Ada-Boost

- Error rate: $\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$

-

AdaBoost M1
1. Initialize weights $w_i = 1/N,\quad i=1,2,\cdots,N$
2. for $m = 1:M$
(a) Fit a classifier $G_m(x)$ to the training data
(b) $err_m = \dfrac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$
(c) $\alpha_m = \frac{1}{2}\log\frac{1-err_m}{err_m}$
(d) $w_i \leftarrow w_i \cdot \exp(\alpha_m \cdot I(y_i \neq G_m(x_i)))$
3. $G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$

- Additive Model: $f(x) = \sum_{m=1}^{M} \beta_m \cdot b(x; \gamma_m)$

Forward Stagewise
1. Initialize $f_0(x) = 0$
2. For $m = 1:M$
(a) $(\beta_m, \gamma_m) = \underset{\beta,\gamma}{\text{argmin}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma_m))$
(b) $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

· Exponential Loss : $(\beta_m, G_m) = \underset{\beta, G}{\arg\min} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \beta G(x_i))]$

$$= \underset{\beta, G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta y_i G(x_i))$$

$$G_m = \underset{G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta) \cdot I(y_i = G(x_i))$$

$$+ \sum_{i=1}^{N} w_i^{(m)} \exp(\beta) \cdot I(y_i \neq G(x_i))$$

$$= \underset{G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} (\exp(\beta) - \exp(-\beta)) I(y_i \neq G(x_i))$$

$$+ \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta)$$

$$= \underset{G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))$$

$$\beta_m = \underset{\beta}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} (\exp(\beta) - \exp(-\beta)) I(y_i \neq G(x_i))$$

$$+ \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta)$$

$$= \frac{1}{2} \log \frac{1 - err_m}{err_m}$$

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\beta_m G_m(x_i) \cdot y_i}$$

· Error Bound

$$err = \frac{1}{N}\sum_{i=1}^{N} I(G(x_i)\neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i \sum_{m=1}^{M} \beta_m G_m(x_i))$$

$$= \sum_{i=1}^{N} w_i^{(1)} \prod_{m=1}^{M} \exp(-\beta_m y_m G_m(x_i))$$

$$= Z_1 \sum_{i=1}^{N} w_i^{(2)} \prod_{m=2}^{M} \exp(-\beta_m y_m G_m(x_i))$$

$$= \cdots = \prod_{m=1}^{M} Z_m$$

since $Z_m = \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta_m y_i G_m(x_i))$

$$= \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta_m) I(y = G_m(x_i)) + \sum_{i=1}^{N} w_i^{(m)} e^{\beta_m} I(y_i \neq G_m(x_i))$$

$$= (1-e_m)e^{-\beta_m} + e_m e^{\beta_m}$$

$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$ $\quad = 2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2}$ where $\gamma_m = \frac{1}{2} - e_m$

then $\prod_{m=1}^{M} Z_m = \prod_{m=1}^{M} \sqrt{(1-4\gamma_m^2)} \leq \exp(-2\sum_{m=1}^{M} \gamma_m^2)$

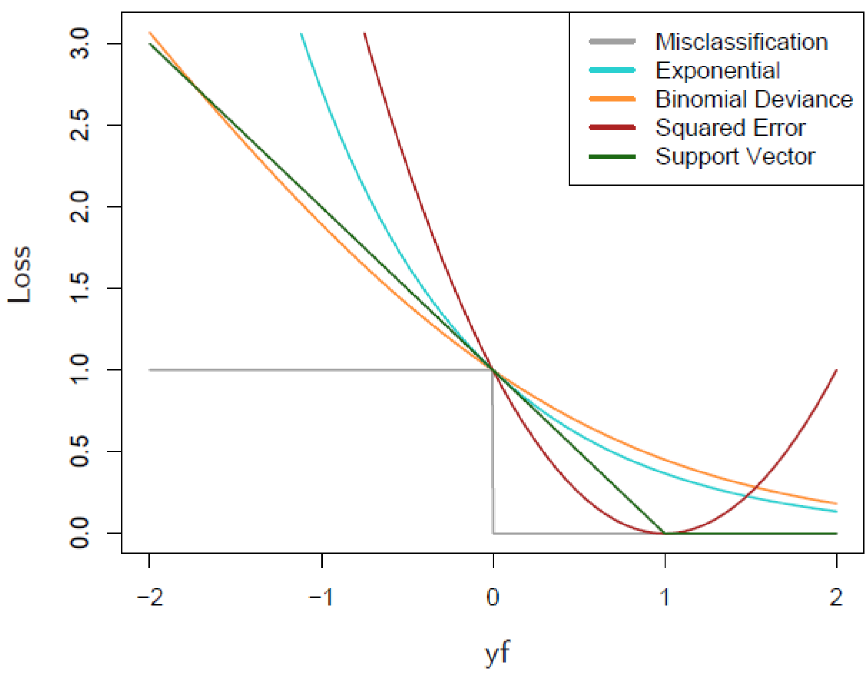

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

Boosting Tree

- **Tree rule:** Partition the space into disjoint regions R_j, $j=1,2,\cdots,J$

  $x \in R_j \Rightarrow f(x) = \gamma_j$, $T(x;\theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$

  where $\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j)$

- **Tree Optimization:**
  - given R_j: $\hat{\gamma}_j = \overline{y_j}$
  - find R_j: $\tilde{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} \tilde{L}(y_i, T(x_i;\Theta))$

- **Boosting Tree:** $f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m)$
  - Stagewise $\tilde{\Theta}_m$: $\tilde{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i;\Theta_m))$

Numerical Boosting

- $\hat{f} = \underset{f}{\operatorname{argmin}} L(f)$, where $f = \{f(x_1), f(x_2), \cdots, f(x_N)\}$
- $f_M = \sum_{m=0}^{M} h_m$, $h_m \in R^N$ is a descent step.

- **Steepest Descent:**

$$g_m = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i) = f_{m-1}(x_i)}$$

$$\rho_m = \underset{\rho}{\operatorname{argmin}} L(f_{m-1} - \rho g_m)$$

then $h_m = -\rho_m g_m$, $f_m = f_{m-1} - \rho_m g_m$

- **Gradient Boosting**
  - $\tilde{\Theta}_m = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} (-g_{im} - T(x_i; \Theta))^2$, a tree closest to gradient

Gradient Tree Boosting

1. Initialized $f_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma)$
2. For $m = 1 : M$
   (a) For $i = 1, 2, \cdots, N$, compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f = f_{m-1}}$$
   (b) Fit a regression tree to fit r_{im} with regions R_{jm}, $j = 1, 2, \cdots, J_m$
   (c) For $j = 1, 2, \cdots, J_m$ compute
   $$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output $\hat{f}(x) = f_M(x)$