Boosting

Announcements

Kaggle competition (extra credit) is coming out soon

Recap on Bagging

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

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

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

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

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 η 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 $\left\langle \hat{g}_t, \nabla L(y_t) \right\rangle > 0$

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 $H_1 = h_1 \in \mathscr{H}$

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

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

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

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

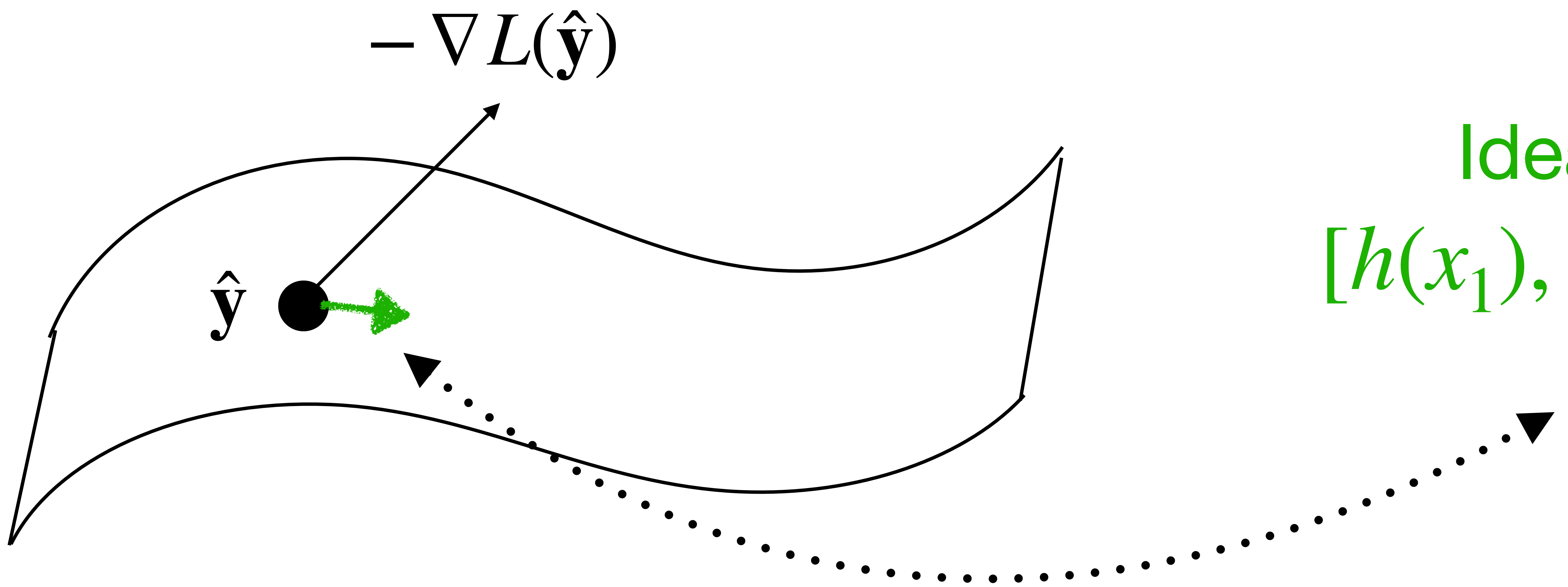
A: no, we want find $\hat{\mathbf{y}}$ that minimizes L, but it needs to be from some ensemble H

Training weak learners

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

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

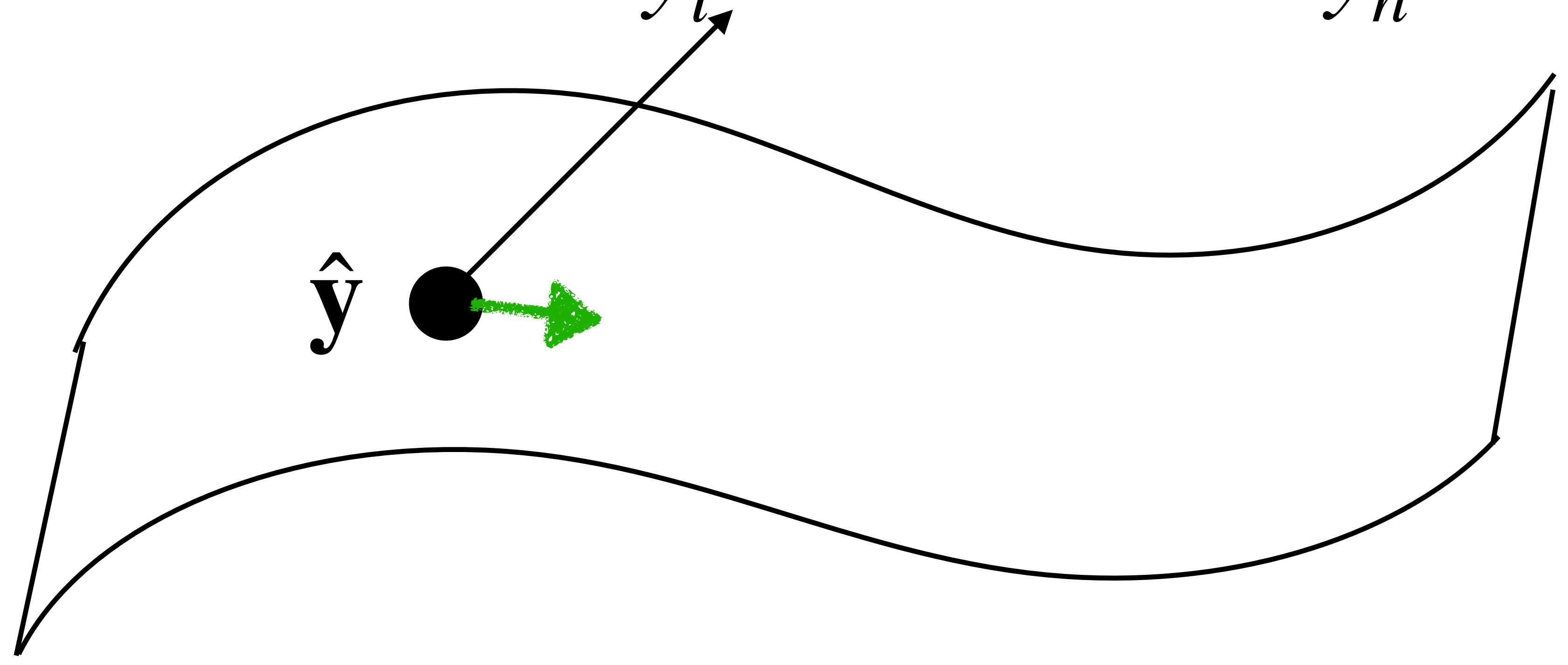
Let us compute $\nabla L(\hat{\mathbf{y}}) \in \mathbb{R}^n$ — the ideal descent direction



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

Training weak learners

$$-\nabla L(\hat{\mathbf{y}}) = [-\frac{\partial \ell(\hat{y}_i, y_i)}{\partial \hat{y}_i}, \ldots, -\frac{\partial \ell(\hat{y}_n, y_n)}{\partial \hat{y}_n}]^\top$$



$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} h(x_i) \cdot \underbrace{\frac{\partial \ell(\hat{y}_i, y_i)}{\partial \hat{y}_i}}_{:=w_i}$$

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

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

Turned it to a weighted classification problem!

$$= \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} \left|w_i\right| \cdot \mathbf{1}(h(x_i) = \mathrm{sign}(w_i)) \quad = \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} \left|w_i\right| \cdot \mathbf{1}(h(x_i) \neq -\mathrm{sign}(w_i))$$

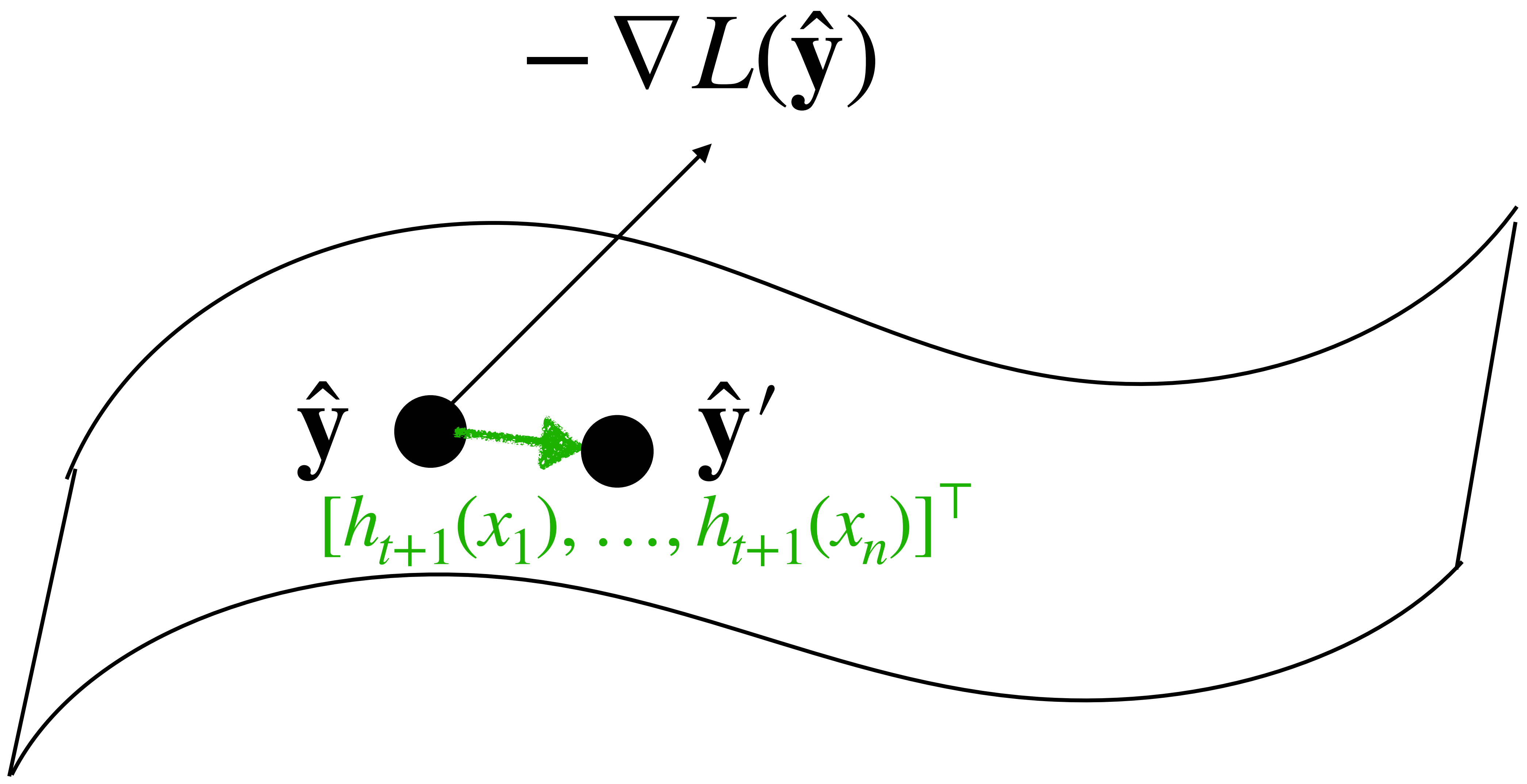
Training weak learners

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

A new training set:

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

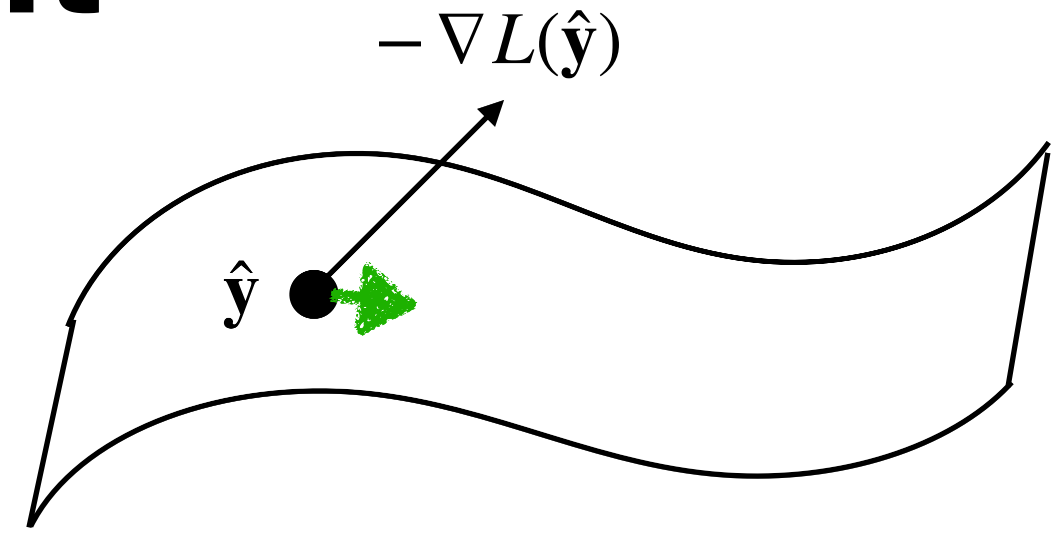
$$h_{t+1} := \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} p_i \cdot \mathbf{1}(h(x_i) \neq -\text{sign}(w_i))$$



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

$$= \left[H_t(x_1) + \alpha h_{t+1}(x_1), \ldots, H_t(x_n) + \alpha h_{t+1}(x_n)\right]^\top$$

The Boosting Algorithm Revisit



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

For t = 1 …

$$\arg\max_{h \in \mathscr{H}} (-\nabla L(\hat{\mathbf{y}}))^\top \begin{bmatrix} h(x_1) \\ h(x_2) \\ \cdots \\ h(x_n) \end{bmatrix}$$

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 $p_i = \left| w_i \right| / \sum_j \left| w_j \right|, \forall i$

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

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., $\ell(\hat{y}, y) = \exp(-y \cdot \hat{y})$

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

$$|w_i| = \exp(-\hat{y}_i y_i) \quad p_i = |w_i| / \sum_j |w_j|$$

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

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

Binary classification on weighted data

$\widetilde{\mathcal{D}} = \{p_i, x_i, y_i\}$, 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 α

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

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 α, set it to zero, and solve for α

Put everything together: AdaBoost

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

For t = 1 …

Weights can be computed incrementally (see note)

Compute $w_i = -y_i \exp(-H_t(x_i)y_i)$, and normalize $p_i = \left| w_i \right| / \sum_j \left| w_j \right|, \forall i$

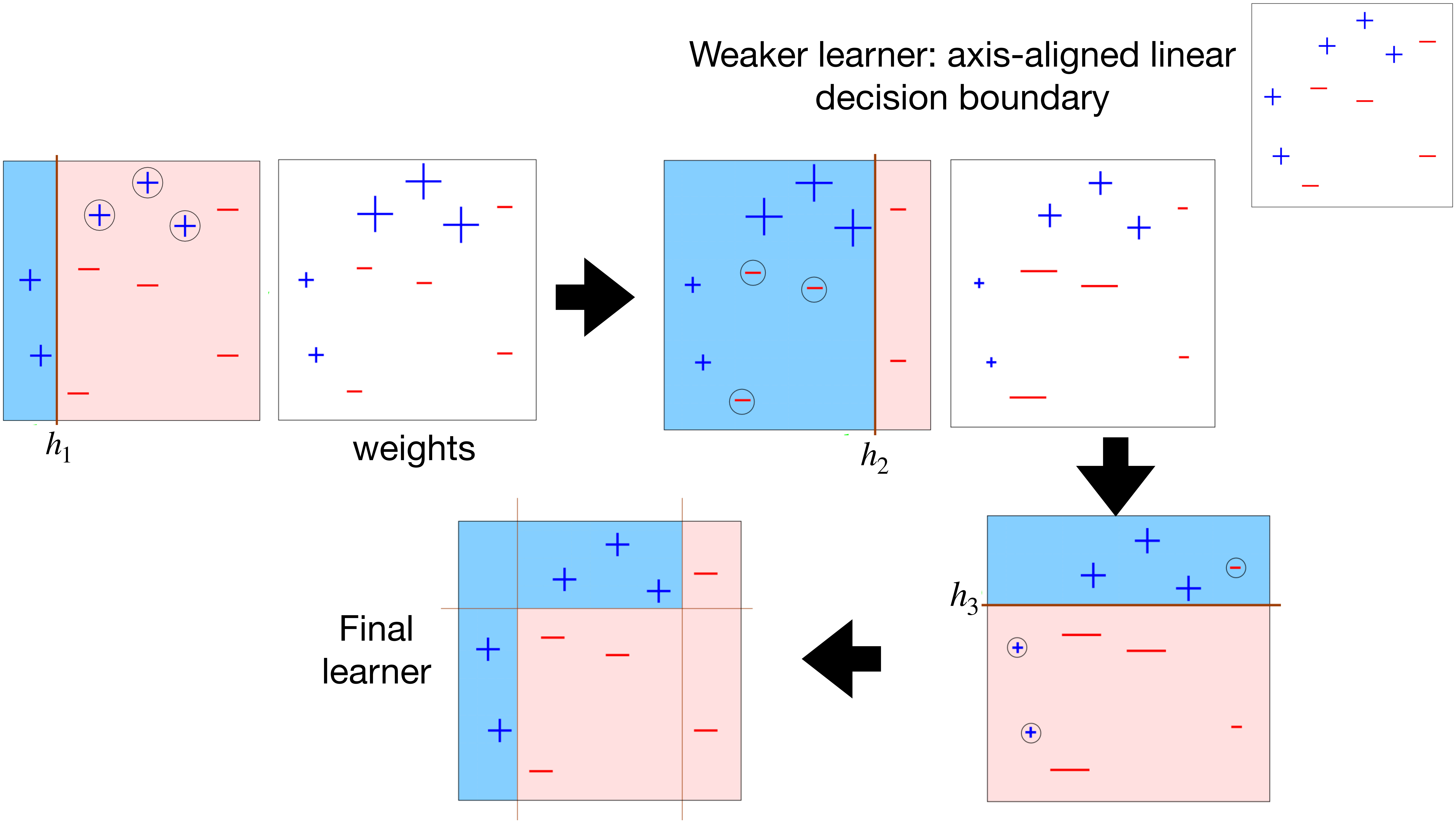
Run classification: $h_{t+1} = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} p_i \cdot \mathbf{1}(h(x_i) \neq y_i)$

Weak learner's loss $\epsilon = \sum_{i: y_i \neq h_{h+1}(x_i)}^{n} p_i$

// 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 $\alpha = 0.5 \ln((1-\epsilon)/\epsilon)$



Weaker learner: axis-aligned linear decision boundary
h_1
weights
h_2
h_3
Final learner

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)