Advanced AdaBoost.

Variant of AdaBoost (Viola & Jones)

Strong classifier: \[ H_n(x) = \frac{1}{Z_n} \sum_{\mu=1}^{M} \left( \alpha_{\mu} h_{\mu}(x) + \beta_{\mu} \right) \]

\( h_{\mu}(x) \) - weak classifier.

Modify the update rule:

\[ D_{t+1}(i) = \frac{1}{Z^*_t} D_t(i) e^{-\omega_i \left( \alpha_{t} h_{t}(x) + \beta_{t} \right)} \]

Let \( \omega_{pq} \) be the sum of the weights of the weak class is \( p \) and the true class is \( q \).

Pick weak classifier to minimize

\[ 2 \left( \frac{1}{\sqrt{W_+W_-}} - 1 + \frac{1}{\sqrt{W_-W_+}} \right) \]

set \( \alpha_t = \beta_t = \log \frac{1}{\sqrt{W_+W_-}} \)

\[ \alpha_t - \beta_t = \log \frac{1}{\sqrt{W_-W_+}} \]

Proof - natural extension of basic proof.
We may want to penalize false positives and false negatives in a different way. For example, detecting faces in images. In a typical image, there will far more non-faces than faces. So we want to be very certain before labelling a window as a face.

Loss function:
\[ L = \sqrt{k_l}, \] if \( w_i = 1 \), and \( H(x_i) = -1 \)
\[ L = \frac{1}{2} \sqrt{k_l}, \] if \( w_i = -1 \), and \( H(x_i) = 1 \).
\[ L = 0, \] otherwise.

Modify the update rule:
\[ D_{t+1}(c) = \frac{1}{Z_t} e^{-\frac{1}{2} \sum \omega_i (d_{t}(x_i) + \beta_t)^2} \times \omega_i \log(\sqrt{k_l}) \]

Verify that the weighted loss is bounded.
\[
\begin{align*}
\left( \frac{1}{N} \sum_{i=1}^{N} \sqrt{k_l} \sum_{w_{i,1}} S_{H(x_i)} - 1 + (1/\sqrt{k_l}) \sum_{\omega_{i,1}} S_{H(x_i)} \right) e^{-\frac{1}{2} \sum \omega_i (d_{t}(x_i) + \beta_t)^2} &\leq \frac{1}{N} \sum_{i=1}^{N} e^{-\frac{1}{2} \sum \omega_i (d_{t}(x_i) + \beta_t)^2} \omega_i \log(\sqrt{k_l})
\end{align*}
\]
(3) This modifies the update rule by
\[ \beta_t \rightarrow \beta_t + (\nu_t) \log \sqrt{\eta_t} \]

Cascade.
Motivation
\[
\text{LogitBoost} \quad (4)
\]

Formulate a log-likelihood:

\[
P(w_i | x_i) = \frac{e^{w_i \lambda \varphi(x_i)}}{\sum_{k=1}^{K} e^{w_k \lambda \varphi(x_i)}}
\]

\[
L(\lambda, \varphi) = \log \prod_{i=1}^{m} P(w_i | x_i)
\]

\[
= - \sum_{i=1}^{m} \log (1 + e^{-2w_i \lambda \varphi(x_i)})
\]

This requires solving the optimization problem:

\[
(\lambda, \varphi) = \arg \max L(\lambda, \varphi)
\]

\[
= \arg \min \sum_{i=1}^{m} \log (1 + e^{-2w_i \lambda \varphi(x_i)})
\]

This differs from AdaBoost (4).

\[
(\lambda, \varphi) = \arg \min \sum_{i=1}^{m} \exp (-w_i \lambda \varphi(x_i))
\]

Claim - these become similar as \( n \to \infty \).

Replace sample mean by expectation:

\[
\text{AdaBoost} \quad (\lambda, \varphi)^{\text{AdaBoost}} = \lambda \mathbb{E} \log (1 + e^{-2w \lambda \varphi(x)})
\]

\[
\text{LogitBoost} \quad (\lambda, \varphi)^{\text{LogitBoost}} = \lambda \mathbb{E} \log (e^{-w \lambda \varphi(x)})
\]
Claim: these problem has the same mean at
\[ F(x) = \mathbb{E}_h f_h(x) = \frac{1}{2} \log \frac{p(x,w=+1)}{p(x,w=-1)} \]

Proof: \[ A(\tau) = \mathbb{E}_{\omega \in \{-1,1\}} \int P(x,y) \log(1 + e^{-2y \tau(x)}) ds \]

\[ SA(\tau) = 0 \]

\[ \Rightarrow p(x,y=+1) - 2e^{-\tau x} + p(x,y=-1) \cdot \frac{2e^{-\tau x}}{1 + e^{-2\tau x}} = 0 \quad \forall x \]

Solving gives:
\[ f^*(x) = \frac{1}{2} \log \frac{p(x,w=+1)}{p(x,w=-1)} \quad \frac{1}{2} \log \frac{p(x,w=+1)}{p(x,w=-1)} \]

Viola & Jones paper.