AdaBoost

Jiri Matas and Jan Šochman

Centre for Machine Perception
Czech Technical University, Prague
http://cmp.felk.cvut.cz
Outline:

- AdaBoost algorithm
  - Why is of interest?
  - How it works?
  - Why it works?
- AdaBoost variants
- AdaBoost with a Totally Corrective Step (TCS)
- Experiments with a Totally Corrective Step
Introduction

- 1990 – Boost-by-majority algorithm (Freund)
- 1995 – AdaBoost (Freund & Schapire)
- 1997 – Generalized version of AdaBoost (Schapire & Singer)
- 2001 – AdaBoost in Face Detection (Viola & Jones)

Interesting properties:

- AB is a linear classifier with all its desirable properties.
- AB output converges to the logarithm of likelihood ratio.
- AB has good generalization properties.
- AB is a feature selector with a principled strategy (minimisation of upper bound on empirical error).
- AB close to sequential decision making (it produces a sequence of gradually more complex classifiers).
AdaBoost is an algorithm for constructing a "strong" classifier as linear combination

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

of "simple" "weak" classifiers \( h_t(x) \).
AdaBoost is an algorithm for constructing a "strong" classifier as linear combination

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

of "simple" "weak" classifiers \( h_t(x) \).

**Terminology**

- \( h_t(x) \) ... “weak” or basis classifier, hypothesis, "feature"
- \( H(x) = \text{sign}(f(x)) \) ... “strong” or final classifier/hypothesis
What is AdaBoost?

- AdaBoost is an algorithm for constructing a "strong" classifier as linear combination
  \[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]
  of "simple" "weak" classifiers \( h_t(x) \).

Terminology

- \( h_t(x) \) ... "weak" or basis classifier, hypothesis, "feature"
- \( H(x) = sign(f(x)) \) ... "strong" or final classifier/hypothesis

Comments

- The \( h_t(x) \)'s can be thought of as features.
- Often (typically) the set \( \mathcal{H} = \{ h(x) \} \) is infinite.
Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, 1\}\)

Initialize weights \(D_1(i) = 1/m\)

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

1. (Call \textit{WeakLearn}), which returns the weak classifier \(h_t : \mathcal{X} \rightarrow \{-1, 1\}\) with minimum error w.r.t. distribution \(D_t\);
2. Choose \(\alpha_t \in \mathbb{R}\),
3. Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is a normalization factor chosen so that \(D_{t+1}\) is a distribution

Output the strong classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
(Discrete) AdaBoost Algorithm – Singer & Schapire (1997)

Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, 1\}\)

Initialize weights \(D_1(i) = 1/m\)

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

1. (Call \textit{WeakLearn}), which returns the weak classifier \(h_t : \mathcal{X} \rightarrow \{-1, 1\}\) with minimum error w.r.t. distribution \(D_t\);
2. Choose \(\alpha_t \in \mathbb{R}\),
3. Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is a normalization factor chosen so that \(D_{t+1}\) is a distribution

Output the strong classifier:

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

Comments

- The computational complexity of selecting \(h_t\) is independent of \(t\).
- All information about previously selected “features” is captured in \(D_t\)!
WeakLearn

Loop step: Call WeakLearn, given distribution $D_t$; returns weak classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)

- WeakLearn examples:
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ infinite
  - Selecting the best one from given finite set $\mathcal{H}$
Loop step: Call *WeakLearn*, given distribution $D_t$; returns weak classifier $h_t : \mathcal{X} \to \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  $$h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [y_i \neq h_j(x_i)]$$

- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)

- *WeakLearn* examples:
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ *infinite*
  - Selecting the best one from given *finite* set $\mathcal{H}$

Demonstration example

Weak classifier = perceptron

\[\bullet \sim \mathcal{N}(0, 1) \quad \bullet \sim \frac{1}{r\sqrt{8\pi^3}}e^{-1/2(r-4)^2}\]
Loop step: Call WeakLearn, given distribution $D_t$; returns weak classifier $h_t: \mathcal{X} \rightarrow \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  
  $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)

- WeakLearn examples:
  
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ infinite
  
  - Selecting the best one from given finite set $\mathcal{H}$

Demonstration example

Training set

Weak classifier = perceptron

\[ \sim N(0, 1) \quad \sim \frac{1}{r\sqrt{8\pi^3}}e^{-1/2(r-4)^2} \]
Loop step: Call WeakLearn, given distribution $D_t$; returns weak classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  
  $$h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)

- WeakLearn examples:
  
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ infinite
  - Selecting the best one from given finite set $\mathcal{H}$

Demonstration example

Training set

Weak classifier = perceptron

\[\sim N(0, 1) \quad \sim \frac{1}{r\sqrt{8\pi^3}}e^{-1/2(r-4)^2}\]
The main objective is to minimize $\varepsilon_{tr} = \frac{1}{m} |\{i : H(x_i) \neq y_i\}|$

It can be upper bounded by $\varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t$
AdaBoost as a Minimiser of an Upper Bound on the Empirical Error

- The main objective is to minimize $\epsilon_{tr} = \frac{1}{m} \left| \left\{ i : H(x_i) \neq y_i \right\} \right|$.

- It can be upper bounded by $\epsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t$.

How to set $\alpha_t$?

- Select $\alpha_t$ to greedily minimize $Z_t(\alpha)$ in each step.

- $Z_t(\alpha)$ is convex differentiable function with one extremum.

  $\Rightarrow h_t(x) \in \{-1, 1\} \text{ then optimal } \alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$

  where $r_t = \sum_{i=1}^{m} D_t(i) h_t(x_i) y_i$.

- $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 1 \text{ for optimal } \alpha_t$.

  $\Rightarrow$ Justification of selection of $h_t$ according to $\epsilon_t$. 

AdaBoost as a Minimiser of an Upper Bound on the Empirical Error

- The main objective is to minimize 
  \[ \varepsilon_{tr} = \frac{1}{m} |\{ i : H(x_i) \neq y_i \}| \]

- It can be upper bounded by 
  \[ \varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t \]

How to set \( \alpha_t \)?

- Select \( \alpha_t \) to greedily minimize \( Z_t(\alpha) \) in each step

- \( Z_t(\alpha) \) is convex differentiable function with one extremum
  \[ h_t(x) \in \{-1, 1\} \text{ then optimal } \alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t}) \]
  where \( r_t = \sum_{i=1}^{m} D_t(i) h_t(x_i) y_i \)

- \( Z_t = 2 \sqrt{\varepsilon_t(1-\varepsilon_t)} \leq 1 \) for optimal \( \alpha_t \)
  \[ \Rightarrow \text{Justification of selection of } h_t \text{ according to } \varepsilon_t \]

Comments

- The process of selecting \( \alpha_t \) and \( h_t(x) \) can be interpreted as a single optimization step minimising the upper bound on the empirical error. Improvement of the bound is guaranteed, provided that \( \varepsilon_t < 1/2 \).

- The process can be interpreted as a component-wise local optimization (Gauss-Southwell iteration) in the (possibly infinite dimensional!) space of \( \bar{\alpha} = (\alpha_1, \alpha_2, \ldots) \) starting from. \( \bar{\alpha}_0 = (0, 0, \ldots) \).
Effect on the training set

Reweighting formula:

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}
\]

\[
\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} 
< 1, & y_i = h_t(x_i) \\
> 1, & y_i \neq h_t(x_i) 
\end{cases}
\]

\[\Rightarrow\] Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!
Reweighting

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q} \]

\[
exp(-\alpha_t y_i h_t(x_i)) \begin{cases} 
< 1, & y_i = h_t(x_i) \\ 
> 1, & y_i \neq h_t(x_i) 
\end{cases}
\]

\[ \Rightarrow \] Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!
Reweighting

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q} \]

\[ \exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases} \]

⇒ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!
Reweighting

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q} \]

\[ \exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases} \]

\[ \Rightarrow \] Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!
Reweighting

Effect on the training set

Reweighting formula:

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}
\]

\[
\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} 
< 1, & y_i = h_t(x_i) \\
> 1, & y_i \neq h_t(x_i)
\end{cases}
\]

⇒ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!
Reweighting

Effect on the training set

Reweighting formula:

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q} \]

\[ \exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases} \]

⇒ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example!

Effect on \( h_t \)

- \( \alpha_t \) minimize \( Z_t \) ⇒

\[ \sum_{i: h_t(x_i) = y_i} D_{t+1}(i) = \sum_{i: h_t(x_i) \neq y_i} D_{t+1}(i) \]

- Error of \( h_t \) on \( D_{t+1} \) is 1/2

- Next weak classifier is the most “independent” one
Summary of the Algorithm
Summary of the Algorithm

Initialization...
Summary of the Algorithm

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

* Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
Summary of the Algorithm

Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- If $\epsilon_t \geq 1/2$ then stop
Summary of the Algorithm

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

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$
Summary of the Algorithm

Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$
- Update
  
  $$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$t = 1$
Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Summary of the Algorithm

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

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Summary of the Algorithm

Initialization...

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

- Find \( h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)] \)
- If \( \epsilon_t \geq 1/2 \) then stop
- Set \( \alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right) \)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Summary of the Algorithm

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

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- If $\epsilon_t \geq 1/2$ then stop

- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$

- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Summary of the Algorithm

Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Summary of the Algorithm

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

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$
- Update

\[ D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \]

Output the final classifier:

\[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]
Summary of the Algorithm

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

- Find $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Summary of the Algorithm

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

- **Find** $h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- **If** $\epsilon_t \geq 1/2$ then stop

- **Set** $\alpha_t = \frac{1}{2} \log \left( \frac{1+r_t}{1-r_t} \right)$

- **Update**

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Does AdaBoost generalize?

Margins in SVM

$$\max_{(x,y) \in S} \min \frac{y(\vec{\alpha} \cdot \vec{h}(x))}{\|\vec{\alpha}\|_2}$$

Margins in AdaBoost

$$\max_{(x,y) \in S} \min \frac{y(\vec{\alpha} \cdot \vec{h}(x))}{\|\vec{\alpha}\|_1}$$

Maximizing margins in AdaBoost

$$P_S[yf(x) \leq \theta] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta}(1 - \epsilon_t)^{1+\theta}}$$

where $$f(x) = \frac{\vec{\alpha} \cdot \vec{h}(x)}{\|\vec{\alpha}\|_1}$$

Upper bounds based on margin

$$P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + O \left( \frac{1}{\sqrt{m}} \left( \frac{d \log^2(m/d)}{\theta^2} + \log(1/\delta) \right)^{1/2} \right)$$
AdaBoost variants

Freund & Schapire 1995

- Discrete ($h : \mathcal{X} \rightarrow \{0, 1\}$)
- Multiclass AdaBoost.M1 ($h : \mathcal{X} \rightarrow \{0, 1, \ldots, k\}$)
- Multiclass AdaBoost.M2 ($h : \mathcal{X} \rightarrow [0, 1]^k$)
- Real valued AdaBoost.R ($Y = [0, 1], h : \mathcal{X} \rightarrow [0, 1]$)

Schapire & Singer 1997

- Confidence rated prediction ($h : \mathcal{X} \rightarrow R$, two-class)
- Multilabel AdaBoost.MR, AdaBoost.MH (different formulation of minimized loss)

... Many other modifications since then (Totally Corrective AB, Cascaded AB)
Pros and cons of AdaBoost

Advantages

- Very simple to implement
- Feature selection on very large sets of features
- Fairly good generalization

Disadvantages

- Suboptimal solution for $\bar{\alpha}$
- Can overfit in presence of noise
Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, 1\}\)
Initialize weights \(D_1(i) = 1/m\)

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

1. (Call \textit{WeakLearn}), which returns the weak classifier \(h_t : \mathcal{X} \rightarrow \{-1, 1\}\) with minimum error w.r.t. distribution \(D_t\);
2. Choose \(\alpha_t \in \mathbb{R}\),
3. Update \(D_{t+1}\)
4. (Call \textit{WeakLearn}) on the set of \(h_m\)'s with non zero \(\alpha\)'s . Update \(\alpha\).
   Update \(D_{t+1}\). Repeat till \(|\epsilon_t - 1/2| < \delta, \forall t\).

Comments

- All weak classifiers have \(\epsilon_t \approx 1/2\), therefore the classifier selected at \(t + 1\) is "independent" of all classifiers selected so far.
- It can be easily shown, that the totally corrective step reduces the upper bound on the empirical error without increasing classifier complexity.
- The TCA was first proposed by Kivinen and Warmuth, but their \(\alpha_t\) is set as in standard Adaboost.
- Generalization of TCA is an open question.
Experiments with TCA on the IDA Database

- Discrete AdaBoost, Real AdaBoost, and Discrete and Real TCA evaluated
- Weak learner: stumps.
- Data from the IDA repository (Ratsch:2000):

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Input dimension</th>
<th>Training patterns</th>
<th>Testing patterns</th>
<th>Number of realizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana</td>
<td>2</td>
<td>400</td>
<td>4900</td>
<td>100</td>
</tr>
<tr>
<td>Breast cancer</td>
<td>9</td>
<td>200</td>
<td>77</td>
<td>100</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>468</td>
<td>300</td>
<td>100</td>
</tr>
<tr>
<td>German</td>
<td>20</td>
<td>700</td>
<td>300</td>
<td>100</td>
</tr>
<tr>
<td>Heart</td>
<td>13</td>
<td>170</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Image segment</td>
<td>18</td>
<td>1300</td>
<td>1010</td>
<td>20</td>
</tr>
<tr>
<td>Ringnorm</td>
<td>20</td>
<td>400</td>
<td>7000</td>
<td>100</td>
</tr>
<tr>
<td>Flare solar</td>
<td>9</td>
<td>666</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>Splice</td>
<td>60</td>
<td>1000</td>
<td>2175</td>
<td>20</td>
</tr>
<tr>
<td>Thyroid</td>
<td>5</td>
<td>140</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>Titanic</td>
<td>3</td>
<td>150</td>
<td>2051</td>
<td>100</td>
</tr>
<tr>
<td>Twonorm</td>
<td>20</td>
<td>400</td>
<td>7000</td>
<td>100</td>
</tr>
<tr>
<td>Waveform</td>
<td>21</td>
<td>400</td>
<td>4600</td>
<td>100</td>
</tr>
</tbody>
</table>

- Note that the training sets are fairly small
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- **Discrete AdaBoost (blue)**, **Real AdaBoost (green)**,
- **Discrete AdaBoost with TCA (red)**, **Real AdaBoost with TCA (cyan)**
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- the black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Results with TCA on the IDA Database

- Training error (dashed line), test error (solid line)
- Discrete AdaBoost (blue), Real AdaBoost (green),
- Discrete AdaBoost with TCA (red), Real AdaBoost with TCA (cyan)
- The black horizontal line: the error of AdaBoost with RBF network weak classifiers from (Ratsch-ML:2000)
Conclusions

- The AdaBoost algorithm was presented and analysed.
- A modification of the Totally Corrective AdaBoost was introduced.
- Initial test show that the TCA outperforms AB on some standard data sets.
HEART

Length of the strong classifier

0.05 0.1 0.15 0.2 0.25 0.3 0.35

0 0.05 0.1 0.15 0.2 0.25 0.3 0.35

Length of the strong classifier
Length of the strong classifier