire, 1990
- AdaBoost Algorithm by Schapire and Yoav Fruend, AT&T 1995
AdaBoost

- **Input:** Training set \( S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \)
- **Weak Learner** \( A \), which will be applied to *distributions* \( D \) over \( S \)
  - If thinking of \( A(S') \) as accepting a sample \( S' \):
    each \((x, y) \in S'\) is set to \((x_i, y_i)\) w.p. \( D_i \) (independently and with replacements)
  - Usually easier to think of \( A \) as operating on a weighted sample, with weights \( D_i \)
- **Output:** hypothesis \( h \) with arbitrarily small \( L_S(h) \)
  - We’ll worry about \( L_D(h) \), which is what we really care about, later

**Initialize** \( D^{(1)} = \left( \frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m} \right) \)

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

- \( h_t = A(D^{(t)}) \)
- \( \epsilon_t = L_{D^{(t)}}(h_t) = \frac{1}{m} \sum_i D_i^{(t)} \cdot [h_t(x_i) \neq y_i] \)
- \( w_t = \frac{1}{2} \log \left( \frac{1}{\epsilon_t} - 1 \right) \)
- \( D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(x_j))} \)

**Output:** \( h_S(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x)) \)
AdaBoost: Weight Update

Initialize $D^{(1)} = \left( \frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m} \right)$

For $t=1, \ldots, T$:

- $h_t = A(D^{(t)})$
- $\epsilon_t = L_{D^{(t)}}(h_t) = \frac{1}{m}\sum_i D_i^{(t)} \cdot [h_t(x_i) \neq y_i]$
- $w_t = \frac{1}{2}\log \left( \frac{1}{\epsilon_t} - 1 \right)$
- $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_i y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp(-w_j y_j h_t(x_j))}$

Output: $h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x))$

- Increase weight on errors, decrease on correct points:
  
  $D_i^{(t+1)} \propto \begin{cases} 
  D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\
  D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i 
  \end{cases}$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$
AdaBoost: Minimizing $L_S(h)$

Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right)$

For $t=1, \ldots, T$:

- $h_t = A(D^{(t)})$
- $\epsilon_t = L_{D^{(t)}}(h_t) = \frac{1}{m} \sum_i D_i^{(t)} \cdot \left[ [h_t(x_i) \neq y_i] \right]$
- $w_t = \frac{1}{2} \log \left( \frac{1}{\epsilon_t} - 1 \right)$
- $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(x_j))}$

Output: $h_S(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x))$

• Theorem: If $\forall_t \epsilon_t \leq \frac{1}{2} - \gamma$, then after $T$ rounds $L_S(h_S) \leq e^{-2\gamma^2 T}$

  If $A(\cdot)$ is a weak learner for $\mathcal{H}$ with params $\delta_0, \epsilon_0 = 1 - \gamma$, and $L_D(h) = 0$

  $\Rightarrow L_S(h) = 0 \Rightarrow L_{D^{(t)}}(h) = 0 \Rightarrow \text{w.p. } 1 - \delta_0, L_{D^{(t)}}(h) \leq \frac{1}{2} - \gamma$

  $\Rightarrow \text{w.p. } 1 - T\delta_0, L_S(h_S) \leq e^{-2\gamma^2 T}$
AdaBoost: Minimizing $L_S(h)$

• Theorem: If $\forall_t \epsilon_t \leq \frac{1}{2} - \gamma$, then after $T$ rounds $L_S(h_S) \leq e^{-2\gamma^2 T}$
  
  If $A(\cdot)$ is a weak learner for $\mathcal{H}$ with params $\delta_0, \epsilon_0 = 1 - \gamma$, and $L_D(h) = 0$
  
  $\Rightarrow L_S(h) = 0 \Rightarrow L_D(\epsilon)(h) = 0 \Rightarrow$ w.p. $1 - \delta$, $L_D(\epsilon)(h) \leq \frac{1}{2} - \gamma$
  
  $\Rightarrow$ w.p. $1 - \delta T$, $L_S(h_S) \leq e^{-2\gamma^2 T}$

• To get any $\epsilon > 0$, run AdaBoost for $T = \frac{\log\left(\frac{1}{\epsilon}\right)}{2\gamma^2}$ rounds

• Setting $\epsilon = \frac{1}{2m}$, after $T = \frac{\log(2m)}{2\gamma^2}$ rounds: $L_S(h_S) = 0$ !

• What about $L_D(h)$?
AdaBoost as Learning a Linear Classifier

• Recall: \( h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x)) \)

• Let \( \mathcal{B} = \{ \text{all hypothesis outputed by } A \} \)
  • “Base Class”, e.g. decision stumps
  \[ h_s \in \{ h_w(x) = \text{sign}(\langle w, \phi(x) \rangle) \mid w \in \mathbb{R}^B \} \]

• AdaBoost can be interpreted as an optimization method for learning a linear predictor
  • Feature space implicitly specified by weak learner
  • “weak learner” \( \approx \) find a coordinate of \( w \) that it will be good to increase

• What about generalization?
  • Even if \( \text{VCdim}(\mathcal{B}) \) is small, \( \text{VCdim}(\mathcal{L}(\mathcal{B})) \) can be exponentially larger
  • E.g. with \(|\mathcal{B}| = 2^r\), \( \text{VCdim}(\mathcal{B}) \leq r \) but can have \( \text{VCdim}(\mathcal{L}(\mathcal{B})) = 2^r \)
Sparse Linear Classifiers

- Recall: \( h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x)) \)
- Let \( \mathcal{B} = \{ \text{all hypothesis outputed by } A \} \)
  - “Base Class”, e.g. decision stumps
    \[ \phi_h(x) = h(x) \]
    \[ h_s \in \{ h_w(x) = \text{sign}(\langle w, \phi(x) \rangle) \mid w \in \mathbb{R}^{\mathcal{B}}, \|w\|_0 \leq T \} \]
    \[ w_h = \sum_{h_t=h} w_t \]
    Class of sparse halfspaces \( \mathcal{L}(\mathcal{B}, T) \)
- We already saw: \( \text{VCdim}(\mathcal{L}(\mathcal{B}, T)) \leq O(T \log|\mathcal{B}|) \)
- Even if \( \mathcal{B} \) is infinite (e.g. in the case of decision stumps): \( \text{VCdim}(\mathcal{L}(\mathcal{B}, T)) \leq \tilde{O}(T \cdot \text{VCdim}(\mathcal{B})) \)
- Number of rounds \( T \) (= sparseness) is complexity control
Complexity Control

• After $T$ iterations of AdaBoost:

$$h_s \in \left\{ h_w(x) = \text{sign}(\langle w, \phi(x) \rangle) \mid w \in \mathbb{R}^B, \|w\|_0 \leq T \right\}$$

Class of sparse halfspaces $L(\mathcal{B}, T)$

• Even with a relatively “weak” base class, with high enough $T$, can get very complex $h_s$.
  • E.g., with decision stumps over $\mathbb{R}$, can get any piecewise constant function, i.e. approximate any function arbitrarily well

• Want low $T$ so that we can generalize

• Realizable case: use first $T$ s.t. $L_S(h_s) = 0$
  • We know this will happen with $T = O \left( \frac{\log(m)}{\gamma^2} \right)$

• More realistically: Use validation/cross-validation to select $T$
  • “Early Stopping” as regularization: we could continue optimizing and lower $L_S(h)$, but stopping the optimization early has a regularization effect
AdaBoost: Beyond Zero Training Error

\[ h^{(T)}(x) = \sum_{t=1}^{T} w_t h_t(x) \]

- Even after \( L^0_1 \left( h^{(T)}(x) \right) = 0 \), AdaBoost keeps improving the margin, and hence generalization.
- But it’s a \( \ell_1 \) margin and not \( \ell_2 \) margin as in SVM...
Example: Viola-Jones Face Detector

• Classify each square in an image as “face” or “no-face”

• We’ll consider all squares in an image, at many scales, of size at least 24x24 original pixels, and represent them as 24x24 grayscale pixels.

• $\mathcal{X} = \text{patches of 24x24 grayscale pixels}$
Viola-Jones “Weak Predictors”/Features

\[ \mathcal{B} = \left\{ \left[ g_{r,t}(x) < \theta \right] \mid \theta \in \mathbb{R}, \text{rect} \ r \text{ in image}, t \in \{ A, B, C, D, \bar{A}, \bar{B}, \bar{C}, \bar{D} \} \right\} \]

where \( g_{r,t}(x) = \text{sum of “blue” pixels} - \text{sum of “red” pixels} \)

First two weak predictors \( h_1, h_2 \) selected in original Viola-Jones implementation:
Viola-Jones Face Detector

• Simple implementation of boosting using generic (non-face specific) “weak learners”/features
  • Can be used also for detecting other objects
• Efficient method using dynamic programing and caching to find good weak predictor
• About 1 million possible $g_{r,t}$, but only very few used in returned predictor
• Sparsity:
  ➔ Generalization
  ➔ Prediction speed! (and small memory footprint)
• To run in real-time (on 2001 laptop), use sequential evaluation
  • First evaluate first few $h_t$ to get rough prediction
  • Only evaluate additional $h_t$ on patches where the leading ones are promising

(and clever engineering in evaluation the required filters on all the patches in all scales....)
Boosting (AdaBoost)

- “Boosting” weak learning to get arbitrary small error
  - Theory is for realizable case
  - Improper learning
- Ensemble method for combining many simpler predictors
  - E.g. combining decision stumps or decision trees
  - Other ensemble methods: bagging, averaging, gating networks
- Method for learning using \textit{sparse} linear predictors with large (infinite?) dimensional feature space
  - Sparsity controls complexity \textit{and} speed
  - Number of iterations controls sparsity $\rightarrow$ early stopping as regularization
- Learning (in high dimensions) with large $\ell_1$ margin
  - Learning guarantee in terms of $\ell_1$ margin
  - Varying \#iter $T$ explores regularization path (balances of loss and norm)