CSCE 633: Machine Learning

Lecture 20: More Boosting!

Texas A&M University

10-9-19

Last Time

- Decision Trees
- Random Forest
- Boosting

Goals of this lecture

- Reminder: Exam 1 - Monday, October 14 in class
- CLOSED BOOK, CLOSED NOTES - Starts right at 1:50. DO NOT BE LATE! YOU WILL NOT GET EXTRA TIME!
- Boosting

Boosting:Formulation

Consider $y \in \{-1, +1\}$ instead of $\{0, 1\}$
Can calculate mean error in a classifier f as:

$$\bar{err} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i \neq f(x_i))$$

Some definitions:

- A weak classifier is one that is just slightly better than random guessing.
- we define $m = 1, 2, \cdots, M$ as a sequence of weak classifier models
- You may see formulations of a strong classifier as G f or H - usually capital to denote the strong classifier from weak
- You may see α weights instead of w weights

Boosting:Formulation

$$f(x) = \beta_0 + \sum_{b=1}^{B} \beta_m \phi_m(x)$$

Or - in other notation

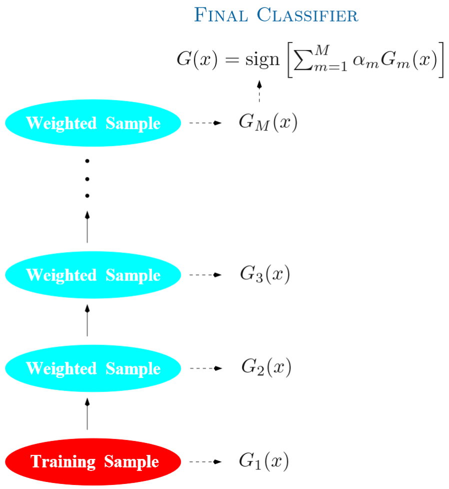
$$f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x)$$

Or - in other notation

$$G(x) = sign(\sum_{m=1}^{M} \alpha_m G_m(x))$$

where α_m are the weights on each weak classifier G_m and the binary classification is just the sum (regression would be mapped to a $[-1, 1]$ interval in the same setting (called Real AdaBoost)

AdaBoost


FINAL CLASSIFIER
$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$
Weighted Sample
$G_M(x)$
Weighted Sample
$G_3(x)$
Weighted Sample
$G_2(x)$
Training Sample
$G_1(x)$

Boosting:Minimizing Loss

Consider $y \in \{-1, +1\}$ instead of $\{0, 1\}$
Can calculate mean error in a classifier f as:

$$e\bar{r}r = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i \neq f(x_i))$$

Boosting aims to minimize loss as:

$$\min_f \sum_{i=1}^{n} L(y_i, f(x_i))$$

Where $L(y_i, \hat(y_i))$ is some loss function. What does this look like visually for binary classification?

Boosting:Minimizing Loss

Consider $y \in \{-1, +1\}$ instead of $\{0, 1\}$
Binary $0 - 1$ loss is not differentiable - so we need convex approximations.

Squared Error Loss

$$f^*(x) = argmin_{f(x)} \mathbb{E}_{y|x}[(Y - f(x))^2] = \mathbb{E}[Y|x]$$

Cannot compute this because it requires $p(y|x)$ to be known. This is commonly known as the population minimizer. Other loss functions with boosting, then, try to approximate this probability.

Boosting: Other Loss Functions

Consider $y \in \{-1, +1\}$ instead of $\{0, 1\}$
Binary $0 - 1$ loss is not differentiable - so we need convex approximations.

Squared Error Loss

$$f^*(x) = argmin_{f(x)} \mathbb{E}_{y|x}[(Y - f(x))^2] = \mathbb{E}[Y|x]$$

Log Loss:

$$f^*(x) = \frac{1}{2} \log \frac{p(\tilde{y} = 1|x)}{p(\tilde{y} = -1|x)}$$

Exponential Loss:

$$L(\tilde{y}, f) = \exp(-\tilde{y} f)$$

Which has the same optimal estimate as log-loss it turns out!

Boosting: Arriving at the optimal

Initially:

$$f_0(x) = argmin_{\gamma)} \sum_{i=1}^{n} L(y_i, f(x_i; \gamma))$$

For squared error, can start with $f_0(x) = \bar{y}$ Then:

$$f_0(x) = \frac{1}{2} \log \frac{\hat{\pi}}{1 - \hat{\pi}}$$

where:

$$\hat{\pi} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i = 1)$$

Boosting: Iterating

Initially:

$$f_0(x) = argmin_{\gamma)} \sum_{i=1}^{n} L(y_i, f(x_i; \gamma))$$

For squared error, can start with $f_0(x) = \bar{y}$ Then:

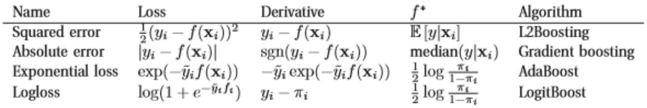
$$f_0(x) = \frac{1}{2} \log \frac{\hat{\pi}}{1 - \hat{\pi}}$$

Iterating:

$$f_m(x) = f_{m-1}(x) + \nu \beta_m \phi(x; \gamma_m)$$

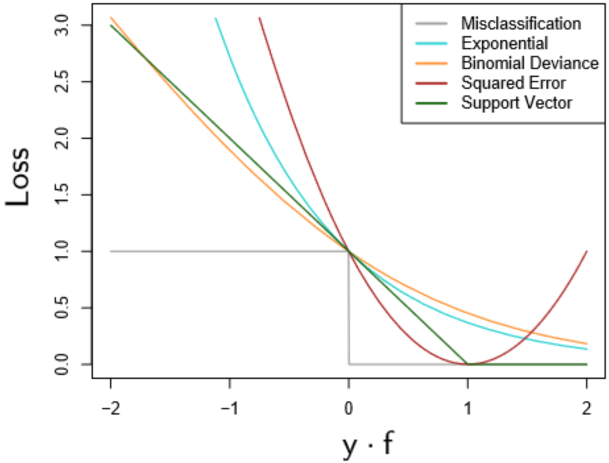
Where $0 < \nu \leq 1$ is a shrinkage parameter to make sure you learn slowly slowly!
Early Stopping: If accuracy (loss) does not improve on a validation set, or AIC, BIC, etc.

Types of Loss

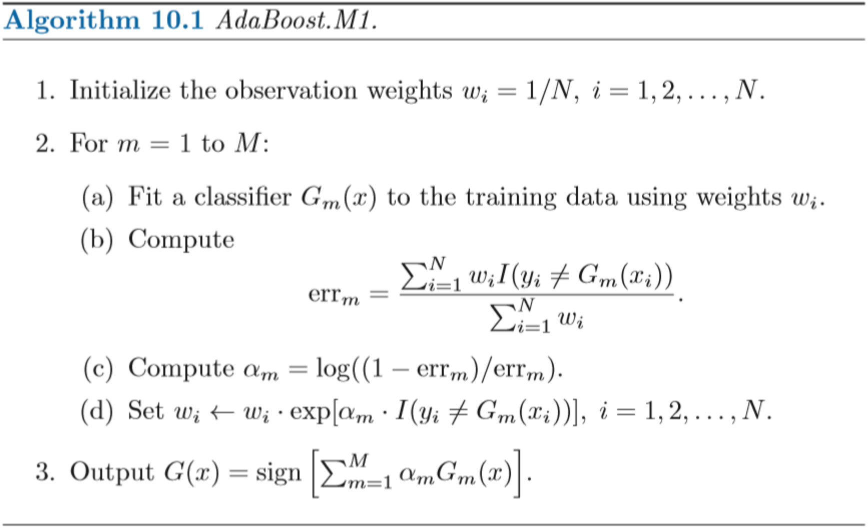


Name	Loss	Derivative	f^*	Algorithm
Squared error	$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y \mid \mathbf{x}_i]$	L2Boosting
Absolute error	$\lvert y_i - f(\mathbf{x}_i) \rvert$	$\operatorname{sgn}(y_i - f(\mathbf{x}_i))$	$\operatorname{median}(y \mid \mathbf{x}_i)$	Gradient boosting
Exponential loss	$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

Types of Loss



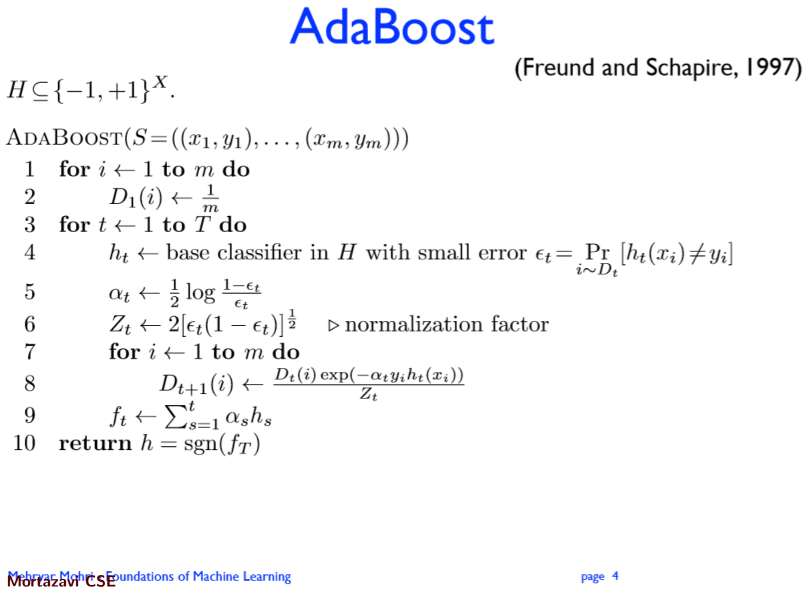
AdaBoost M1



Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

AdaBoost M1



AdaBoost

(Freund and Schapire, 1997)

$H \subseteq \{-1,+1\}^X$.

```
AdaBoost($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      $D_1(i) \leftarrow \frac{1}{m}$
 3  for $t \leftarrow 1$ to $T$ do
 4      $h_t \leftarrow$ base classifier in $H$ with small error $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
 5      $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
 6      $Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$    $\triangleright$ normalization factor
 7      for $i \leftarrow 1$ to $m$ do
 8          $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
 9      $f_t \leftarrow \sum_{s=1}^{t} \alpha_s h_s$
10  return $h = \mathrm{sgn}(f_T)$
```

AdaBoost M1

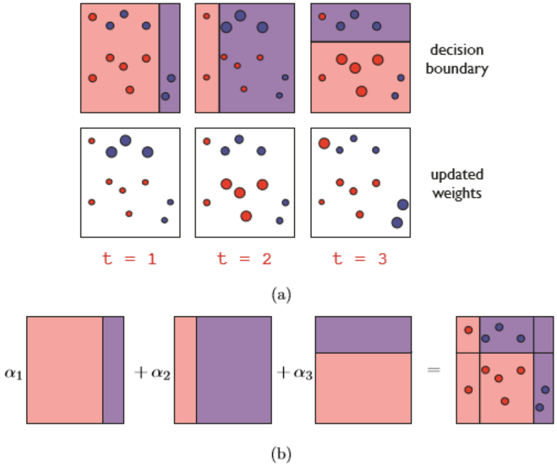


Figure 7.2
Example of AdaBoost with axis-aligned hyperplanes as base classifiers. (a) The top row shows decision boundaries at each boosting round. The bottom row shows how weights are updated at each round, with incorrectly (resp., correctly) points given increased (resp., decreased) weights. (b) Visualization of final classifier, constructed as a non-negative linear combination of base classifiers.

AdaBoost Loss

Consider the exponential loss function:

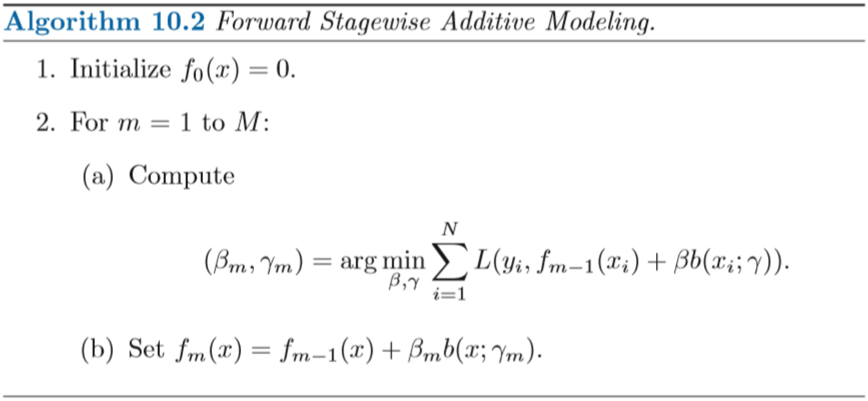
$$L(y, f(x)) = e^{-yf(x)}$$

which means we want to solve:

$$(\beta_m, G_m) = argmin_{\beta,G} \sum_{i=1}^{N} exp[-y_i f_{m-1}(x_i) + \beta G(x_i))]$$

which is a representation of forward stagewise additive modeling

Additive Models



Algorithm 10.2 *Forward Stagewise Additive Modeling.*

```
1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to $M$:
    (a) Compute
        $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$
    (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.
```

AdaBoost Loss

$$(\beta_m, G_m) = argmin_{\beta,G} \sum_{i=1}^{N} exp[-y_i f_{m-1}(x_i) + \beta G(x_i))]$$

Can be re-written as:

$$(\beta_m, G_m) = argmin_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} exp[-\beta y_i G(x_i))]$$

Which we can solve in two steps

AdaBoost Loss

$$(\beta_m, G_m) = argmin_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} exp[-\beta y_i G(x_i))]$$

For $\beta > 0$:

$$G_m = argmin_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))$$

Which minimizes misclassifications

AdaBoost Loss

For $\beta > 0$:

$$G_m = argmin_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))$$

in other words

$$e^{-\beta} \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq G(x_i)} w_i^{(m)}$$

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

AdaBoost Loss

Plugging in:

$$(\beta_m, G_m) = argmin_{\beta,G} \sum_{i=1}^{N} exp[-y_i f_{m-1}(x_i) + \beta G(x_i))]$$

yields:

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

,
where err_m is:

$$err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$$

AdaBoost Loss

and then we update our approximate f as
$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$, which updates weights as
$w_i^{(m+1)} = w_i^{(m)} e^{-\beta_m y_i G_m(x_i)}$
and since

$$-y_i G_m(x_i) = 2 * I(y_i \neq G_m(x_i)) - 1$$

$$w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m I(y_i \neq G_m(x_i))} e_m^{\beta}$$

Why does this work?

$$f^*(x) = argmin_{f(x)} E_{Y|x}(e^{-Yf(x)}) = \frac{1}{2}\log\frac{Pr(Y=1|x)}{Pr(Y=-1|x)}$$

or

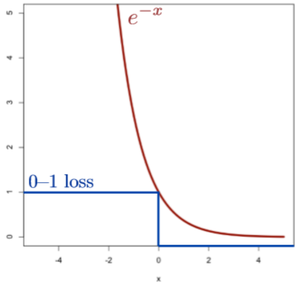
$$Pr(Y=1|x) = \frac{1}{1+e^{-2f^*(x)}}$$

Coordinate Descent

AdaBoost = Coordinate Descent

- **Objective Function**: convex and differentiable.

$$F(\bar{\boldsymbol{\alpha}}) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)}.$$



Coordinate Descent

- Direction: unit vector $\mathbf{e}_k$ with best directional derivative:

$$F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) = \lim_{\eta \to 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta}.$$

- Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) = \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} h_j(x_i) - \eta y_i h_k(x_i)}$,

$$\begin{aligned}
F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) &= -\frac{1}{m} \sum_{i=1}^{m} y_i h_k(x_i) e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} h_j(x_i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} y_i h_k(x_i) \bar{D}_t(i) \bar{Z}_t \\
&= -\left[\sum_{i=1}^{m} \bar{D}_t(i) 1_{y_i h_k(x_i)=+1} - \sum_{i=1}^{m} \bar{D}_t(i) 1_{y_i h_k(x_i)=-1} \right] \frac{\bar{Z}_t}{m} \\
&= -\left[(1 - \bar{\epsilon}_{t,k}) - \bar{\epsilon}_{t,k} \right] \frac{\bar{Z}_t}{m} = \left[\boxed{2\bar{\epsilon}_{t,k} - 1} \right] \frac{\bar{Z}_t}{m}.
\end{aligned}$$

Thus, direction corresponding to base classifier with smallest error.

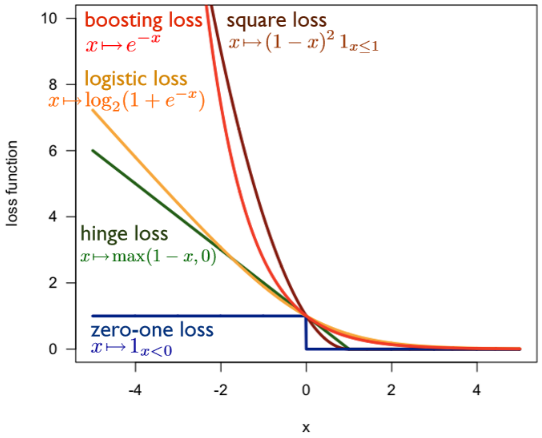
Coordinate Descent

- Step size: η chosen to minimize $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta\, \mathbf{e}_k)$;

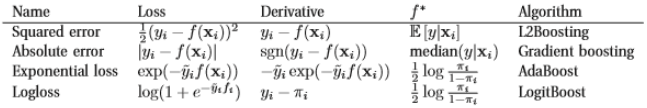
$$\begin{aligned}
\frac{dF(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)}{d\eta} = 0 &\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} h_j(x_i)} e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) \bar{D}_t(i) \bar{Z}_t e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) \bar{D}_t(i) e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\left[(1 - \bar{\epsilon}_{t,k}) e^{-\eta} - \bar{\epsilon}_{t,k} e^{\eta}\right] = 0 \\
&\Leftrightarrow \boxed{\eta = \frac{1}{2} \log \frac{1 - \bar{\epsilon}_{t,k}}{\bar{\epsilon}_{t,k}}.}
\end{aligned}$$

$\searrow$ Thus, step size matches base classifier weight of AdaBoost.

Alternative Loss Functions

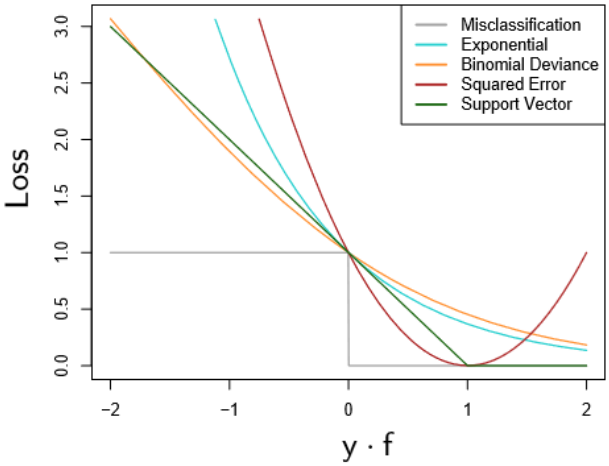


Types of Loss



Name	Loss	Derivative	f^*	Algorithm
Squared error	$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y \vert \mathbf{x}_i]$	L2Boosting
Absolute error	$\vert y_i - f(\mathbf{x}_i)\vert$	$\text{sgn}(y_i - f(\mathbf{x}_i))$	$\text{median}(y \vert \mathbf{x}_i)$	Gradient boosting
Exponential loss	$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

Types of Loss



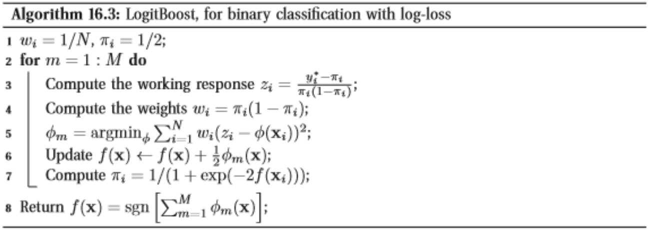
Logit Boost

- Adaboost with exponential loss puts a lot of weight on misclassified examples.
- Additionally - it is hard to interpret probabilities from f(x)
- If we use log-loss instead - mistakes are only punished linearly
- Can also generalize to multiple classes

$$p(y=1|x) = \frac{e^{f(x)}}{e^{-f(x)} + e^{f(x)}} = \frac{1}{1+e^{-2f(x)}}$$

$$L_m(\phi) = \sum_{i=1}^{n} log(1 + \exp(-2\tilde{y}_i(f_{m-1}(x) + \phi(x_i)))$$

Logit Boost: Algorithm



Algorithm 16.3: LogitBoost, for binary classification with log-loss

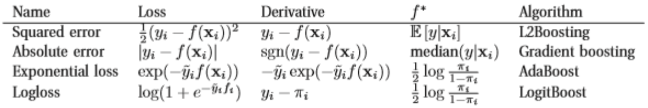
```
1 $w_i = 1/N$, $\pi_i = 1/2$;
2 for $m = 1 : M$ do
3     Compute the working response $z_i = \frac{y_i^* - \pi_i}{\pi_i(1-\pi_i)}$;
4     Compute the weights $w_i = \pi_i(1 - \pi_i)$;
5     $\phi_m = \text{argmin}_\phi \sum_{i=1}^N w_i(z_i - \phi(\mathbf{x}_i))^2$;
6     Update $f(\mathbf{x}) \leftarrow f(\mathbf{x}) + \frac{1}{2}\phi_m(\mathbf{x})$;
7     Compute $\pi_i = 1/(1 + \exp(-2f(\mathbf{x}_i)))$;
8 Return $f(\mathbf{x}) = \text{sgn}\left[\sum_{m=1}^M \phi_m(\mathbf{x})\right]$;
```

Gradient Descent Boosting

- Rather than rebuild the method per loss function, can we generalize?
- Imagine we want to minimize $\hat{f} = argmin_f L(f)$ where the f are the parameters of a model
- Then at step m let g_m be the gradient of $L(f)$ at step $f = f_{m-1}$

$$g_{im} = [\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]$$

Types of Loss



Name	Loss	Derivative	f^*	Algorithm
Squared error	$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y\|\mathbf{x}_i]$	L2Boosting
Absolute error	$\|y_i - f(\mathbf{x}_i)\|$	$\text{sgn}(y_i - f(\mathbf{x}_i))$	$\text{median}(y\|\mathbf{x}_i)$	Gradient boosting
Exponential loss	$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

Functional Gradient Descent

$$g_{im} = [\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]$$

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

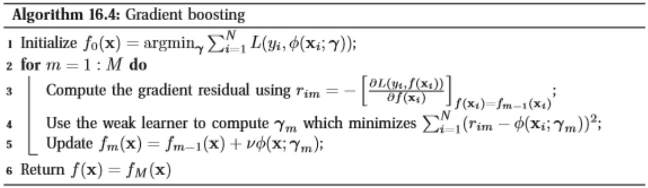
where ρ_m is the step length and

$$\rho = argmin_\rho L(f_{m-1} - \rho g_m)$$

But this does not generalize, only optimizes f for a fixed n. so we have to fit weak learners to approximate the negative gradient signal

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} (-g_{im} - \phi(x_i; \gamma))^2$$

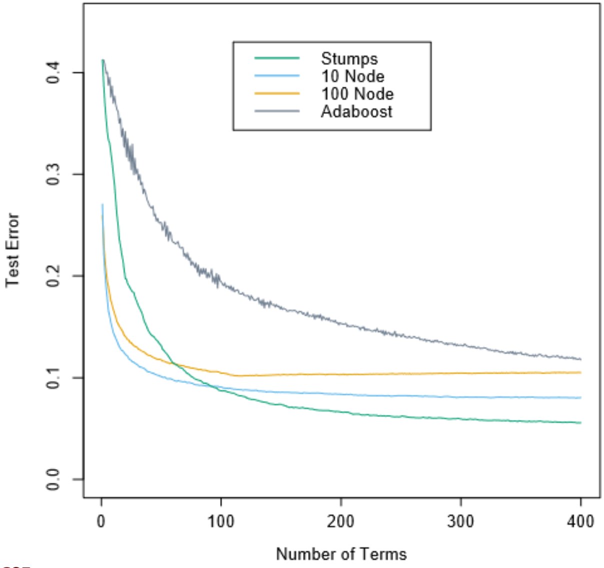
Gradient Descent Boosting: Algorithm



Algorithm 16.4: Gradient boosting

```
1 Initialize $f_0(\mathbf{x}) = \text{argmin}_{\boldsymbol{\gamma}} \sum_{i=1}^{N} L(y_i, \phi(\mathbf{x}_i; \boldsymbol{\gamma}))$;
2 for $m = 1 : M$ do
3     Compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i)}$;
4     Use the weak learner to compute $\boldsymbol{\gamma}_m$ which minimizes $\sum_{i=1}^{N}(r_{im} - \phi(\mathbf{x}_i; \boldsymbol{\gamma}_m))^2$;
5     Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}; \boldsymbol{\gamma}_m)$;
6 Return $f(\mathbf{x}) = f_M(\mathbf{x})$
```

Boosting Comparisons



Takeaways and Next Time

- Boosting
- Next Time: Discussion - Boosting and Exam Review
- Reminder: Exam 1 - Monday, October 14