Boosting

CSci 5525: Machine Learning

Instructor: Nicholas Johnson

November 12, 2020

Announcements

- Project progress report due in < 1 week (Nov 17)
- Exam 2 coming up (Monday Nov 23, due 48 hours later)
  - Covers lectures 11 (Deep Learning I) - 21 (tentatively PCA)

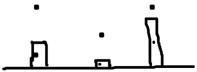
Boosting

- Boosting is a class of algorithms which combine weak learners to create a strong learner
- Rooted in learning theory
- Works very well in practice
- Idea:
  - Apply algorithm to subset of data
  - Obtain weak leaner
  - Apply algorithm to another subset of data
  - Obtain another learner
  - ...
  - Combine learners at the end

Boosting

- How to choose data in each round?
    - Concentrate on "hardest" samples (those misclassified by previous weak learners)
- How to combine learners into single strong learner?
    - Weighted majority vote
- Adaboost is boosting method that implements these ideas

The Boosting Model



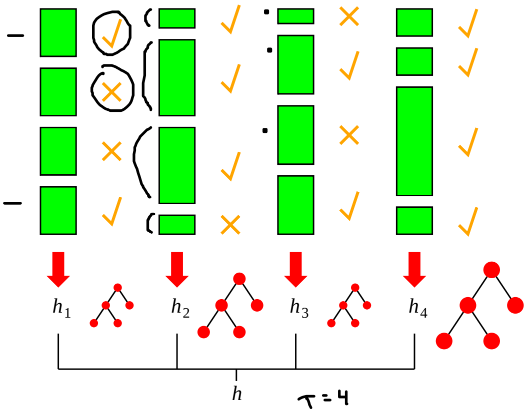
- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds w_t

$c(x)$ = class of x

$\gamma \in (0, \frac{1}{2}]$

  - Booster constructs weight distribution D_t on X (train set)
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x\sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$ G_t
  - After T rounds, the weak hypotheses $h_t, [t]_1^T$ are combined into a final hypothesis h_{final}
- We need procedures w_t
  - → for obtaining D_t at each step
  - → for combining the weak hypotheses

Boosting Algorithms



- Weight decreased on correct samples
- Weight increased on incorrect samples

Adaboost

Final Classifier

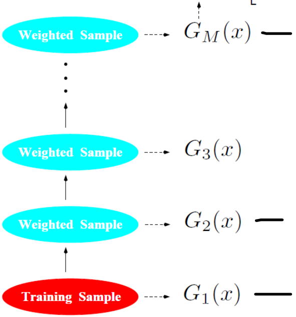
$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

> 0 c_1

< 0 c_2

classifier weight

ensemble



Adaboost Training

$(x_1, y_1), \cdots, (x_n, y_n) \in \mathcal{X} \times \{-1, 1\}$

- Weight on $(\mathbf{x}_i, y_i)$ is $D_t(i) = \underline{w_t(i)}$, learn classifier $G_t(\mathbf{x})$
- The error rate

$$\epsilon_t = \underline{P_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y]} = \sum_{i=1}^{n} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i)) = \sum_{i: y_i \neq G_t(x_i)} w_t(i)$$

- The combined classifier

$$g(\mathbf{x}) = \text{sign}\left[\sum_{t=1}^{T} \alpha_t G_t(\mathbf{x})\right]$$

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \{-1, 1\}$

Algorithm: Initialize $w_1(i) = 1/n$ $\forall i$

For $t = 1, \ldots, T$

- Train a weak learner using distribution w_t
- Get weak hypothesis G_t with error $\epsilon_t = \sum_i w_t(i)\mathbb{1}[G_t(\mathbf{x}_i) \neq y_i]$
- Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ ← classifier weight

  $w_{t+1}(i) = w_t(i)\exp(-\alpha_t)$

- Update

$$w_{t+1}(i) = \frac{w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))}{Z_t}$$

where Z_t is the normalization factor

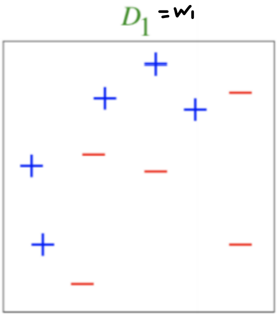
$w_{t+1}(i) = w_t(i)\exp(\alpha_t)$

Output:

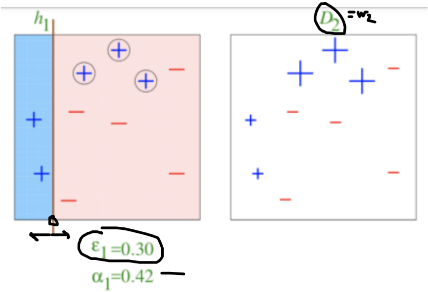
test

$$g(\mathbf{x}) = \text{sign}\left[\sum_{t=1}^{T} \alpha_t G_t(\mathbf{x})\right]$$

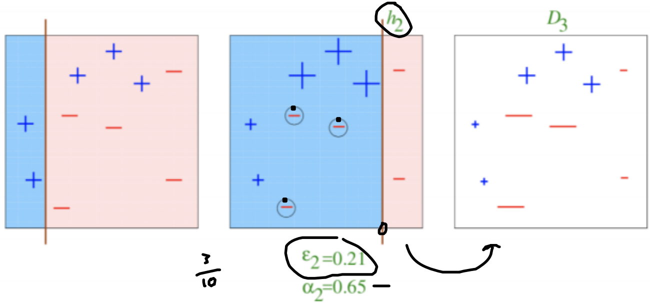
Adaboost Example



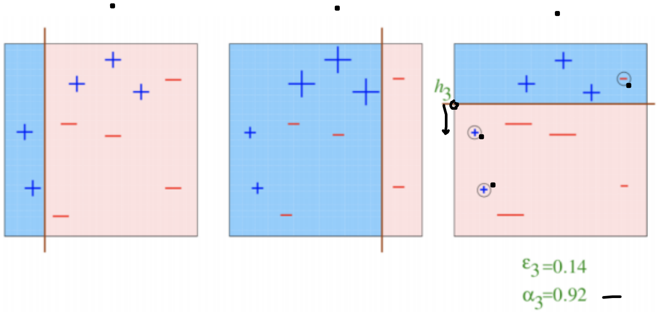
Adaboost Example



Adaboost Example

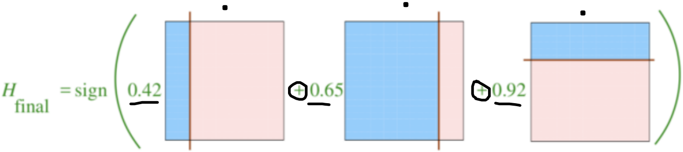


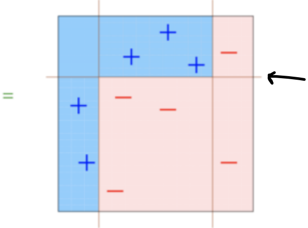
Adaboost Example



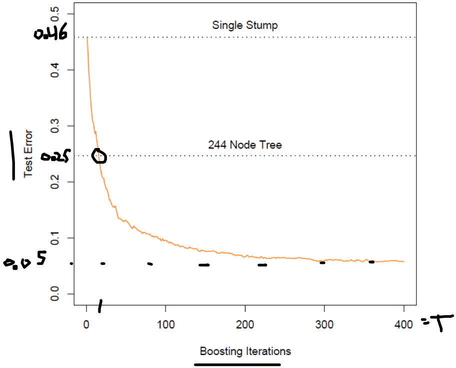
Adaboost Example

$$H_{\text{final}} = \text{sign}\left(0.42 \; \square + 0.65 \; \square + 0.92 \; \square\right)$$





Adaboost Example



- $X_1, \ldots, X_{10}$ are univariate independent Gaussians

$$Y = \begin{cases} 1 & \text{if } \sum_j X_j^2 > \chi^2_{10}(0.5) \\ -1 & \text{otherwise} \end{cases}$$

- $\chi^2_{10}(0.5) = 9.34$, median of chi-squared r.v. with 10 degrees of freedom

Ideas Behind Adaboost

- Hypothesis class Adaboost selects from is

$$\{x \to \text{sign}(f(x)) : f(x) = \sum_{t=1}^{T} \alpha_t G_t(x) \text{ for some}$$
$$\alpha_1, \ldots, \alpha_T \geq 0 \text{ and}$$
$$G_1, \ldots, G_T \in \mathcal{H} \text{ and}$$
$$T \geq 1\}$$

decision stump

- $\mathcal{H}$ is weak learner class (e.g. decision stumps)
- Adaboost greedily minimizes empirical exponential loss $\frac{1}{n} \sum_i \exp(-y_i f(x_i))$

Ideas Behind Adaboost

- Let $f_t = \sum_{\tau=1}^{t} \alpha_\tau G_\tau$
- At round t algorithm greedily improves on f_{t-1} by finding G_t and α_t to minimize

$$\sum_{i=1}^{n} \exp(-y_i f_t(x_i)) = \sum_{i=1}^{n} \exp(-y_i f_{t-1}(x_i)) \exp(-\alpha_t y_i G_t(x_i))$$

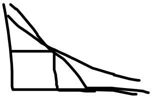
$$\propto \sum_{i=1}^{n} w_t(i) \exp(-\alpha_t y_i G_t(x_i))$$

loss

- Last line follows from definition of w_t:

$$w_t(i) \propto w_{t-1}(i) \exp(-\alpha_{t-1} y_i G_{t-1}(x_i)) \propto \cdots \propto \exp(-y_i f_{t-1}(x_i))$$

The Training Error



- The training error of the final classifier is bounded

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}(G(\mathbf{x}_i)\neq y_i)\leq\frac{1}{n}\sum_{i=1}^{n}\exp(-y_iG(\mathbf{x}_i))=\prod_{t=1}^{T}Z_t$$

- Training error can be reduced most rapidly by choosing α_t that minimizes

$$Z_t=\sum_{i=1}^{n}w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))$$

- Adaboost chooses the optimal α_t
  - Margin is (sort of) maximized
- Other boosting algorithms minimize other upper bounds

Obtaining α_t

- For a given $G_t(\mathbf{x})$, goal is to minimize

samples w/ correct pred

incorrect

$$
\begin{aligned}
f(\alpha) &= \sum_{i=1}^{n} w_t(i) \exp(-\alpha y_i G_t(\mathbf{x}_i)) \\
&= \overbrace{\sum_{i:y_i = G_t(\mathbf{x}_i)} w_t(i) \exp(-\alpha)}^{\text{samples w/ correct pred}} + \overbrace{\sum_{i:y_i \neq G_t(\mathbf{x}_i)} w_t(i) \exp(\alpha)}^{\text{incorrect}} \\
&= (1 - \epsilon_t) \exp(-\alpha) + \epsilon_t \exp(\alpha) \quad (\epsilon_t \text{ is error rate of } G_t) \\
&= \epsilon_t (\exp(\alpha) - \exp(-\alpha)) + \exp(-\alpha)
\end{aligned}
$$

- Recall the error rate at t: $\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i))$
- Minimizing $f(\alpha)$ over α gives:

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

The Training Error (cont.)

- The training error of the final classifier is bounded

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

- For round t, with $\epsilon_t = 1/2 - \gamma_t/2$

$$Z_t = \sqrt{1-\gamma_t^2}$$

- Then the total training error

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \prod_{t=1}^{T}\sqrt{1-\gamma_t^2}$$

Boosting is General Framework

- Boosting is a general framework for constructing ensembles
- Can by viewed as gradient descent in function space with convex cost function
- Many other boosting algorithms: LogitBoost, Gradient Boosted Regression Tree (i.e., gradient boosting), etc.
- Works very well in practice