AdaBoost

Sujay Sanghavi

Motivation

(Let us consider binary classification for now)

Samples are $(x^{(i)}, y^{(i)})$ where each $x^{(i)} \in \mathbb{R}^d$ and each $y^{(i)} \in \{-1, 1\}$

A **classifier** is a function that assigns a label to any given feature vector $f : \mathbb{R}^d \to \{+1, -1\}$

Weak learner: **A classifier which is better than random guessing**, but possibly not much better.

Q: If we have many weak learners, can we make a strong learner by combining them ?

Weak Learners

$$\text{Error of } f \;=\; \frac{1}{n}\sum_i \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

We will also be interested in **weighted error**

$$\sum_i w^{(i)} \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

(for some weights w such that $\sum_i w^{(i)} = 1$ and each $w^{(i)} \geq 0$)

Random guessing has expected error of $1/2$.

We can combine many weak learners by majority $\text{sign}(\sum_t f_t(x))$ or weighted majority $\text{sign}(\sum_t \alpha_t f_t(x))$

Boosting

Task: given **weak learners**, find a sequence of them and combine to make a **strong learner**

- Initialize with all samples having the same "weight"

Boosting

Task: given **weak learners**, find a sequence of them and combine to make a **strong learner**

- Initialize with all samples having the same "weight"
- At each step t,
  - find a "weak learner" that minimizes **weighted error** with weights $w_t^{(i)}$

Boosting

Task: given **weak learners**, find a sequence of them and combine to make a **strong learner**

- Initialize with all samples having the same "weight"
- At each step t,
  - find a "weak learner" that minimizes **weighted error** with weights $w_t^{(i)}$
  - Evaluate the errors made by **this** weak learner on all training samples. The (final) importance of this learner will depend on its error.

Boosting

Task: given **weak learners**, find a sequence of them and combine to make a **strong learner**

- Initialize with all samples having the same "weight"
- At each step t,
  - find a "weak learner" that minimizes **weighted error** with weights $w_t^{(i)}$
  - Evaluate the errors made by **this** weak learner on all training samples. The (final) importance of this learner will depend on its error.
  - Update weights: increase $w_{t+1}^{(i)}$ over $w_t^{(i)}$ for samples where error was made at step t

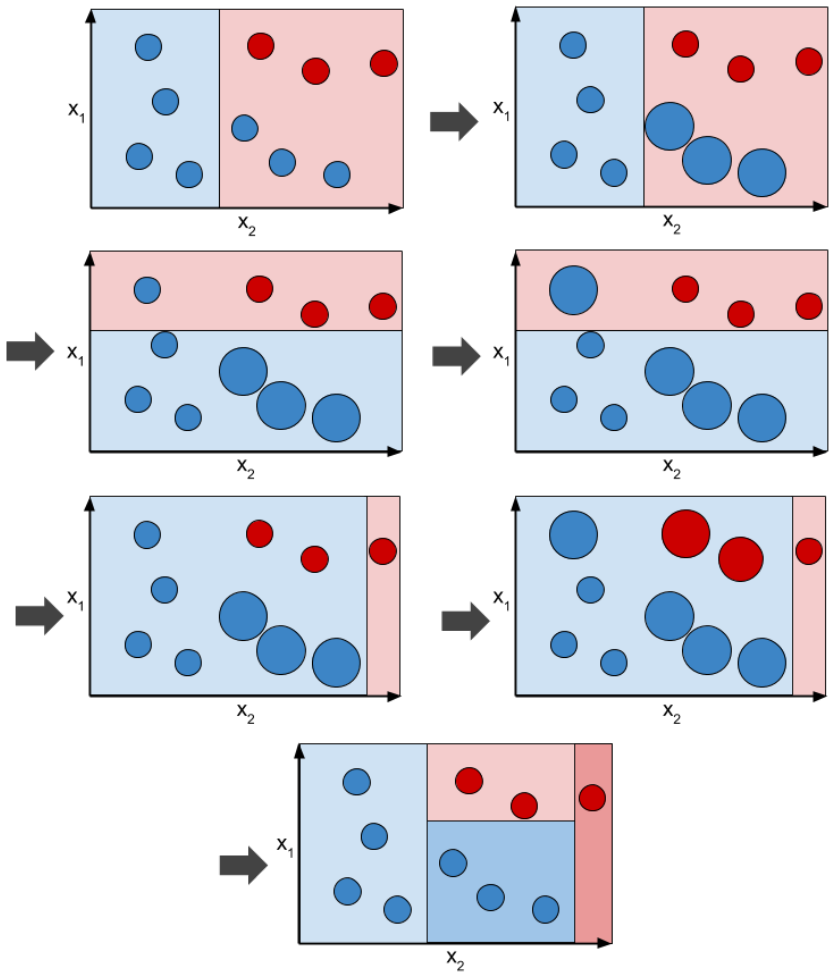
Boosting

Task: given **weak learners**, find a sequence of them and combine to make a **strong learner**

- Initialize with all samples having the same "weight"
- At each step t,
  - find a "weak learner" that minimizes **weighted error** with weights $w_t^{(i)}$
  - Evaluate the errors made by **this** weak learner on all training samples. The (final) importance of this learner will depend on its error.
  - Update weights: increase $w_{t+1}^{(i)}$ over $w_t^{(i)}$ for samples where error was made at step t
- Make final model by combining the individual weak learners

Now: we will see how to do this for classification

Boosting for Classification



Credit: Kevin Ghorbani

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$
- Initial weights $w_1^{(i)} = \frac{1}{n}$

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$
- Initial weights $w_1^{(i)} = \frac{1}{n}$
- At each iteration t
  - Find the weak learner that (approximately) minimizes the weighted error

$$f_t = \widetilde{\arg} \min_{f \in \mathcal{F}} \sum_i w_t^{(i)} \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$
- Initial weights $w_1^{(i)} = \frac{1}{n}$
- At each iteration t
  - Find the weak learner that (approximately) minimizes the weighted error

$$f_t \;=\; \widetilde{\arg}\,\min_{f \in \mathcal{F}} \sum_i w_t^{(i)} \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

  - Evaluate the weighted error and, hence the final importance α, of f_t:

$$\epsilon_t \;=\; \sum_i w_t^{(i)} \mathbf{1}\left\{ f_t(x^{(i)}) \neq y^{(i)} \right\} \qquad \alpha_t \;=\; \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$
- Initial weights $w_1^{(i)} = \frac{1}{n}$
- At each iteration t
  - Find the weak learner that (approximately) minimizes the weighted error

$$f_t \;=\; \widetilde{\arg}\,\min_{f\in\mathcal{F}} \sum_i w_t^{(i)} \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

  - Evaluate the weighted error and, hence the final importance α, of f_t:

$$\epsilon_t \;=\; \sum_i w_t^{(i)} \mathbf{1}\left\{ f_t(x^{(i)}) \neq y^{(i)} \right\} \qquad \alpha_t \;=\; \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

  - update classifier $g = g + \alpha_t f_t$ and set weights $w_{t+1}^{(i)} = e^{-y_i g(x^{(i)})}$
- Output g as final classifier

AdaBoost

Basic idea: minimize the loss $\sum_i e^{-g(x^{(i)})y^{(i)}}$

- Given: Samples $(x^{(i)}, y^{(i)})$, *weak learners:* $f \in \mathcal{F}$, $f(x) \in \{+1, -1\}$
- Initial weights $w_1^{(i)} = \frac{1}{n}$, classifier $g_0(\cdot) = 0$.
- At each iteration t, for $t = 1, \ldots, T$
  - Find the weak learner that (approx) min.s the weighted error

$$f_t \;=\; \widetilde{\arg}\min_{f \in \mathcal{F}} \sum_i w_t^{(i)} \mathbf{1}\left\{ f(x^{(i)}) \neq y^{(i)} \right\}$$

  - Evaluate the weighted error ϵ_t and thus α_t:

$$\epsilon_t \;=\; \sum_i w_t^{(i)} \mathbf{1}\left\{ f_t(x^{(i)}) \neq y^{(i)} \right\} \qquad \alpha_t \;=\; \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

  - update classifier $g_t = g_{t-1} + \alpha_t f_t$ and set weights $w_{t+1}^{(i)} = \frac{1}{Z_t} e^{-y^{(i)} g_t(x^{(i)})}$. Here $Z_t = \sum_i e^{-y^{(i)} g_t(x^{(i)})}$ is the normalization constant.
- Output final classifier $\widehat{y}(x) = \mathit{sign}\{g_T(x)$

Maini Result: Training Error of AdaBoost

Theorem: If, for **any** set of sample weights, we can always find a weak learner with error $\epsilon \leq 1/2 - \gamma$ for some $\gamma > 0$, then the error of the AdaBoost classifier **on the training samples** will go to zero exponentially in the number of steps.

Proof of Training Error Bound

We first look at the weight updates.

$$w_{t+1}^{(i)} = \frac{1}{Z_t} e^{-y^{(i)} g_t(x^{(i)})}$$

which can be rewritten as

$$w_{t+1}^{(i)} = \underbrace{\frac{1}{Z_t}}_{\text{normalization}} \underbrace{w_t^{(i)}}_{\text{old weight}} \underbrace{e^{-y^{(i)} \alpha_t f_t(x^{(i)})}}_{\text{update}}$$

$$w_1^{(i)} = \frac{1}{n} \text{ (initialization)}$$

$$w_2^{(i)} = \underbrace{\frac{1}{Z_1}}_{\text{normalization}} \underbrace{\frac{1}{n}}_{\text{old weight}} \underbrace{e^{-y^{(i)} \alpha_1 f_1(x^{(i)})}}_{\text{update}}$$

$$w_3^{(i)} = \underbrace{\frac{1}{Z_2}}_{\text{normalization}} \underbrace{\left[\frac{1}{Z_1}\frac{1}{n} e^{-y^{(i)} \alpha_1 f_1(x^{(i)})}\right]}_{\text{old weight}} \underbrace{e^{-y^{(i)} \alpha_2 f_2(x^{(i)})}}_{\text{update}}$$

Proof of Training Error Bound

$$\vdots$$

$$w_{T+1}^{(i)} = \frac{\frac{1}{n}\exp\left(-\sum_{t=1}^{T} y^{(i)}\alpha_t f_t(x^{(i)})\right)}{\prod_{t=1}^{T} Z_t}$$

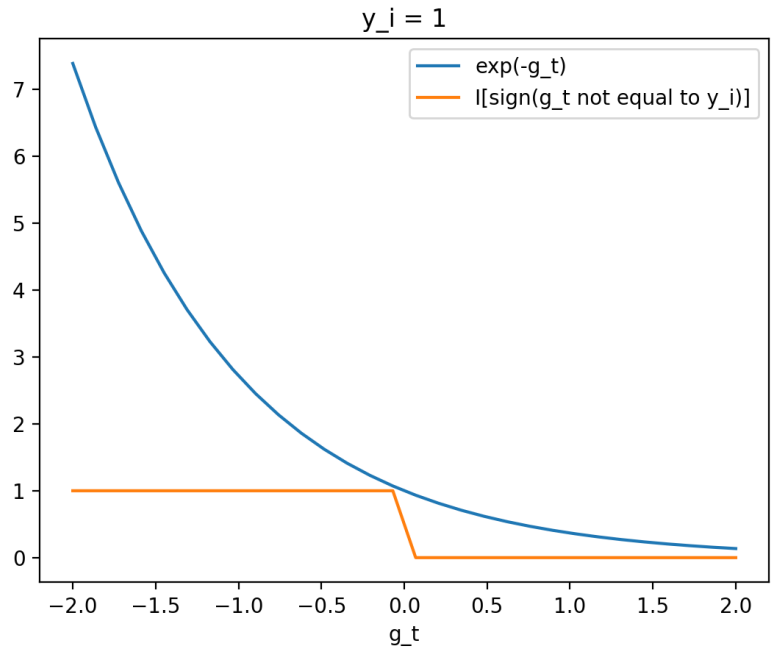
$$= \frac{\frac{1}{n}\exp\left(-y^{(i)}\sum_{t=1}^{T}\alpha_t f_t(x^{(i)})\right)}{\prod_{t=1}^{T} Z_t}$$

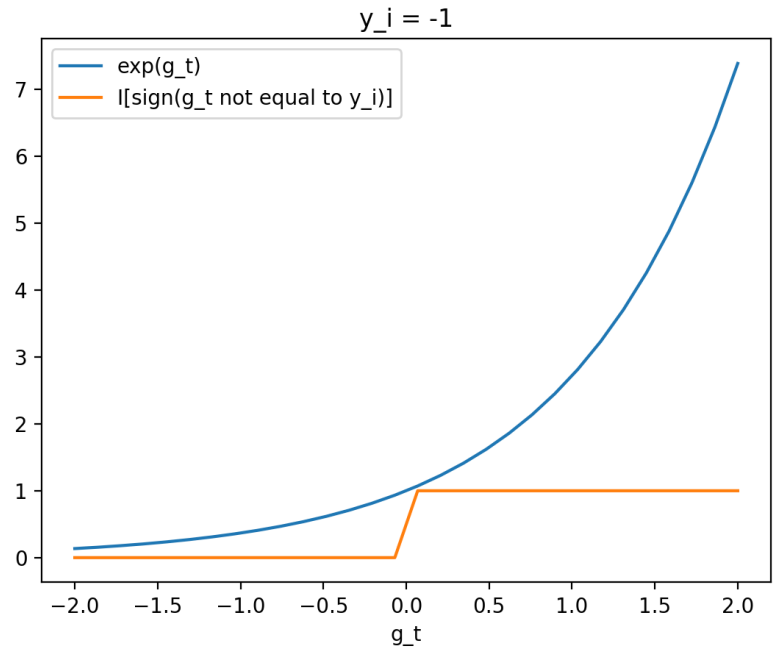
$$= \frac{\frac{1}{n}\exp\left(-y^{(i)} g_t(x^{(i)})\right)}{\prod_{t=1}^{T} Z_t}$$

Proof of Training Error Bound

$$\sum_i w_{T+1}^{(i)} = 1 \text{ by choice of } Z_{T+1}$$

$$\implies \frac{1}{n} \sum_i \exp\left(-y^{(i)} g_t(x^{(i)})\right) = \prod_{t=1}^{T} Z_t \quad \text{(Result 1)}$$





Proof of Training Error Bound

$$\begin{aligned}
\text{Error} &= \frac{1}{n}\sum_i \mathbf{1}\{sign(g_t(x^{(i)})) \neq y^{(i)}\} \\
&\leq \frac{1}{n}\sum_i \exp\left(-y^{(i)} g_t(x^{(i)})\right) \\
\implies \text{Error} &\leq \prod_{t=1}^{T} Z_t \quad \text{(from Result 1)}
\end{aligned}$$

So to minimize error, we want a minimum value of $\prod_{t=1}^{T} Z_t$. We choose α_t such that $\prod_{t=1}^{T} Z_t$ is minimized.

Proof of Training Error Bound - choosing α_t

$$
\begin{aligned}
Z_t &= \sum_i w_t^{(i)} e^{-y^{(i)} \alpha_t f_t(x^{(i)})} \\
&= \sum_{f_t(x^{(i)}) \neq y^{(i)}} w_t^{(i)} e^{\alpha_t} + \sum_{f_t(x^{(i)}) = y^{(i)}} w_t^{(i)} e^{-\alpha_t} \\
&= e^{\alpha_t} \epsilon_t + e^{-\alpha_t}(1 - \epsilon_t)
\end{aligned}
$$

To choose α_t which minimizes this, take derivative and set to 0:

$$
\begin{aligned}
& \frac{d}{d\alpha_t} \left(e^{\alpha_t} \epsilon_t + e^{-\alpha_t}(1 - \epsilon_t) \right) = 0 \\
\implies & e^{\alpha_t} \epsilon_t - e^{-\alpha_t}(1 - \epsilon_t) = 0 \\
\implies & e^{\alpha_t} \epsilon_t = e^{-\alpha_t}(1 - \epsilon_t) \\
\implies & e^{2\alpha_t} = \frac{1 - \epsilon_t}{\epsilon_t} \\
\implies & \alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)
\end{aligned}
$$

Proof of Training Error Bound

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

Plugging α_t in this:

$$\begin{aligned} Z_t &= \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-(1-2\epsilon_t)^2} \\ &\leq \exp(-(1-2\epsilon_t)^2) \end{aligned}$$

Now

$$\begin{aligned} \text{Error} &\leq \prod_{t=1}^{T} Z_t \ \text{(from Result 1)} \\ &\leq \exp\left(-2\sum_{t=1}^{T}\left(\frac{1}{2}-\epsilon_t\right)^2\right) \end{aligned}$$

So the training error decreases exponentially in T.

Bound on Test Error

$$\text{Error}_{\text{test}}(\widehat{y_T}) \leq \text{Error}_{\text{train}}(\widehat{y_T}) + O\left(\sqrt{\frac{Td}{n}}\right)$$

n - Number of train samples - more train samples makes test error close to train error

T - Number of iterations - increase leads to overfitting, and thus higher test error

d - VC dimension - captures how much work is done at each step

Logistic Regression vs AdaBoost

Logistic Regression

- Find classifier f which minimizes Loss $= \sum_i \log(1 + e^{-y^{(i)} f(x^{(i)})})$
- Final classifier $f(x) = sign(\sum_i w_j x_j)$
- w_js are learnt jointly and are the best possible

AdaBoost

- Find classifier f which minimizes Loss $= \sum_i e^{-y^{(i)} f(x^{(i)})})$
- Final classifier $f(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$
- α_ts are learnt sequentially

Gradient Boosting

A generalized framework for AdaBoost, Boosted Regression Trees, and so on.

Samples: $(x^{(i)}, y^{(i)})$
Base (weak) Learners: $\mathcal{F}$ where $f_t \in \mathcal{F}$.
Loss function: $\sum_i L(y^{(i)}, \widehat{y^{(i)}})$

Initialize $g_0(x) = \arg\min_{\gamma} \sum_i L(y^{(i)}, \gamma)$

For $t = 1, \ldots, T$

- Compute pseudo-residual $r_t^{(i)} = -\frac{\partial L}{\partial a}(y^{(i)}, a)\big|_{a=g_{t-1}(x^{(i)})}$
- Fit base learner f_t to weighted samples with weights $r_t^{(i)}$
- $\alpha_t = \arg\min_{\alpha} \underbrace{\sum_i L\left(y^{(i)}, g_{t-1}(x^{(i)}) + \alpha f_t(x^{(i)})\right)}_{\text{new training loss}}$
- Update Classifier $g_t = g_{t-1} + \alpha_t f_t$