AdaBoost

Lecturer:
Jan Šochman

Authors:
Jan Šochman, Jiří Matas

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

Motivation

AdaBoost with trees is the best off-the-shelf classifier in the world. (Breiman 1998)

Outline:

- AdaBoost algorithm
  - How it works?
  - Why it works?
- Online AdaBoost and other variants
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 \( h_t(x) : \mathcal{X} \rightarrow \{-1, +1\} \).

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

Interesting properties
- AB is capable reducing both bias (e.g. stumps) and variance (e.g. trees) of the weak classifiers
- AB has good generalisation properties (maximises margin)
- AB output converges to the logarithm of likelihood ratio
- AB can be seen as a feature selector with a principled strategy (minimisation of upper bound on empirical error)
- AB is close to sequential decision making (it produces a sequence of gradually more complex classifiers)
The AdaBoost Algorithm

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

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

Initialise weights \(D_1(i) = 1/m\)
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T\):
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

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

\(t = 1\)
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) \mathbb{1}[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
The AdaBoost Algorithm

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

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

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

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

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

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

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 \(\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)

- Update

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

where \(Z_t\) is normalisation factor
The AdaBoost Algorithm

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

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

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 \(\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\)
- Update
  \[D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\]
  where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

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

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_1^m D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left( \frac{1-\epsilon_t}{\epsilon_t} \right)\)
- Update

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

 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 \(\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)

 - Update

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})\)
- Update

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
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 \(\alpha_t = \frac{1}{2} \log \left( \frac{1-\epsilon_t}{\epsilon_t} \right)\)
- Update
  \[
  D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]
  where \(Z_t\) is normalisation factor

Output the final classifier:

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

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

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

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

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

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

where \(Z_t\) is normalisation factor

Output the final classifier:

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

Effect on the training set

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

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

⇒ Increase (decrease) weight of wrongly (correctly) classified examples
⇒ The weight is the upper bound on the error of a given example
⇒ All information about previously selected “features” is captured in \( D_t \)
Reweighting

Effect on the training set

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

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

⇒ Increase (decrease) weight of wrongly (correctly) classified examples

⇒ The weight is the upper bound on the error of a given example

⇒ All information about previously selected “features” is captured in \( D_t \)
Reweighting

Effect on the training set

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

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

⇒ Increase (decrease) weight of wrongly (correctly) classified examples
⇒ The weight is the upper bound on the error of a given example
⇒ All information about previously selected “features” is captured in \( D_t \)
Upper Bound Theorem

**Theorem:** The following upper bound holds on the training error of $H$

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leq \prod_{t=1}^{T} Z_t$$

**Proof:** By unravelling the update rule

$$D_{T+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i))}{m \prod_t Z_t} = \frac{\exp(-y_i f(x_i))}{m \prod_t Z_t}$$

If $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $\exp(-y_i f(x_i)) > 1$, thus

$$\prod_{i} [H(x_i) \neq y_i] \leq \exp(-y_i f(x_i))$$

$$\frac{1}{m} \sum_{i} [H(x_i) \neq y_i] \leq \frac{1}{m} \sum_{i} \exp(-y_i f(x_i))$$

$$= \sum_{i} (\prod_{t} Z_t) D_{T+1}(i) = \prod_{t} Z_t$$
Consequences of the Theorem

- Instead of minimising the training error, its upper bound can be minimised
- This can be done by minimising $Z_t$ in each training round by:
  - Choosing optimal $h_t$, and
  - Finding optimal $\alpha_t$
- AdaBoost can be proved to maximise margin
- AdaBoost iteratively fits an additive logistic regression model
Choosing $\alpha_t$

We attempt to minimise $Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$:

$$\frac{dZ}{d\alpha} = -\sum_{i=1}^{m} D(i) y_i h(x_i) e^{-y_i \alpha h(x_i)} = 0$$

$$- \sum_{i:y_i=h(x_i)} D(i) e^{-\alpha} + \sum_{i:y_i\neq h(x_i)} D(i) e^{\alpha} = 0$$

$$-e^{-\alpha}(1 - \epsilon) + e^{\alpha} \epsilon = 0$$

$$\alpha = \frac{1}{2} \log \frac{1 - \epsilon}{\epsilon}$$

$\Rightarrow$ The minimisator of the upper bound is $\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$
Choosing $h_t$

Weak classifier examples

- Decision tree (or stump), Perceptron – $\mathcal{H}$ infinite
- Selecting the best one from given finite set $\mathcal{H}$

Justification of the weighted error minimisation

Having $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$

$$Z_t = \sum_{i=1}^{m} D_t(i) e^{-y_i\alpha_t h_t(x_i)}$$

$$= \sum_{i:y_i=h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t}$$

$$= (1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$$

$$= 2 \sqrt{\epsilon_t (1 - \epsilon_t)}$$

$\Rightarrow$ $Z_t$ is minimised by selecting $h_t$ with minimal weighted error $\epsilon_t$
Generalisation (Schapire & Singer 1999)

Maximising margins in AdaBoost

\[
P_{(x,y)\sim S}[y f(x) \leq \theta] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta} (1 - \epsilon_t)^{1+\theta}} \quad \text{where } f(x) = \frac{\vec{\alpha} \cdot \vec{h}(x)}{\|\vec{\alpha}\|_1}
\]

- Choosing \( h_t(x) \) with minimal \( \epsilon_t \) in each step one minimises the margin
- Margin in SVM use the \( L_2 \) norm instead: \( (\vec{\alpha} \cdot \vec{h}(x))/\|\vec{\alpha}\|_2 \)

Upper bounds based on margin

With probability \( 1 - \delta \) over the random choice of the training set \( S \)

\[
P_{(x,y)\sim \mathcal{D}}[y f(x) \leq 0] \leq P_{(x,y)\sim S}[y f(x) \leq \theta] + \mathcal{O}\left( \frac{1}{\sqrt{m}} \left( \frac{d \log^2(m/d)}{\theta^2} + \log(1/\delta) \right)^{1/2} \right)
\]

where \( \mathcal{D} \) is a distribution over \( \mathcal{X} \times \{+1, -1\} \), and \( d \) is pseudodimension of \( \mathcal{H} \).

Problem: The upper bound is very loose. In practice AdaBoost works much better.
Convergence (Friedman et al. 1998)

**Proposition 1** The discrete AdaBoost algorithm minimises $J(f(x)) = E(e^{-yf(x)})$ by adaptive Newton updates.

**Lemma** $J(f(x))$ is minimised at

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)}$$

Hence

$$P(y = 1|x) = \frac{e^{f(x)}}{e^{-f(x)} + e^{f(x)}}$$

and

$$P(y = -1|x) = \frac{e^{-f(x)}}{e^{-f(x)} + e^{f(x)}}$$

**Additive logistic regression model**

$$\sum_{t=1}^{T} a_t(x) = \log \frac{P(y = 1|x)}{P(y = -1|x)}$$

**Proposition 2** By minimising $J(f(x))$ the discrete AdaBoost fits (up to a factor 2) an additive logistic regression model.
The Algorithm Recapitulation

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

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
The Algorithm Recapitulation

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

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

For \(t = 1, \ldots, T:\)
The Algorithm Recapitulation

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

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

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

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

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

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

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

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

- If \(\epsilon_t \geq 1/2\) then stop
The Algorithm Recapitulation

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

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

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 \(\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\)
The Algorithm Recapitulation

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

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

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

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

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

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

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

Output the final classifier:

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

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

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

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

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

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update

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

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})\)
- Update \(D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\)

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})\)
- Update

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

Output the final classifier:

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

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

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

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

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

Output the final classifier:

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

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

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

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 \(\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\)
- Update
\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

Output the final classifier:
\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
The Algorithm Recapitulation

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

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

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

Output the final classifier:

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

![Graph showing training error](image-url)
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 1999

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

Oza 2001

- Online AdaBoost

Many other modifications since then: cascaded AB, WaldBoost, probabilistic boosting tree, ...
Online AdaBoost

### Offline

**Given:**
- Set of labeled training samples
  \[ \mathcal{X} = \{(x_1, y_1), ..., (x_m, y_m)| y = \pm 1\} \]
- Weight distribution over \( \mathcal{X} \)
  \[ D_0 = 1/m \]

**For** \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \[ h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \]
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

**Output:**
\[ F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \]

---

### Online

**Given:**

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

**Output:**
\[ F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \]
## Online AdaBoost

### Offline

**Given:**
- Set of labeled training samples
  \[ \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m) \mid y = \pm 1\} \]
- Weight distribution over \( \mathcal{X} \)
  \[ D_0 = \frac{1}{m} \]

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \[ h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \]
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

**Output:**
\[ F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \]

### Online

**Given:**
- One labeled training sample
  \( (x, y) \mid y = \pm 1 \)
- Strong classifier to update

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

**Output:**
\[ F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \]
**Online AdaBoost**

**Offline**

Given:
- Set of labeled training samples
  \[ \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m)|y = \pm 1\} \]
- Weight distribution over \( \mathcal{X} \)
  \[ D_0 = \frac{1}{m} \]

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \[ h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \]
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

Output:
\[ F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \]

**Online**

Given:
- One labeled training sample
  \( (x, y)|y = \pm 1 \)
- Strong classifier to update
- Initial importance \( \lambda = 1 \)

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

Output:
\[ F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \]
Online AdaBoost

**Offline**

Given:
- Set of labeled training samples
  \( \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m) | y = \pm 1\} \)
- Weight distribution over \( \mathcal{X} \)
  \( D_0 = 1/m \)

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \( h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \)
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

Output:
\[
F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))
\]

**Online**

Given:
- **One** labeled training sample
  \( (x, y) | y = \pm 1 \)
- Strong classifier to update
- Initial importance \( \lambda = 1 \)

For \( t = 1, \ldots, T \)
- Update the weak classifier using the sample and the importance
  \( h_t(x) = \mathcal{L}(h_t, (x, y), \lambda) \)

Output:
\[
F(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))
\]
Online AdaBoost

Offline

Given:
- Set of labeled training samples \( \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m)| y = \pm 1\} \)
- Weight distribution over \( \mathcal{X} \)
  \( D_0 = 1/m \)

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \( h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \)
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

Output:
\[
F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))
\]

Online

Given:
- One labeled training sample \( (x, y)| y = \pm 1 \)
- Strong classifier to update
- Initial importance \( \lambda = 1 \)

For \( t = 1, \ldots, T \)
- Update the weak classifier using the sample and the importance
  \( h_t(x) = \mathcal{L}(h_t, (x, y), \lambda) \)
- Update error estimation \( \epsilon_t \)
- Update weight \( \alpha_t \) based on \( \epsilon_t \)

Output:
\[
F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))
\]
Online AdaBoost

**Offline**

Given:
- Set of labeled training samples
  \( \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m) | y = \pm 1 \} \)
- Weight distribution over \( \mathcal{X} \)
  \( D_0 = \frac{1}{m} \)

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \( h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \)
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

Output:
\[
F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))
\]

**Online**

Given:
- One labeled training sample
  \( (x, y) | y = \pm 1 \)
- Strong classifier to update
- Initial importance \( \lambda = 1 \)

For \( t = 1, \ldots, T \)
- Update the weak classifier using the sample and the importance
  \( h_t(x) = \mathcal{L}(h_t, (x, y), \lambda) \)
- Update error estimation \( \epsilon_t \)
- Update weight \( \alpha_t \) based on \( \epsilon_t \)
- Update importance weight \( \lambda \)

Output:
\[
F(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))
\]
Online AdaBoost

Offline

Given:
- Set of labeled training samples \( \mathcal{X} = \{(x_1, y_1), \ldots, (x_m, y_m) | y = \pm 1\} \)
- Weight distribution over \( \mathcal{X} \)
  \( D_0 = 1/m \)

For \( t = 1, \ldots, T \)
- Train a weak classifier using samples and weight distribution
  \( h_t(x) = \mathcal{L}(\mathcal{X}, D_{t-1}) \)
- Calculate error \( \epsilon_t \)
- Calculate coefficient \( \alpha_t \) from \( \epsilon_t \)
- Update weight distribution \( D_t \)

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

Online

Given:
- One labeled training sample \( (x, y) | y = \pm 1 \)
- Strong classifier to update
- Initial importance \( \lambda = 1 \)

For \( t = 1, \ldots, T \)
- Update the weak classifier using the sample and the importance
  \( h_t(x) = \mathcal{L}(h_t, (x, y), \lambda) \)
- Update error estimation \( \epsilon_t \)
- Update weight \( \alpha_t \) based on \( \epsilon_t \)
- Update importance weight \( \lambda \)

Output:
\[
F(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Online AdaBoost

Converges to offline results given the same training set and the number of iterations $N \rightarrow \infty$

Pros and Cons of AdaBoost

Advantages

- Very simple to implement
- General learning scheme - can be used for various learning tasks
- Feature selection on very large sets of features
- Good generalisation
- Seems not to overfit in practice (probably due to margin maximisation)

Disadvantages

- Suboptimal solution (greedy learning)
Selected references


- http://www.boosting.org
Selected references


- http://www.boosting.org

Thank you for attention