Boosting
Announcements

Kaggle competition (extra credit) is coming out soon
Recap on Bagging

Construct $\hat{P}$, s.t., $\hat{P}(x_i, y_i) = 1/n, \forall i \in [n]$

$\hat{D}_1 \quad \hat{D}_2 \quad \cdots \quad \hat{D}_k$

$\hat{h}_1 = \text{ID3}(\hat{D}_1) \quad \hat{h}_2 = \text{ID3}(\hat{D}_2) \quad \hat{h}_k = \text{ID3}(\hat{D}_k)$

$\hat{h} = \sum_{i=1}^{k} \frac{\hat{h}_i}{k}$

Does bagging reduce bias?
Today’s Question

Can we combine weak learners into a strong learner?
Outline of Today

1. Gradient Descent without accurate gradient

2. Boosting as Approximate Gradient Descent

3. Example: the AdaBoost Algorithm
Gradient Descent without an accurate gradient

Consider minimizing the following function $L(y), y \in \mathbb{R}^n$

Gradient descent:

$$y_{t+1} = y_t - \eta g_t, \text{ where } g_t = \nabla L(y_t)$$

When $\eta$ is small and $g_t \neq 0$, we know $L(y_{t+1}) < L(y_t)$
Gradient Descent without an accurate gradient

Consider minimizing the following function $L(y), y \in \mathbb{R}^n$

Approximate Gradient descent:

$$y_{t+1} = y_t - \eta \hat{g}_t, \text{ where } \hat{g}_t \neq \nabla L(y_t)$$

Q: Under what condition of $\hat{g}_t$, can we still guarantee $L(y_{t+1}) < L(y_t)$?

A: As long as $\langle \hat{g}_t, \nabla L(y_t) \rangle > 0$
Outline of Today

1. Gradient Descent without accurate gradient

2. Boosting as Approximate Gradient Descent

3. Example: the AdaBoost Algorithm
Key question that Boosting answers:

Combine weak learners together to generate a strong learner with lower bias

(Weak learners: classifiers whose accuracy is slightly above 50%)
Setup

We have a binary classification data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, $(x_i, y_i) \sim P$

Hypothesis class $\mathcal{H}$, hypothesis $h : X \mapsto \{-1, +1\}$

Loss function $\ell(h(x), y)$, e.g., exponential loss $\exp(-yh(x))$

Goal: learn an ensemble $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$, where $h_t \in \mathcal{H}$
The Boosting Algorithm

Initialize $H_1 = h_1 \in \mathcal{H}$

For $t = 1$ …

Find a new classifier $h_{t+1}$, s.t., $H_{t+1} = H_t + \alpha h_{t+1}$ has smaller training error
Training weak learners

Denote $\hat{y} = [H_t(x_1), H_t(x_2), \ldots, H_t(x_n)]^\top \in \mathbb{R}^n$

Define $L(\hat{y}) = \sum_{i=1}^{n} \ell(\hat{y}_i, y_i)$, where $\hat{y}_i = H_t(x_i)$

$L(\hat{y})$: the total training loss of ensemble $H_t$

Q: To minimize $L(\hat{y})$, cannot we just do GD on $\hat{y}$ directly?

A: no, we want to find $\hat{y}$ that minimizes $L$, but it needs to be from some ensemble $H$
Training weak learners

Denote \( \hat{y} = \left[ H_t(x_1), H_t(x_2), \ldots, H_t(x_n) \right]^\top \in \mathbb{R}^n \)

Define \( L(\hat{y}) = \sum_{i=1}^{n} \ell(\hat{y}_i, y_i) \), where \( \hat{y}_i = H_t(x_i) \)

Let us compute \( -\nabla L(\hat{y}) \in \mathbb{R}^n \) — the ideal descent direction

Idea: find a \( h \in \mathcal{H} \), such that \( [h(x_1), \ldots h(x_n)]^\top \) is close to \( -\nabla L(\hat{y}) \)
Training weak learners

\[ -\nabla L(\hat{y}) = \left[ -\frac{\partial \ell(\hat{y}_i, y_i)}{\partial \hat{y}_i}, \ldots, -\frac{\partial \ell(\hat{y}_n, y_n)}{\partial \hat{y}_n} \right]^T \]

\[
\arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} h(x_i) \cdot \frac{\partial \ell(\hat{y}_i, y_i)}{\partial \hat{y}_i} \]

\[ := w_i \]

\[ = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} |w_i| \left( h(x_i) \cdot \text{sign}(w_i) \right) \]

\[ = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} |w_i| \left( 1(h(x_i) = \text{sign}(w_i)) - 1(h(x_i) \neq \text{sign}(w_i)) \right) \]

\[ = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} |w_i| \cdot 1(h(x_i) = \text{sign}(w_i)) = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} |w_i| \cdot 1(h(x_i) \neq -\text{sign}(w_i)) \]

Turned it to a weighted classification problem!
Training weak learners

Finding $[h(x_1), \ldots, h(x_n)]^T$ that is close to $-\nabla L(\hat{y})$ can be done via weighted binary classification:

A new training set:

$$\{p_i, x_i, -\text{sign}(w_i)\}, \text{ where } p_i = \frac{|w_i|}{\sum_{j=1}^n |w_j|}$$

$$h_{t+1} := \arg\min_{h \in \mathcal{H}} \sum_{i=1}^n p_i \cdot \mathbf{1}(h(x_i) \neq -\text{sign}(w_i))$$

$$-\nabla L(\hat{y})$$

$$\hat{y} \rightarrow \hat{y}'$$

$$\hat{y}' = \hat{y} + \alpha [h_{t+1}(x_1), \ldots, h_{t+1}(x_n)]^T$$

$$= [H_t(x_1) + \alpha h_{t+1}(x_1), \ldots, H_t(x_n) + \alpha h_{t+1}(x_n)]^T$$
The Boosting Algorithm Revisit

Initialize $H_1 = h_1 \in \mathcal{H}$

For $t = 1$ ...

Compute $\hat{y}_i = H_t(x_i), \forall i \in [n]$

Compute $w_i := \partial \ell(\hat{y}_i, y_i)/\partial \hat{y}_i$, and normalize $p_i = \left| \frac{w_i}{\sum_j w_j} \right|, \forall i$

Run classification: $h_{t+1} = \arg \min \sum_{i=1}^{n} p_i \cdot 1(h(x_i) \neq -\text{sign}(w_i))$

Add $h_{t+1}$: $H_{t+1} = H_t + \alpha h_{t+1}$
Outline of Today

1. Gradient Descent without accurate gradient

2. Boosting as Approximate Gradient Descent

3. Example: the AdaBoost Algorithm
Train Weak learner

We will choose the exponential loss, i.e., $\ell(\hat{y}, y) = \exp(-y \cdot \hat{y})$

$$w_i = \frac{\partial \ell(\hat{y}_i, y_i)}{\partial \hat{y}_i} = - \exp(-\hat{y}_i y_i) y_i$$

$$|w_i| = \exp(-\hat{y}_i y_i)$$

$$p_i = \frac{|w_i|}{\sum_j |w_j|}$$

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^n p_i \mathbf{1}(h(x_i) \neq -\text{sign}(w_i))$$

$$= \arg\min_{h \in \mathcal{H}} \sum_{i=1}^n p_i \cdot \mathbf{1}(h(x_i) \neq y_i)$$

Binary classification on weighted data

$\tilde{\mathcal{D}} = \{p_i, x_i, y_i\}$, where $\sum_i p_i = 1$, $p_i \geq 0, \forall i$

Q: what does it mean if $p_i$ is large?
Compute learning rate

Select the best learning rate $\alpha$

$$h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} p_i \cdot 1(h(x_i) \neq y_i) \quad H_{t+1} = H_t + \alpha h_{t+1}$$

Find the best learning rate via optimization:

$$\arg\min_{\alpha > 0} \sum_{i=1}^{n} \ell(H_t(x_i) + \alpha h_{t+1}(x_i), y_i)$$

Compute the derivative wrt $\alpha$, set it to zero, and solve for $\alpha$
Put everything together: AdaBoost

Initialize $H_1 = h_1 \in \mathcal{H}$

For $t = 1 \ldots$

Compute $w_i = -y_i \exp(-H_t(x_i)y_i)$, and normalize $p_i = \frac{|w_i|}{\sum_j |w_j|}, \forall i$

Run classification: $h_{t+1} = \arg \min_{h \in \mathcal{H}} \sum_{i=1}^{n} p_i \cdot 1(h(x_i) \neq y_i)$

Weak learner’s loss $\epsilon = \sum_{i:y_i \neq h_{t+1}(x_i)} p_i$ // total weight of examples where $h_{t+1}$ made mistakes

$H_{t+1} = H_t + \frac{1}{2} \ln \frac{1 - \epsilon}{\epsilon} \cdot h_{t+1}$ // the best $\alpha = 0.5 \ln((1 - \epsilon)/\epsilon)$

Weights can be computed incrementally (see note)
Weaker learner: axis-aligned linear decision boundary

\( h_1 \) weights

Final learner

\( h_2 \)

\( h_3 \)
Take home message

Boosting combines weak learners into a stronger learner; it can reduce bias (e.g., it combines linear decision boundaries into a non-linear decision boundary)