Introduction to Machine Learning
TTIC 31020

Prof. Nati Srebro

Lecture 12:

Boosting

Feature Selection

"Weak" vs "Strong" Learning

Rule $A(\cdot)$ is a **strong learner** for $\mathcal{H}$ (in the realizable setting) if:

For any $\mathcal{D}$ s.t. $\exists_{h^* \in \mathcal{H}} L_{\mathcal{D}}(h^*) = 0$, and **any $\boldsymbol{\epsilon} > \mathbf{0}$**, using $m(\epsilon)$ sample, $\mathbb{E}_{S \sim \mathcal{D}^m}\left[L_{\mathcal{D}}\big(A(S)\big)\right] < \epsilon$

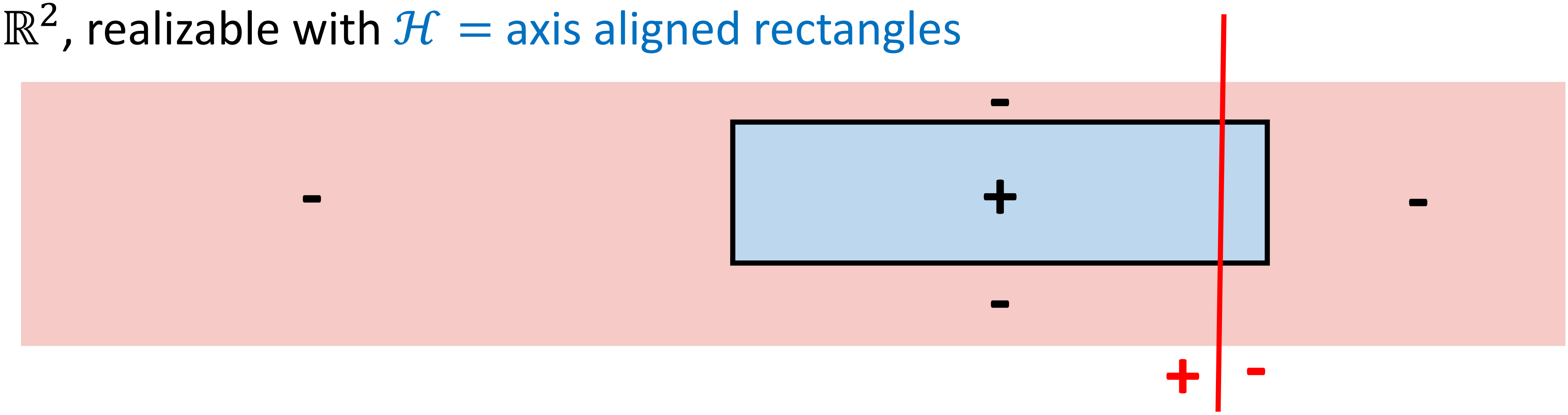
Rule $A(\cdot)$ is a **weak learner** for $\mathcal{H}$ if:

There **exists $\boldsymbol{\epsilon}$** $= {}^1/_2 - \gamma < {}^1/_2$, m, s.t. for any $\mathcal{D}$ with $\exists_{h^* \in \mathcal{H}} L_{\mathcal{D}}(h^*) = 0$, $\mathbb{E}_{S \sim \mathcal{D}^m}\left[L_{\mathcal{D}}\big(A(S)\big)\right] < \epsilon$
(e.g. $\epsilon = 0.49, \gamma = 0.01$)

- If $\mathcal{H}$ is weakly learnable, is it also strongly learnable?
    - Yes: $\mathcal{H}$ is weakly learnable ➔ VCdim($\mathcal{H}$)<∞ ➔ $\mathcal{H}$ is (strongly) learnable
- If we have access to a (**tractable**) weak learner $A(\cdot)$,
  can we use it to build a (**tractable**) strong learner?

Example: Weak Learning with a Weak Class

- $\mathcal{X} = \mathbb{R}^2$, realizable with $\mathcal{H}$ = axis aligned rectangles



- Decision stumps: $\mathcal{B} = \{ [[\, s \cdot x[i] < \theta]] \mid i = 1,2,\ s = \pm 1,\ \theta \in \mathbb{R} \}$
- Claim: For any $\mathcal{D}$, if $\exists_{h_{\blacksquare} \in \mathcal{H}} L_{\mathcal{D}}(h_{\blacksquare}) = 0$ ➔ $\exists_{h \in \mathcal{B}} L_{\mathcal{D}}(h) \leq \frac{3}{7} < 0.429$
- Since VCdim($\mathcal{B}$)=3, with $m = m_{VC}(D = 3, \epsilon = 0.001)$: $\mathbb{E}_{S \sim \mathcal{D}^m}\left[L_{\mathcal{D}}\left(ERM_{\mathcal{B}}(S) \right) \right] < 0.43$
- Conclusion:
  $ERM_{\mathcal{B}}(\cdot)$ is a **weak learner** with $\epsilon = 0.43 < 0.5$

Boosting the Confidence

If the learning algorithm works only with some very small fixed probability $1-\delta_0$ (e.g. $1-\delta_0 = 0.01$), can we construct a new algorithm that works with arbitrarily high probability $1-\delta$ (for any $\delta > 0$) ?

- For any δ:

1. For i=1..k: $\left(k = \frac{\log\ 2/\delta}{\log 1/\delta_0}\right)$
   Collect m_0 independent samples S_i
   $h_i = A(S_i)$
2. Collect $m_{\text{val}} = \frac{4\log\left({}^{4k}/_{\delta}\right)}{\epsilon^2}$ additional independent samples S_{val}
3. Return $\hat{h} = \arg\min_{h_1,\ldots,h_k} L_{S_{val}}(h_i)$

w.p. $\geq 1-\delta$,
$\inf_i L(h_i) \leq \epsilon_0$

ERM from
class of size k

- Claim: w.p. $\geq 1-\delta$, $L(\hat{h}) \leq \epsilon_0 + \epsilon$
- Total samples used: $O\left(m_0(\epsilon_0)\cdot\log\frac{1}{\delta} + \frac{\log\frac{1}{\delta}}{\epsilon^2}\right)$

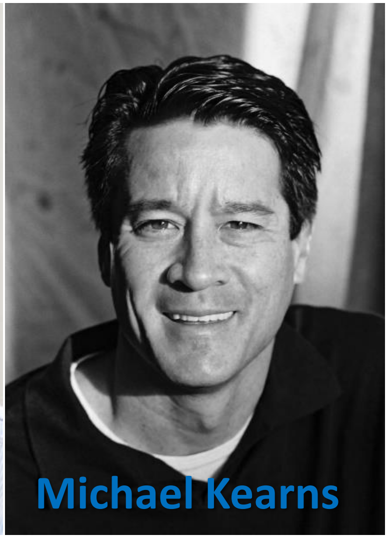
E.g. if $\epsilon_0 = \frac{1}{2} - \gamma < 1/2$, take $\epsilon = \frac{\gamma}{2}$ so that $\epsilon_0 + \epsilon = \frac{1}{2} - \frac{\gamma}{2} < \frac{1}{2}$

Optional material.
Not covered in class.
Not required.

Boosting the Error

If a learning algorithm always returns a predictor that is guaranteed to be slightly better then chance, i.e. has error $\epsilon_0 = \frac{1}{2} - \gamma < \frac{1}{2}$ (for some $\gamma > 0$), can we construct a new algorithm that achieves arbitrarily low error ϵ?

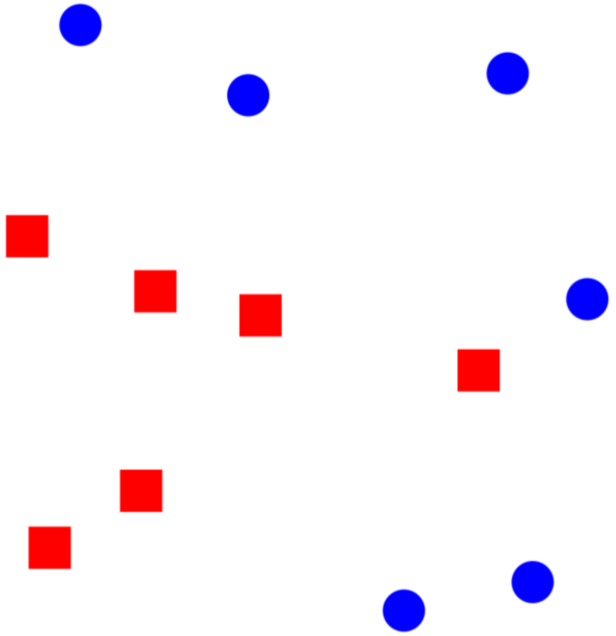
- Posed (as a theoretical question) by Valiant and Kearns, Harvard 1988
- Solved by MIT student Robert Schapire, 1990
- AdaBoost Algorithm by Schapire and Yoav Fruend, AT&T 1995



AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \leq \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \neq y_i]] \leq \frac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

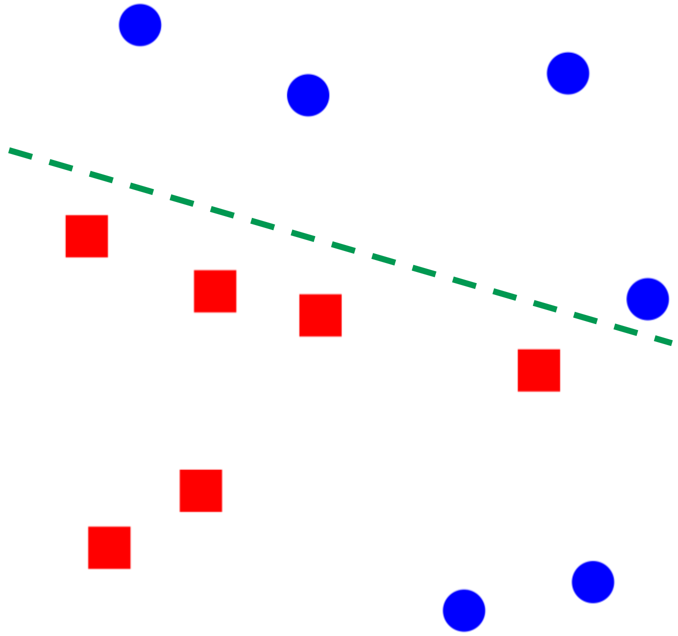
$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \leq \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \neq y_i]] \leq \frac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp(-\alpha_t y_j h_t(x_j))}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

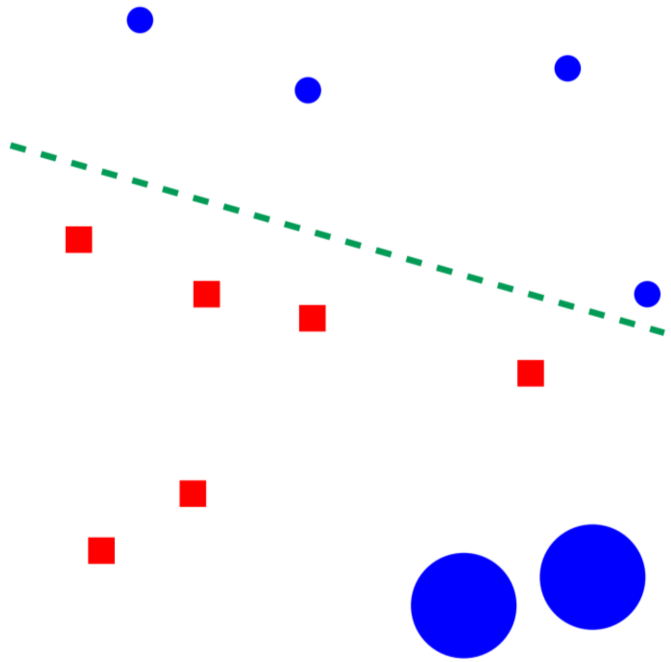
$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
    - If thinking of $A(S')$ as accepting a sample S':

      S' subsamples (with replacement) iid from dist D over S
      i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
    - Usually easier to think of A as operating on a weighted sample, with weights D_i
    - Returns predictors with $L_D(h) \le \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \neq y_i]] \le \tfrac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

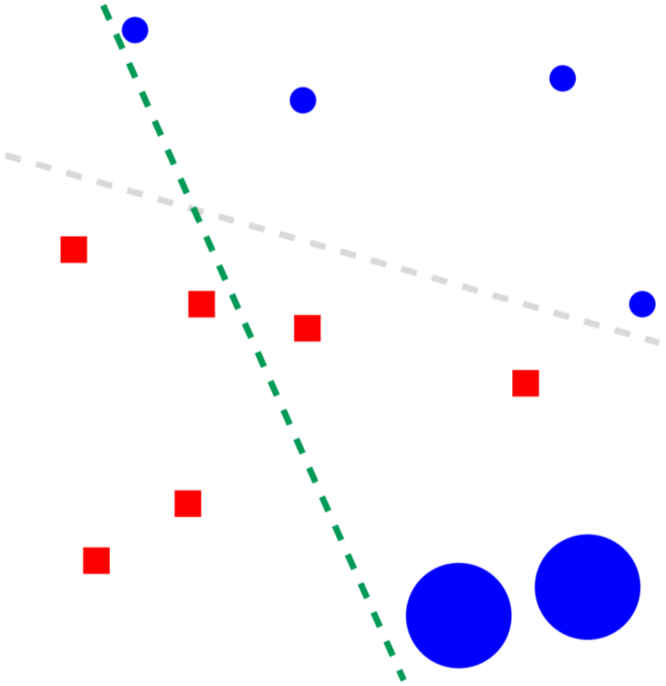
$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions* D *over* S
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \le \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \ne y_i]] \le \tfrac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

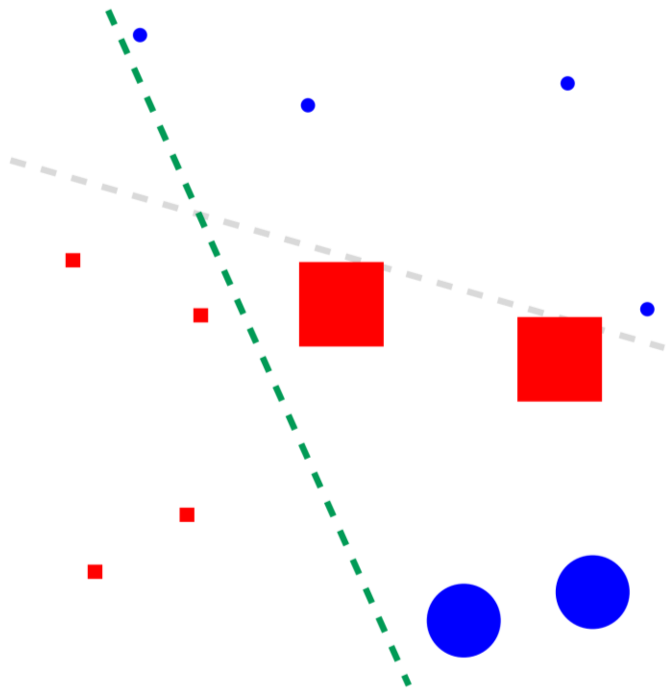
$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \ne y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \leq \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \neq y_i]] \leq \frac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

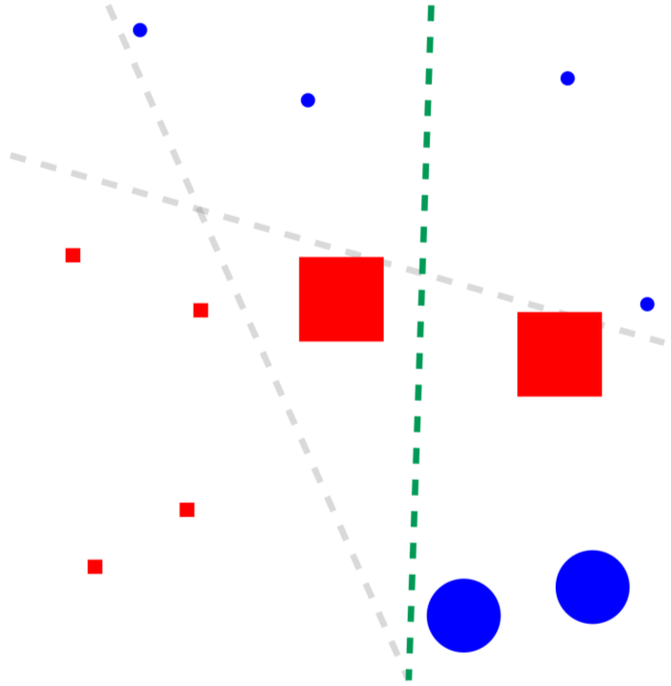
$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost

- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \le \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)



Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \ne y_i]] \le \frac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

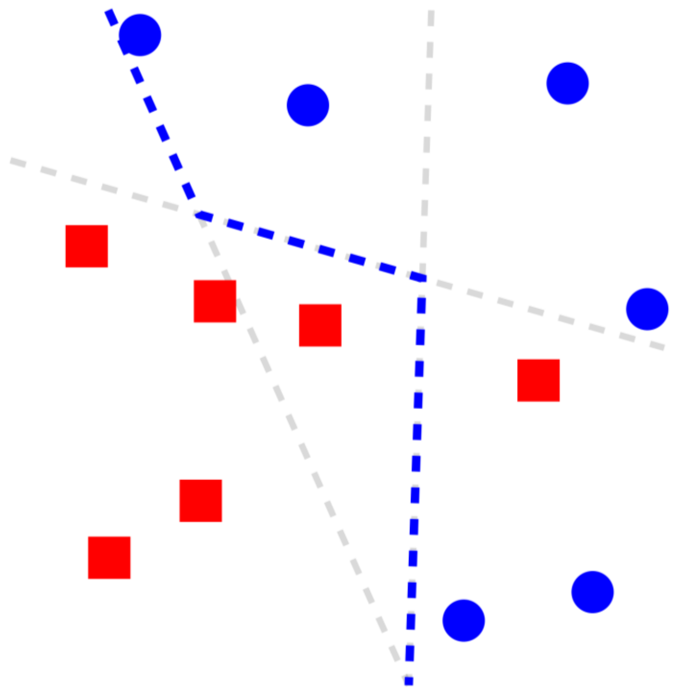
Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \ne y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost



- Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
- Weak Learner A, which will be applied to *distributions D over S*
  - If thinking of $A(S')$ as accepting a sample S':

    S' subsamples (with replacement) iid from dist D over S
    i.e. each $(x, y) \in S'$ is set to (x_i, y_i) with prob D_i
  - Usually easier to think of A as operating on a weighted sample, with weights D_i
  - Returns predictors with $L_D(h) \leq \frac{1}{2} - \gamma$
- Output: hypothesis h (which we want to have small $L_{\mathcal{D}}(h)$)

Initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right)$

For t=1, …, T:

$$h_t = A(D^{(t)})$$

$$\epsilon_t = L_{D^{(t)}}(h_t) = \sum_i D_i^{(t)} \cdot [[h_t(x_i) \neq y_i]] \leq \frac{1}{2} - \gamma$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{\sum_j D_j^{(t)} \exp\left(-\alpha_t y_j h_t(x_j)\right)}$$

Output: $\bar{h}_T(x) = sign\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

- Increase weight on errors, decrease on correct points:

$$D_i^{(t+1)} \propto \begin{cases} D_i^{(t)} \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if } h_t(x_i) \neq y_i \\ D_i^{(t)} \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if } h_t(x_i) = y_i \end{cases}$$

- Claim: $L_{D^{(t+1)}}(h_t) = 0.5$

AdaBoost as Learning a Linear Classifier

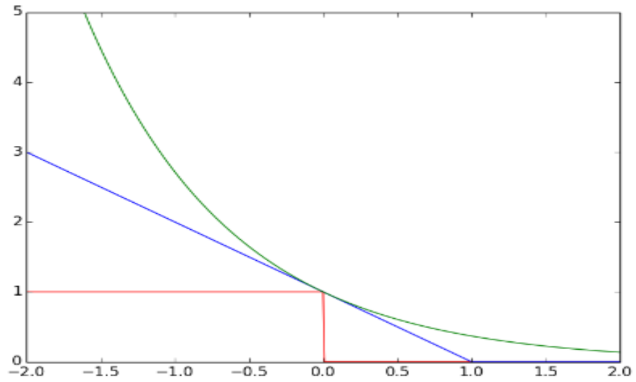
- Recall: $\bar{h}_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
- Let $\mathcal{B} = \{\, all\ hypothesis\ outputed\ by\ A \,\}$
  - "Base Class", e.g. decision stumps

$\phi(x)[h] = h(x)$

$$\underbrace{\bar{h}_T \in \{\, h_w(x) = sign(\langle w, \phi(x)\rangle) \mid w \in \mathbb{R}^{\mathcal{B}} \,\}}_{\text{Class of halfspaces } \mathcal{L}(\mathcal{B})}$$

$w[h] = \sum_{h_t = h} \alpha_t$

$$L_S^{\exp}(w) = \frac{1}{m} \sum \ell^{\exp}(h_w(x_i); y_i) \qquad \ell^{\exp}(z, y) = e^{-yz}$$



- Each step of AdaBoost: Coordinate descent on $L_S^{\exp}(w)$
  - Choose coordinate h of $\phi(x)$ s.t. $\frac{\partial}{\partial w[h]} L_S^{\exp}(\mathrm{w})$ is high
  - Update $w[h] = \arg\min L_S^{\exp}(w)$ s.t. $\forall_{h' \neq h} w[h']$ is unchanged

Coordinate Descent on $L_S^{\exp}(w)$

- $\frac{\partial}{\partial w[h]} L_S^{exp}(w) = \frac{\partial}{\partial w[h]} \frac{1}{m} \sum e^{-y_i h_w(x_i)}$

$$= \frac{1}{m} \sum e^{-y_i h_w(x_i)} \left(-y_i \frac{\partial h_w(x_i)}{\partial w[h]}\right) = \frac{1}{m} \sum e^{-y_i h_w(x_i)} (-y_i h(x_i))$$

$$= \frac{1}{m} \sum \underbrace{e^{-y_i \sum_{t=1}^{T-1} \alpha_t h_t(x_i)}}_{\prod_{t=1}^{T-1} e^{-y_i \alpha_t h_t(x_i)} \propto D_i^{(T)}} (-y_i h(x_i)) \propto \mathbf{1 - 2} L_{D^{(T)}}(h)$$

- Minimize $L_{D^{(T)}}(h)$ ➔ Maximize $\frac{\partial}{\partial w[h]} L_S^{\exp}(w)$

- Updating $w[h]$: set $w^{(t)}[h_t] = w^{(t-1)}[h_t] + \alpha$

$$\alpha = \arg\min L_S^{\exp}(w^{(t)})$$

➔ $0 = \frac{\partial}{\partial \alpha} L_S^{\exp}(w^{(t)}) = \frac{\partial}{\partial w[h_t]} L_S^{\exp}(w^{(t)}) \propto 1 - 2L_{D^{(t+1)}}(h_t)$

➔ choose α s.t. $L_{D^{(t+1)}}(h_t) = \frac{1}{2}$

AdaBoost as Optimization: Minimizing $L_S(h)$

- **Theorem**: If $\forall_t\ \epsilon_t \le \frac{1}{2} - \gamma$, then $L_S^{01}\left(\bar{h}_T\right) \le L_S^{\exp}\left(\bar{h}_T\right) \le e^{-2\gamma^2 T}$

$D_i^{T+1} = \frac{1}{m}\prod_{t=1}^{T}\frac{e^{-y_i\alpha_t h_t(x_i)}}{z_t}$

$\sum_i D_i^{T+1} = 1$

Proof: $L_S^{\exp}\left(\bar{h}_T\right) = \frac{1}{m}\sum_i e^{-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i)} = \frac{1}{m}\sum_i\left(D_i^{(T+1)} m \prod_{t=1}^{T} z_t\right) = \prod_{t=1}^{T} z_t$

$= \prod_{t-1}^{T}\left(2\sqrt{\epsilon_t(1-\epsilon_t)}\right) \le \left((1-2\gamma)(1+2\gamma)\right)^{T/2} = (1-4\gamma^2)^{T/2} \le e^{-2\gamma^2 T}$

If $\mathbb{E}\left[L_{\mathcal{D}}\left(A(S)\right)\right] \le \frac{1}{2} - \gamma$, can construct $\tilde{A}$ that ensures $L_{\mathcal{D}}\left(\tilde{A}(\tilde{S})\right) \le \frac{1}{2} - \frac{\gamma}{2}$ with prob$\ge 1-\delta_0$ using $|\tilde{S}| = O(|S|\log 1/\delta_0)$

- If $A(\cdot)$ is a **weak learner** returning $L_{\mathcal{D}}\left(A(S)\right) \le \epsilon_0 = \frac{1}{2} - \gamma$ with probability $\ge 1 - \delta_0 = 1 - \frac{\delta}{T}$

$\exists_{h^*} L_{\mathcal{D}}(h^*) = 0$ ➔ $L_S(h^*) = 0$ ➔ $L_{D^{(t)}}(h^*) = 0$ ➔ w.p. $1-\delta$, $L_{D^{(t)}}(h_t) \le \frac{1}{2} - \gamma$

➔ w.p. $1-\delta$, $L_S\left(\bar{h}_T\right) \le e^{-2\gamma^2 T}$

- To get $L_S\left(\bar{h}_T\right) \le \epsilon$ for any $\epsilon > 0$, run AdaBoost for $T = \frac{\log\left(\frac{1}{\epsilon}\right)}{2\gamma^2}$ rounds
- Setting $\epsilon = \frac{1}{2m}$, after $T = \frac{\log(2m)}{2\gamma^2}$ rounds: $L_S\left(\bar{h}_T\right) = 0$!
- What about $L_{\mathcal{D}}\left(\bar{h}_T\right)$?

AdaBoost Learns a Linear Predictor

"Base Class" $\mathcal{B}$={predictors outputted by weak learner}, e.g. decision stumps

- After T iterations of AdaBoost:

$$\bar{h}_T = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \in \underbrace{\{ h_w(x) = sign(\langle w, \phi(x) \rangle) \mid w \in \mathbb{R}^{\mathcal{B}} \}}_{\text{Class of halfspaces } \mathcal{L}(\mathcal{B})}$$

- VCdim($\mathcal{L}(\mathcal{B})$) $= |\mathcal{B}| = \infty$
- Even with a relatively "weak" base class, can get very complex $\bar{h}_T$.
  - E.g., with decision stumps over $\mathbb{R}$, can get any piecewise constant function; approximate any function arbitrarily well

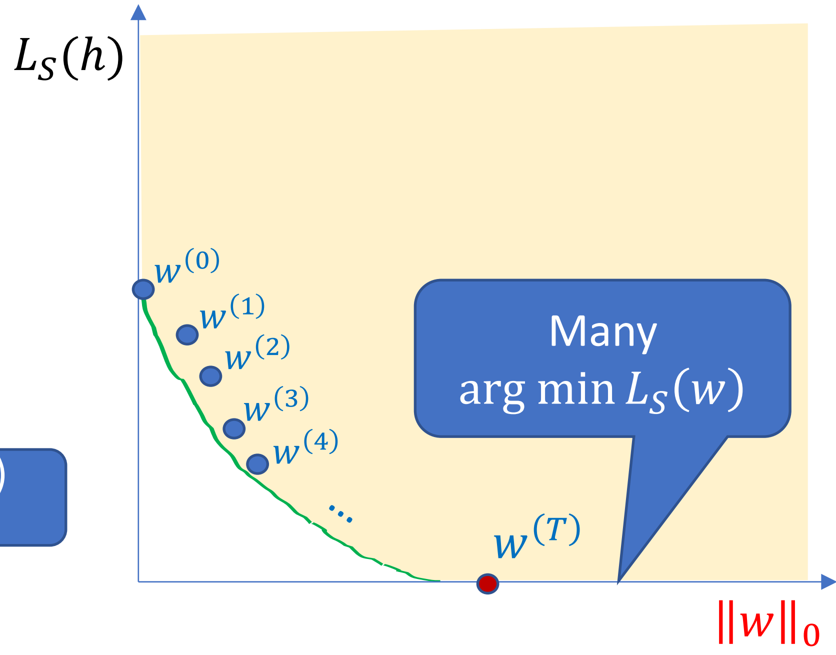
AdaBoost Learns a Sparse Linear Predictor

"Base Class" $\mathcal{B}$={predictors outputted by weak learner}, e.g. decision stumps

- After T iterations of AdaBoost:

$$\bar{h}_T = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \in \underbrace{\left\{ h_w(x) = sign(\langle w, \phi(x)\rangle) \mid w \in \mathbb{R}^{\mathcal{B}}, \|w\|_0 \le T \right\}}_{\text{Class of } \textbf{sparse} \text{ halfspaces } \mathcal{L}(\mathcal{B},T)}$$

- VCdim($\mathcal{L}(\mathcal{B})$) $= |\mathcal{B}| = \infty$
- Even with a relatively "weak" base class, can get very complex $\bar{h}_T$.
  - E.g., with decision stumps over $\mathbb{R}$, can get any piecewise constant function; approximate any function arbitrarily well



- For finite $\mathcal{B}$: VCdim($\mathcal{L}(\mathcal{B},T)$) $\le O(T \log|\mathcal{B}|)$
  - "Parameter counting" intuition: $\binom{|\mathcal{B}|}{T}$ choices for support, requires $\log \binom{|\mathcal{B}|}{T} = O(T \log|\mathcal{B}|)$ bits, plus T params for weights.
- Even if $\mathcal{B}$ is infinite (e.g. in the case of decision stumps):

$$\mathrm{VCdim}\big(\mathcal{L}(\mathcal{B},T)\big) \le \tilde{O}\big(T \cdot \mathrm{VCdim}(\mathcal{B})\big)$$

VC(decision stumps) $=O(\log d)$

- **Number of rounds T (= sparseness) is complexity control**

Complexity Control: Sparsity(=#iterations) T

- After T iterations of AdaBoost:

$$\bar{h}_T = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \in \underbrace{\left\{ h_w(x) = sign(\langle w, \phi(x) \rangle) \mid w \in \mathbb{R}^{\mathcal{B}}, \|w\|_0 \leq T \right\}}_{\text{Class of } \textbf{sparse} \text{ halfspaces } \mathcal{L}(\mathcal{B},T)}$$

- Want low T so that we can generalize

- Realizable case (MDL): use first T s.t. $L_S\left(\bar{h}_T\right) = 0$

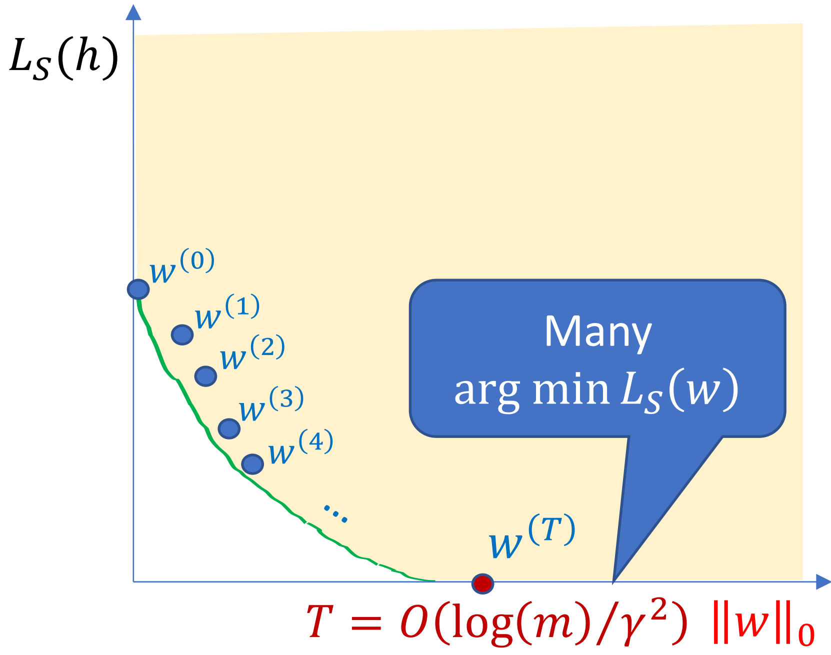
➔ We know this will happen with $T = O\left(\frac{\log(m)}{\gamma^2}\right)$

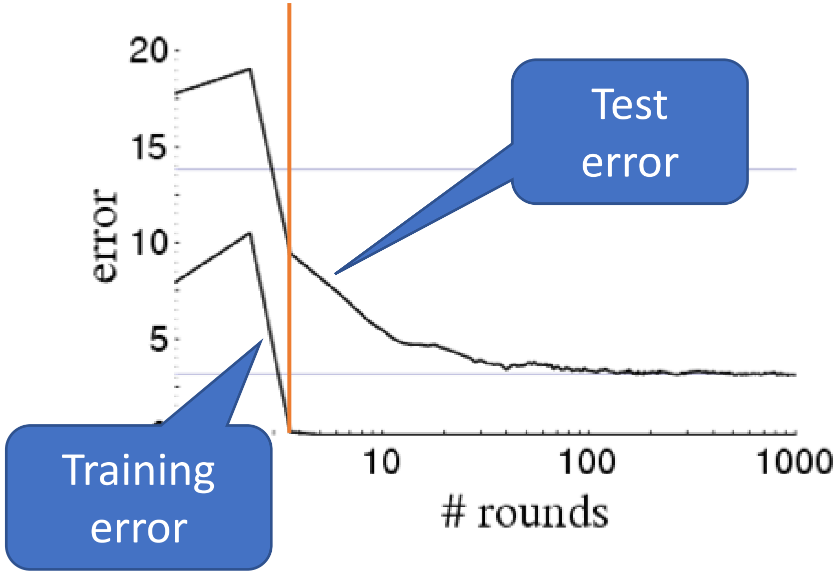
VC(decision stumps) $=O(\log d)$

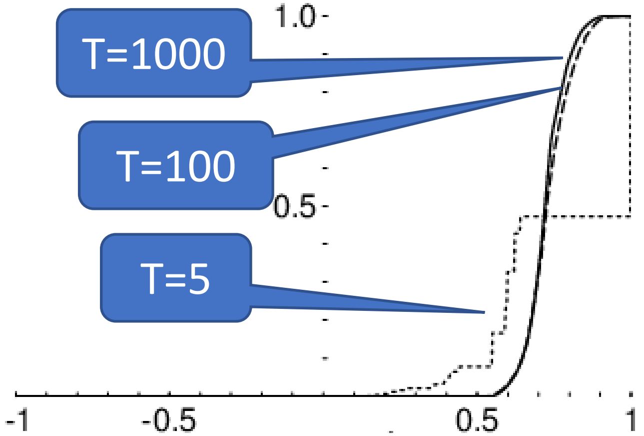
➔ $L\left(\bar{h}_T\right) \leq \tilde{O}\left(\frac{VC(\mathcal{L}(\mathcal{B},T))}{m}\right) = \tilde{O}\left(\frac{T \cdot VC(\mathcal{B})}{m}\right) = \tilde{O}\left(\frac{VC(\mathcal{B}) \log(m)}{\gamma^2 \, m}\right)$

➔ sample complexity: $m = \tilde{O}\left({VC(\mathcal{B})}/{\gamma^2}\right)$

- More generally: Balance fit $L_S\left(\bar{h}_T\right)$ and complexity T (SRM)
  - "Early Stopping" as regularization
  - Use validation (or cross-validation) to select T







$$\frac{y_i\langle w,\phi(x_i)\rangle}{\|w\|_1}=\frac{\sum\alpha_t h_t(x)}{\sum\alpha_t}$$

$$\overline{h}_T(x)=\sum_{t=1}^{T}\alpha_t h_t(x)\ =\langle w,\phi(x)\rangle$$

$$\|w\|_1=\sum_i|w[i]|=\sum_t\alpha_t$$

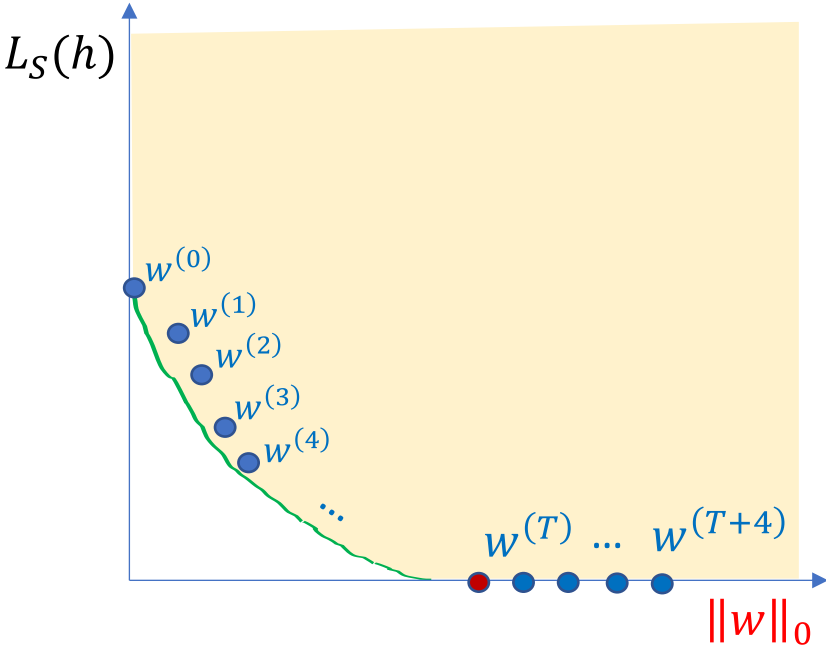
- Even after $L_S^{01}\left(\overline{h}_T(x)\right)=0$, AdaBoost keeps improving the *margin*, and hence generalization
- But it's a ℓ_1 margin and not ℓ_2 margin as in SVM…

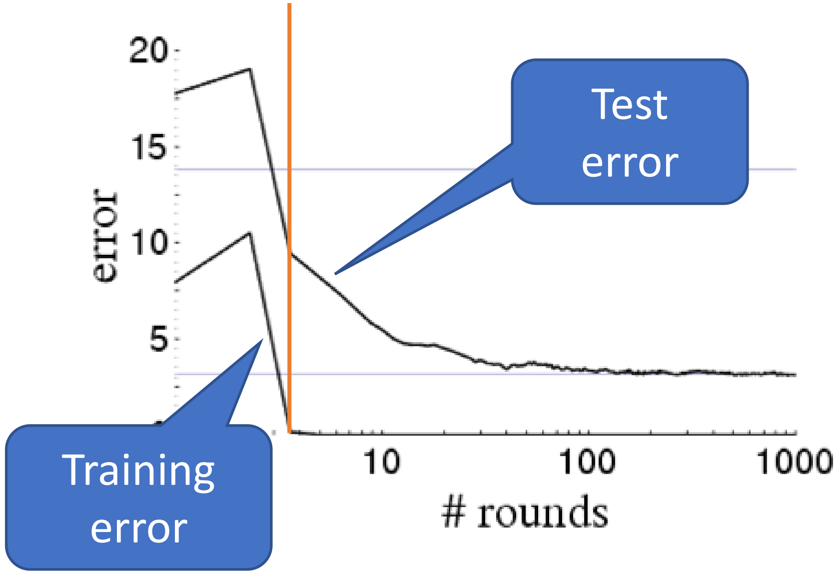
The Annals of Statistics
1998, Vol. 26, No. 5, 1651–1686

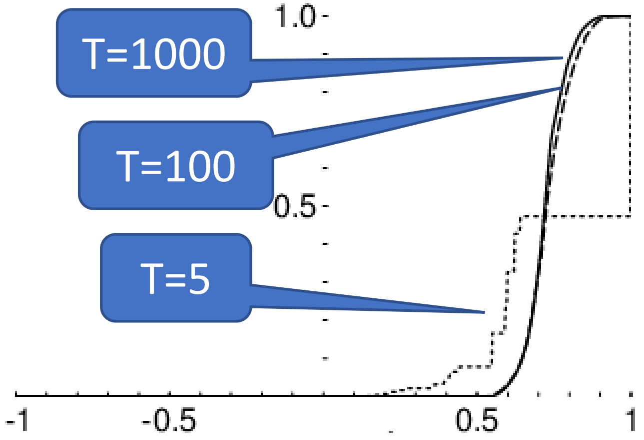
BOOSTING THE MARGIN: A NEW EXPLANATION FOR THE EFFECTIVENESS OF VOTING METHODS

By Robert E. Schapire, Yoav Freund, Peter Bartlett and Wee Sun Lee

AT&T Labs, AT&T Labs, Australian National University and Australian Defence Force Academy







$$\frac{y_i\langle w,\phi(x_i)\rangle}{\|w\|_1} = \frac{\sum \alpha_t h_t(x)}{\sum \alpha_t}$$

$$\overline{h}_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x) = \langle w,\phi(x)\rangle$$

$$\|w\|_1 = \sum_i |w[i]| = \sum_t \alpha_t$$

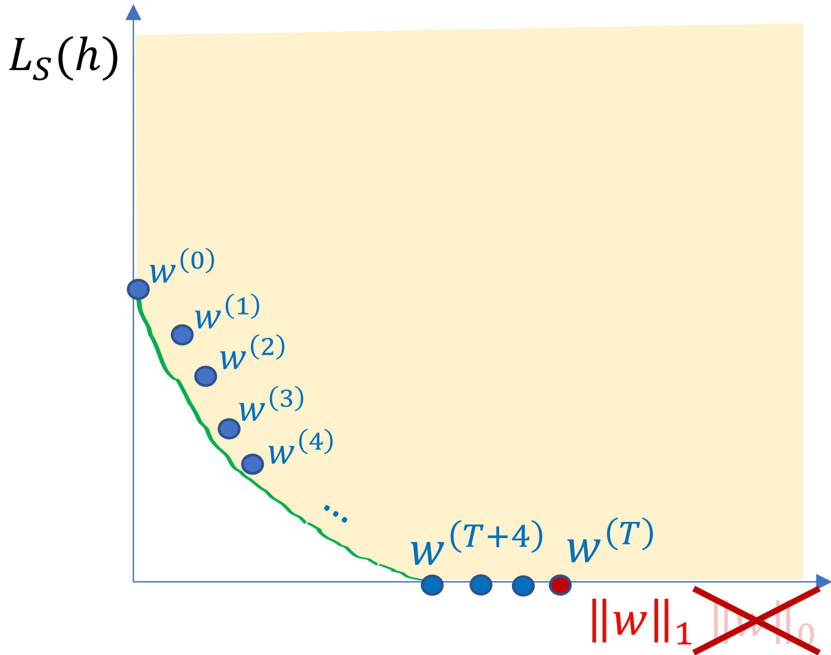
- Even after $L_S^{01}\left(\overline{h}_T(x)\right) = 0$, AdaBoost keeps improving the *margin*, and hence generalization
- But it's a ℓ_1 margin and not ℓ_2 margin as in SVM…

The Annals of Statistics
1998, Vol. 26, No. 5, 1651–1686

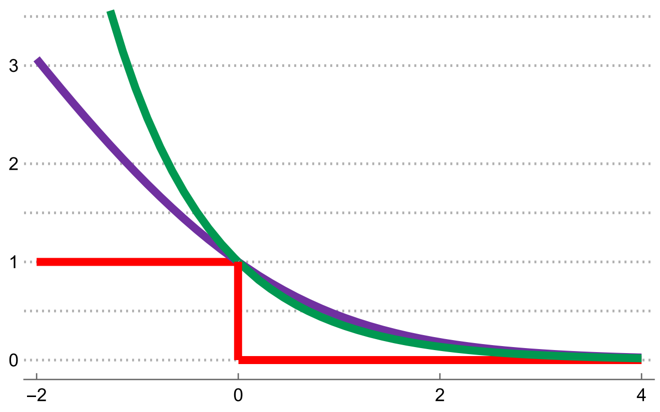
BOOSTING THE MARGIN: A NEW EXPLANATION FOR THE EFFECTIVENESS OF VOTING METHODS

BY ROBERT E. SCHAPIRE, YOAV FREUND, PETER BARTLETT AND WEE SUN LEE

AT & T Labs, AT & T Labs, Australian National University and Australian Defence Force Academy



- Recall in SVM, we measured "margin" as $\frac{y_i\langle w,\phi(x_i)\rangle}{\|w\|_2}$
  - Geometrically: require $\|w\|_2 = 1$ so that $y_i\langle w, \phi(x_i)\rangle$ was *Euclidean* distance from seperator
  - Hard-Margin SVM: $\hat{\boldsymbol{w}}_{\boldsymbol{SVM}} = \min \|w\|_{\mathbf{2}}$ s.t. $\forall_i y_i\langle w, \phi(x_i)\rangle \geq 1$
  - Recall: GD (on L_S^{lgst} or L_S^{exp}) converges to $\lim \frac{w_T}{\|w_T\|} \propto \hat{\boldsymbol{w}}_{\boldsymbol{SVM}}$
  - Soft-Margin SVM: balanced $\|w\|_2$ and $L_S^{\text{hinge}}(w)$

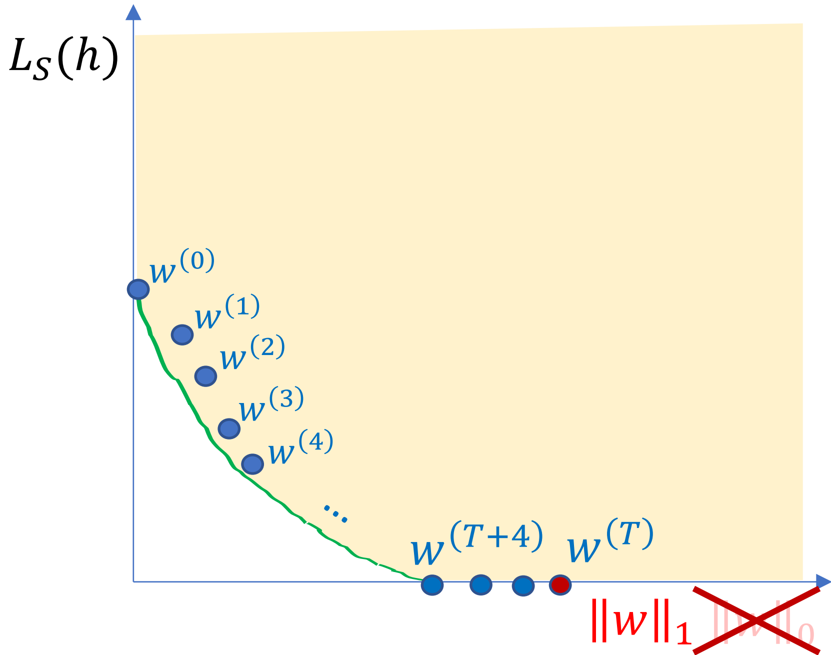


$\|w\|_1 = \sum_i |w[i]|$

- The ℓ_1 margin is measured as $\frac{y_i\langle w,\phi(x_i)\rangle}{\|w\|_1}$
  - Hard-Margin: $\hat{\boldsymbol{w}}_{\mathbf{1}} = \min \|w\|_{\mathbf{1}}$ s.t. $\forall_i y_i\langle w, \phi(x_i)\rangle \geq 1$
  - AdaBoost* converges to $\lim \frac{\bar{w}_T}{\|\bar{w}_T\|} \propto \hat{\boldsymbol{w}}_{\mathbf{1}}$

    Or coordinate descent on L_S^{lgst}

    Coordinate Descent on L_S^{exp}
  - Balance $\|w\|_1$ and $L_S^{\text{hinge}}(w)$ using explicit optimization



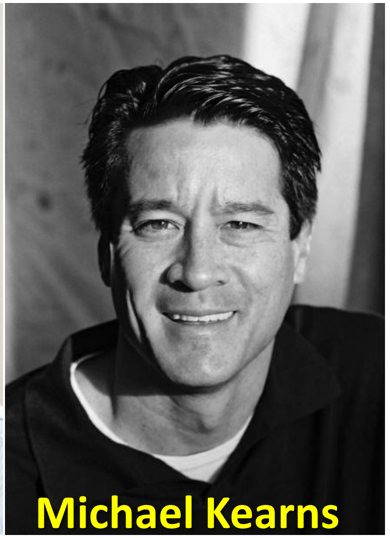
*AdaBoost might not converge, but slight variant with fixed "stepsize" α_t will

- Theoretical question (can we construct a strong learner from weak learner), Valiant and Kearns, 1988
- Solved by Rob Schapire, 1990

  Inductive bias: sparsity over base predictor
- AdaBoost Algorithm by Schapire and Yoav Fruend, AT&T 1995

  Interpretation as coordinate descent for exp-loss
- ℓ_1 margin interpretation, Schapire, Fruend, Bartlett and Lee, 1997

  Inductive bias: $\|w\|_1$ for linear comb of base predictor
- Viola and Jones Face Detector, 2001



Example: Viola-Jones Face Detector

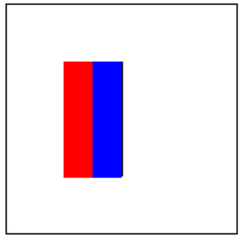
- Classify each square in an image as “face” or “no-face”

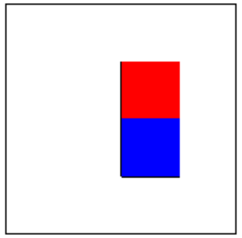
- We’ll consider all squares in an image, at many scales, of size at least 24x24 original pixels, and represent them as 24x24 grayscale pixels.
- $\mathcal{X}$ = patches of 24x24 grayscale pixels

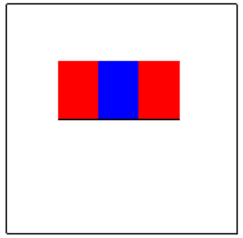
Viola-Jones “Weak Predictors”/Features

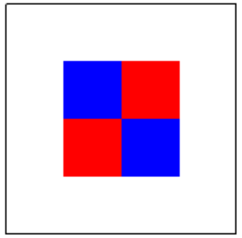
$$\mathcal{B} = \left\{ \left[\left[g_{r,t}(x) < \theta \right]\right] \mid \theta \in \mathbb{R}, \text{rect } r \text{ in image}, t \in \{A, B, C, D, \bar{A}, \bar{B}, \bar{C}, \bar{D}\} \right\}$$

where $g_{r,t}(x)$ =sum of “blue” pixels – sum of “red” pixels

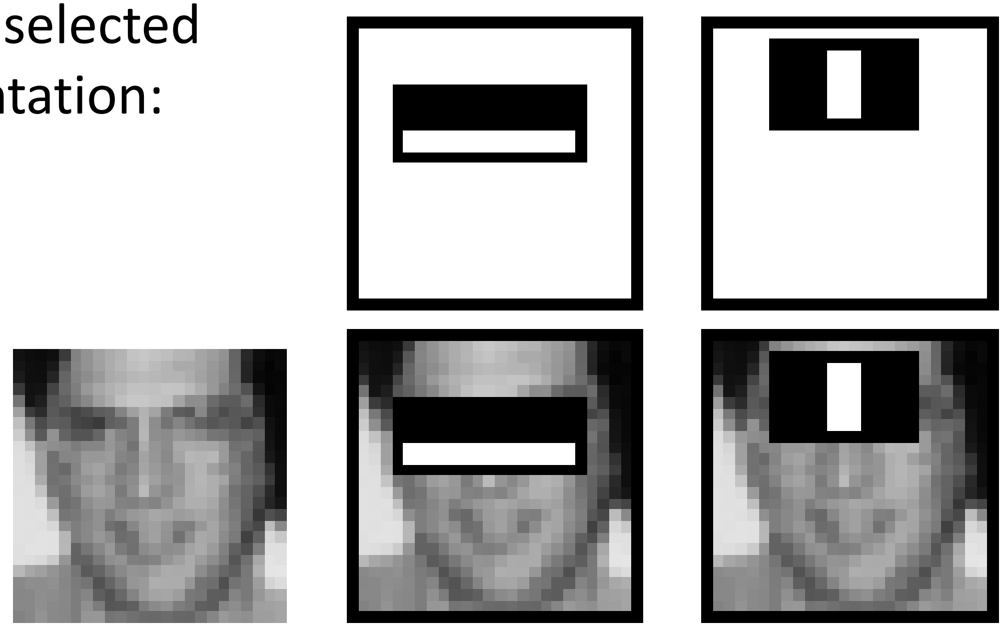
A: 

B: 

C: 

D: 

First two weak predictors h_1, h_2 selected in original Viola-Jones implementation:



Viola-Jones Face Detector

- Simple implementation of boosting using generic (non-face specific) “weak learners”/features
  - Can be used also for detecting other objects
- Efficient method using dynamic programing and caching to find good weak predictor
- About 1 million possible $g_{r,t}$, but only very few used in returned predictor
- Sparsity:
  - ➔Generalization
  - ➔**Prediction speed!** (and small memory footprint)
- To run in real-time (on 2001 laptop), use sequential evaluation
  - First evaluate first few h_t to get rough prediction
  - Only evaluate additional h_t on patches where the leading ones are promising

(and clever engineering in evaluating the required filters on all the patches in all scales….)

Boosting in Practice

- Frequently used with “decision stumps” (thresholds of features) or (smallish) decision trees (decision stump=single-node tree)
- Also: Linear classifiers (halfspaces), and specialized features (as in Viola-Jones)
- Or: a few boosting iterations on complex models (eg neural nets)

Interpretations of AdaBoost

- "Boosting" weak learning to get arbitrary small error
  - Our analysis relied on VCdim(outputs of weak learner), but what if weak learner is not $ERM_{\mathcal{B}}$?
    → actual proof uses "compression bounds" (beyond scope of course)
  - Theory is for realizable case, extending to non-realizable is a challenge
  - "Improper learning": we output a predictor not from the class $\mathcal{H}$ being learned, nor from the base class $\mathcal{B}$
- Ensemble method for combining many simpler predictors
  - E.g. combining decision stumps or decision trees
  - Other ensemble methods:
    bagging: combining $h_i = A(S_i)$ trained on ***random*** reweighting S_i of S, or with other randomness
    gating networks: use h_0 to decide whether to output h_1 or h_2
- Learning using ***sparse*** linear predictors with large (infinite?) dimensional feature space
  - Sparsity controls complexity **and speed**
  - Number of iterations controls sparsity ➔ early stopping as regularization
- Learning (in high dimensions) with large ℓ_1 margin
  - Converges to max ℓ_1 margin predictor, even after training error=0
  - $||w||_1$ controls complexity, learning guarantee in terms of ℓ_1 margin
- Coordinate-wise optimization of $L_S^{exp}(w)$
- "Forward greedy feature selection"

Regularized Risk Minimization

$$\arg\min_w \; R(w), \; L_S(h_w)$$

View 1: $R(w) = \|w\|_0$

- Complexity control is sparsity
- Non-convex, greedily optimize with forward selection

View 2: $R(w) = \|w\|_1$

- Better explains generalization behavior
- Connection to "weak learning" / having an edge under any distribution
- Converge to hard margin (MDL)

  $\arg\min\|w\|_1 \quad s.t.\ \langle w, \phi(x)\rangle \geq 1$
- Early Stopping ≈ regularization path

$$L_S(h) = L_S^{exp}(h)$$

Variations: other convex (surrogate) loss

$L_S^{\text{logistic}}, L_S^{\text{hinge}}, L_S^{\text{sq}}$

Optimization:

Coordinate descent

- Select coordinate $i = \arg\min \partial_i L_S(w)$
- Update $\arg\min_{\alpha_i} L_S(w + \alpha_i e_i)$

Variations:

- Fixed (small) stepsize α_i
- Add *and* remove features
- After adding each feature, reoptimize all α ("fully corrective")

Feature Selection

- Lots of features, i.e. very high dimensional $\phi(x) \in \mathbb{R}^D$
- Predict using few features $\phi(x)|_I \;\; I \subset D, |I| \ll D$
- In the context of linear prediction:
  $h_w = \langle w, \phi(x) \rangle, supp(w) \subseteq I$, i.e. $\|w\|_0 \ll D$

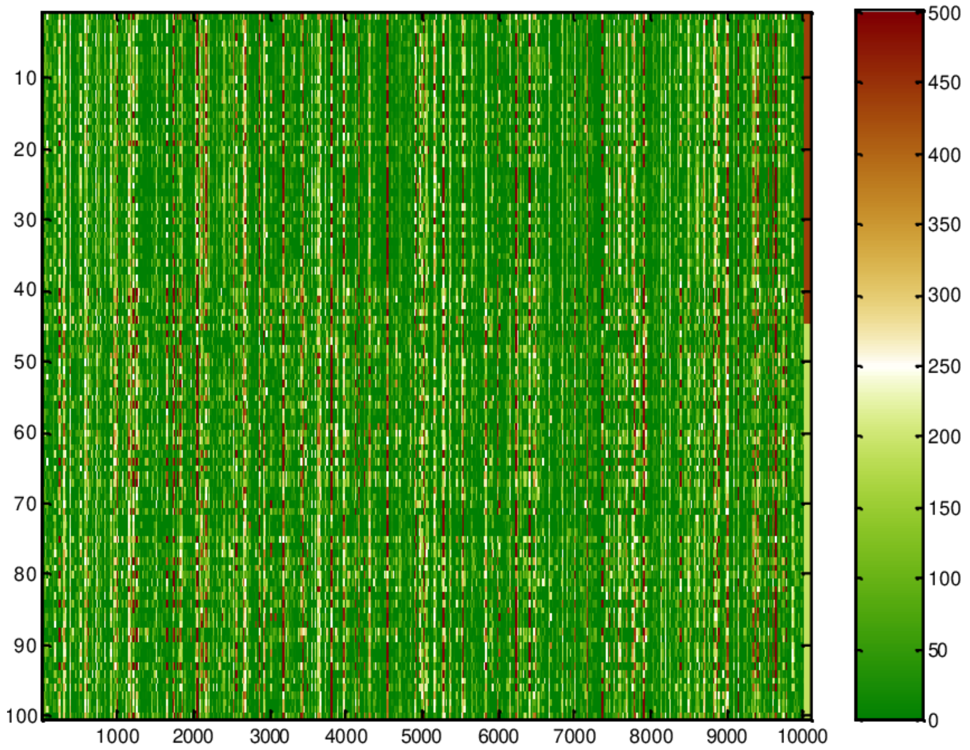
$\|w\|_0 = |\{i | w[i] \; \neq 0\}|$

➔ $\arg\min \; L_s(w) \quad , \;\; \|w\|_0$

E.g. $\quad \arg\min L_S(w) \;\;$ s.t. $\|w\|_0 \leq k$

$\quad \arg\min \|w\|_0 \;\;$ s.t. $L_S(w) = 0$

- NP-hard
- Only known method: enumerate over $\binom{D}{k} = O(D^k)$



$\phi(x)[i]$=abundance of protein i in blood
y= cancer?

Forward Greedy Selection (Coordinate Descent)

$$\arg\min \; L_S(w) \quad , \qquad \|w\|_0$$

Initialize $w^{(0)} = 0, I^{(0)} = \emptyset$

At each iteration k:

- **Find "good" feature $\boldsymbol{i}$**

  Highest directional derivative: $\arg\max \left| \frac{\partial L_S(w^{(t)})}{\partial w[i]} \right|$ [AdaBoost]

  Biggest benefit: $\underset{supp(w) \subseteq I \cup \{i\}}{\arg\min} \; L_S(w)$

  ...

- **Add feature: $\boldsymbol{I^{(k+1)} = I^{(k)} \cup \{i\}}$**
- **Update $\boldsymbol{w}$ to include $\boldsymbol{w[i]}$**

  Incrementally: $w^{(k+1)} = \arg\min L_S(w) \;\; s.t. \; \forall_{i' \neq i} w[i'] = w^{(k)}[i']$ [AdaBoost]

  Fully Corrective: $w^{(k+1)} = \arg\min L_S(w) \;\; s.t. \; \mathrm{supp}(w) \subseteq I^{(k+1)}$

  ...

Variations: Also allow removing (pruning) or replacing features
Consider groups of 2 or 3 features at a time

Forward Greedy Selection as a "Wrapper" for $A(S)$

Initialize $I^{(0)} = \emptyset$

At each iteration k:

- **Find "good" feature $\boldsymbol{i}$**

  Biggest empirical benefit: $\underset{supp(w) \subseteq I \cup \{i\}}{\arg\min} \; L_S(A(S|_I))$

  Best validation benefit: $\underset{supp(w) \subseteq I \cup \{i\}}{\arg\min} \; L_{S_{val}}(A(S_{train}|_I))$ (or cross validation)

- **Add feature: $\boldsymbol{I^{(k+1)} = I^{(k)} \cup \{i\}}$**

Variations: Also allow removing (pruning) or replacing features
Consider groups of 2 or 3 features at a time
Start with I ={all features} are remove features gradually

Convex Surrogate: L_1 Regularization

$$\arg\min_w L_S(w) \quad , \quad \|w\|_0$$

$\|w\|_1 = \sum_i |w[i]|$

- Original Lasso: $\ell^{\text{sq}}(\langle w, \phi(x)\rangle, y) = \frac{1}{2}(y - \langle w, \phi(x)\rangle)^2$; could use any other loss functions
- L_1 regularization:
  - Ensures generalization (effective dimension $\propto \|w\|_1^2 \|\phi\|_\infty^2 \log D$)
  - Can be thought of as convex surrogate for sparsity
  - Induces sparsity due to non-differentiability at 0

If $\phi(x) \in \{\pm 1\}^D$: $\|\phi(x)\|_2^2 = D$
but $\|\phi(x)\|_\infty^2 = \max_i |\phi(x)[i]| = 1$

ℓ_1

$$\|(1,0,\ldots,0)\|_1^2 \ll \left\|\left(\frac{1}{\sqrt{D}},\frac{1}{\sqrt{D}},\ldots,\frac{1}{\sqrt{D}}\right)\right\|_1^2 = D$$

ℓ_2

$$\|(1,0,\ldots,0)\|_2^2 = \left\|\left(\frac{1}{\sqrt{D}},\frac{1}{\sqrt{D}},\frac{1}{\sqrt{D}},\ldots,\frac{1}{\sqrt{D}}\right)\right\|_2^2 = 1$$

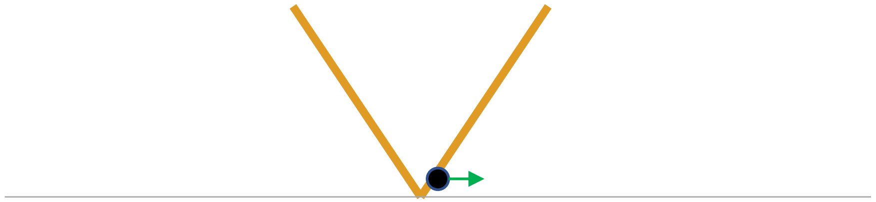
$$F_1(w) = L_S(w) + \lambda\|w\|_1$$

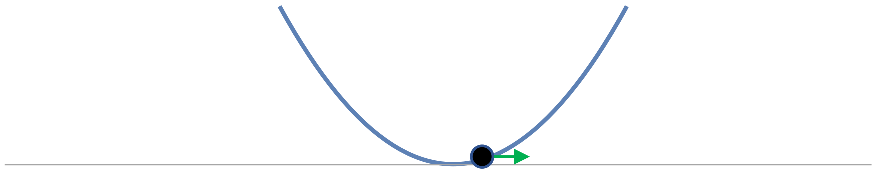
$$F_2(w) = L_S(w) + \lambda\|w\|_2^2$$

what happens when $w[i]$ is already very close to zero:

$$\partial_i F_w(w) = \partial_i L_S(w) + \lambda\, sign(w[i])$$

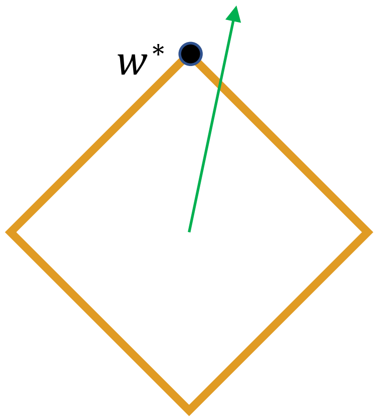
$$\partial_i F_w(w) = \partial_i L_S(w) + 2\lambda\, w[i]$$

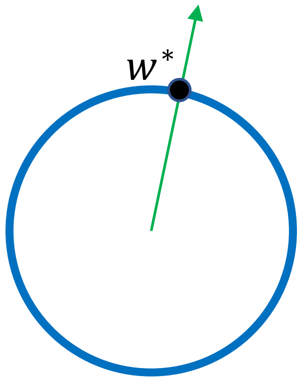




$$\arg\min L_S(w) \quad s.t.\ \|w\|_1 \le B$$

$$\arg\min L_S(w) \quad s.t.\ \|w\|_2 \le B$$





Why do Feature Selection?

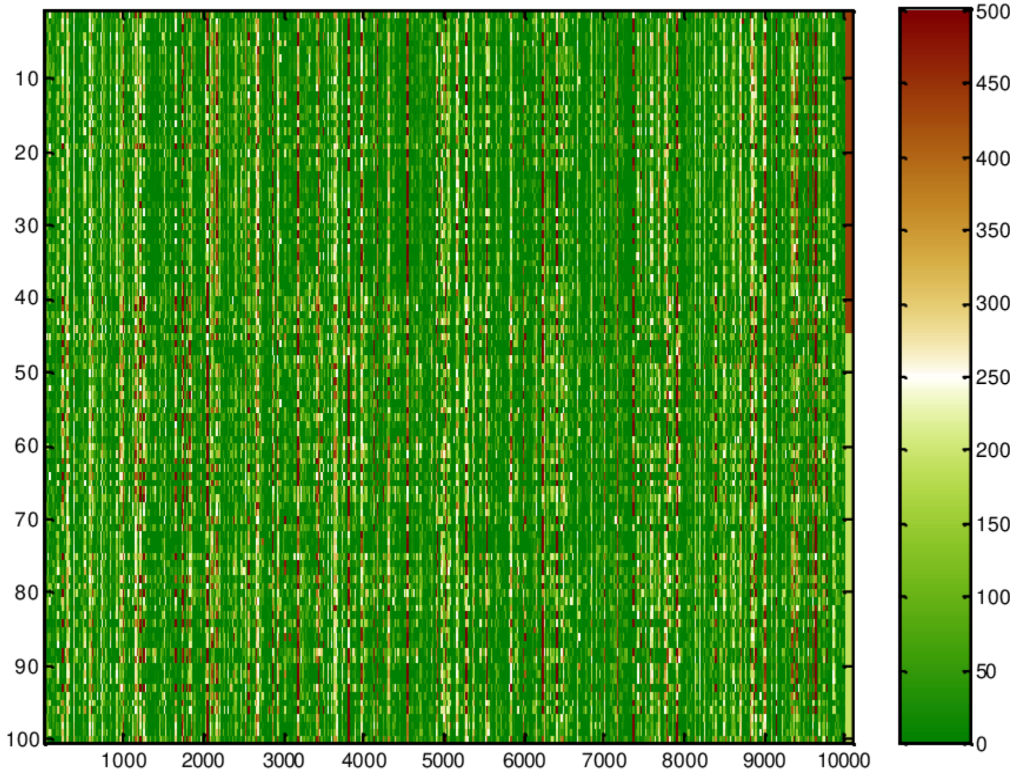
- We want to know what the relevant features are

  ➔ not a learning goal

- Inductive bias / complexity control

  ➔ enough to compete with best sparse predictor:

$$L_{\mathcal{D}}(\hat{w} = A(S)) \leq \inf_{\|w\|_0 \leq k} L_{\mathcal{D}}(w) + \epsilon$$



$\phi(x)[i]$=abundance of protein i in blood

y= cancer?

Why do Feature Selection?

- We want to know what the relevant features are
  - ➔ not a learning goal
- Inductive bias / complexity control
  - ➔ enough to compete with best sparse predictor:

$$L_{\mathcal{D}}(\hat{w} = A(S)) \leq \inf_{\|w\|_0 \leq k} L_{\mathcal{D}}(w) + \epsilon$$

  - ℓ_1 regularized learning often does this, even if not sparse
  - Bayesian approach: integrate over posterior (over uncertainty) ➔ dense predictor
  - Don't need to worry about getting ONLY correct features (especially If there are many features correlated with the ``correct'' feature)
- Small memory footprint of predictor and fast prediction runtime
  - ➔ often better to learn dense predictor (even if reality sparse), then try to sparsify while preserving accuracy (as much as possible)

Method (Team)	test error	$\%\|w\|_0$	% decoy features
BayesNN-DFT (*Neal/Zhang*	6.48	80.3	47.8
BayesNN-large (*Neal*)	7.27	60.3	28.5
BayesNN-small (*Neal*)	7.13	4.7	2.9
final_2-3 (*Chen*)	7.91	24.9	9.9
BayesNN-large (*Neal*)	7.83	60.3	28.5
final2-2 (*Chen*)	8.80	24.6	6.7
Ghostminer1 (*Ghostminer*)	7.89	80.6	36.1
RF+RLSC (*Torkkola/Tuv*)	8.04	22.4	17.5
Ghostminer2 (*Ghostminer*)	7.86	80.6	36.1
RF+RLSC (*Torkkola/Tuv*)	8.23	22.4	17.5
FS+SVM (*Lal*)	8.99	20.9	17.3
Ghostminer3 (*Ghostminer*)	8.24	80.6	36.1
CBAMethod3E (*CBAGroup*	8.14	12.8	0.1
CBAMethod3E (*CBAGroup*	8.14	12.8	0.1
Nameless (*Navot/Bachrach*)	7.78	32.3	16.2

Feature Selection vs Feature Learning

- Feature Selection: from a given set of features $\{\phi(x)[i]\}$ select a subset $\{\phi(x)[i] | i \in I\}$ to use
- Feature Learning: method to construct *new* features $\psi(x)$

  …based on x, i.e. $\psi(x)[i] = g_i(x)$

  E.g. linear combinations of features, products or monomials in the features, other combinations of features

  …from a class $\mathcal{B} = \{g: \mathcal{X} \to \mathbb{R}\}$ of possible "feature generators"

  E.g. linear functions, stumps, small decision trees, …

  $\equiv$ "feature selection" from $\phi(x) \in \mathbb{R}^{\mathcal{B}}$, $\phi(x)[g] = g(x)$

  ➔ Selection from infinitely many features, but that doesn't scare us!

- Either way: key is sparsity, sparsity-related complexity control/generalization, or a sparsity inducing regularizer/constraint