AdaBoost

Lecturer:
Jiří Matas

Authors:
Jan Šochman, Jiří Matas,
Jana Kostlivá, Ondřej Drbohlav

Centre for Machine Perception
Czech Technical University, Prague
http://cmp.felk.cvut.cz

Last update: 16.11.2017

AdaBoost

Presentation outline

- AdaBoost algorithm
  - Why is it of interest?
  - How it works?
  - Why it works?
- AdaBoost variants

History

- 1990 – Boost-by-majority algorithm (Freund)
- 1995 – AdaBoost (Freund & Schapire)
- 1997 – Generalized version of AdaBoost (Schapire & Singer)
- 2001 – AdaBoost in Face Detection (Viola & Jones)

What is Discrete AdaBoost?

AdaBoost is an algorithm for designing a *strong* classifier $H(\mathbf{x})$ from *weak* classifiers $h_t(\mathbf{x})$ $(t = 1, ..., T)$ selected from the weak classifier set $\mathcal{B}$. The strong classifier $H(\mathbf{x})$ is constructed as:

$$H(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x}))\,,$$

where

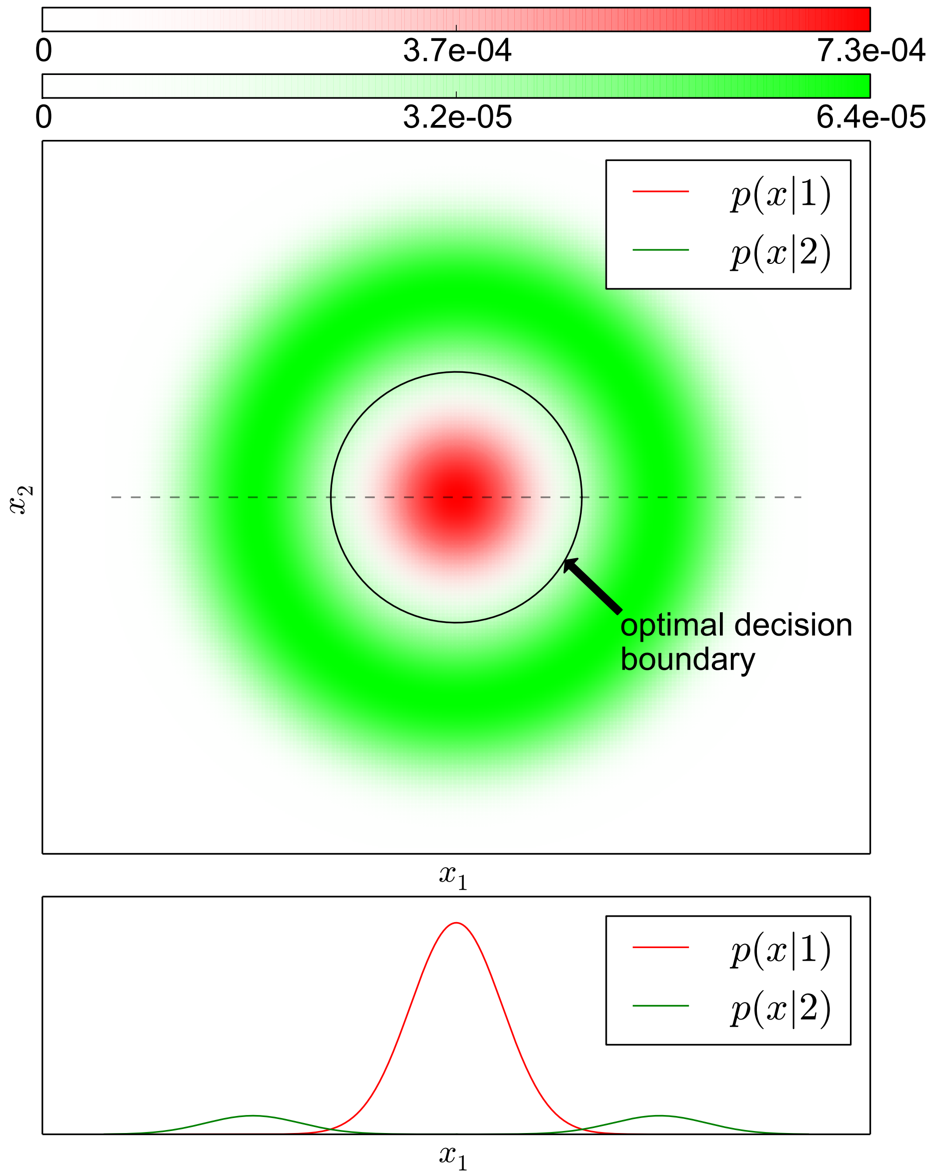
$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

is a linear combination of weak classifiers $h_t(\mathbf{x})$ with positive weights $\alpha_t > 0$. Every weak classifier h_t is a binary classifier which outputs -1 or 1.

Adaboost deals both with the selection of $h_t(\mathbf{x}) \in \mathcal{B}$, and with choosing α_t, for gradually increasing t.

The set of weak classifiers $\mathcal{B} = \{h(\mathbf{x})\}$ can be finite or infinite.

Example 1 – Training Set and Weak Classifier Set



the profile of the distributions along the shown line

Training set:

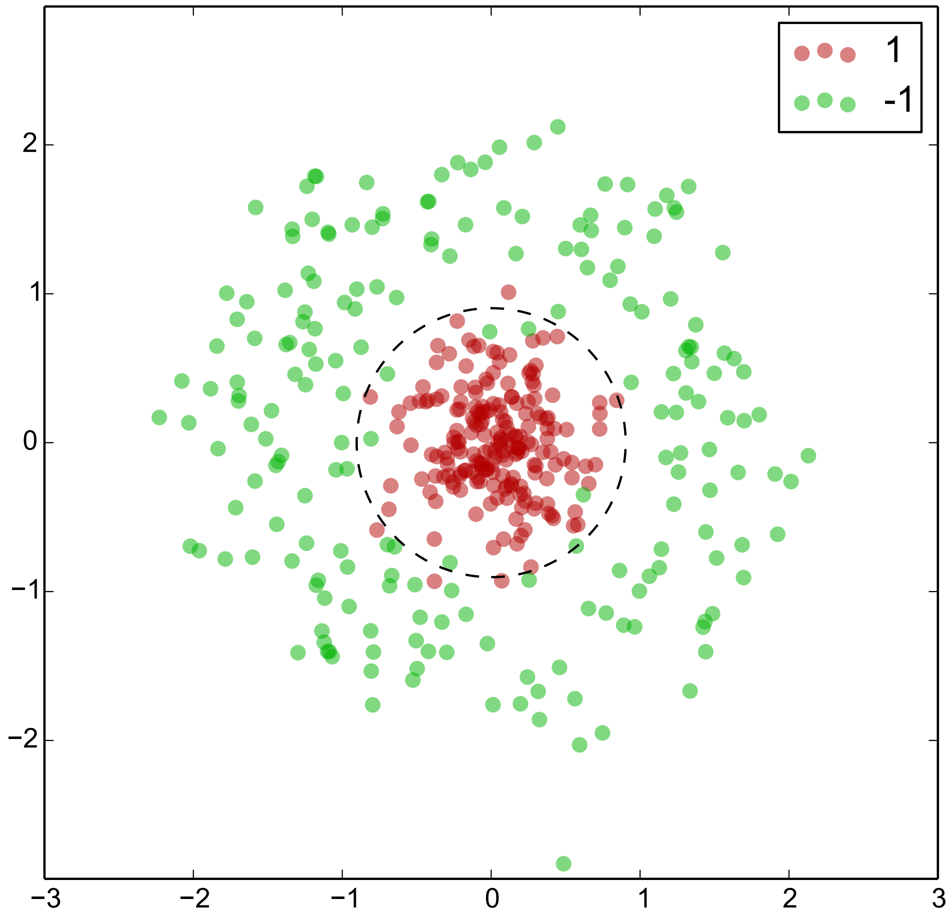
Samples generated from the two distributions shown, with

$$p(1) = p(2) = 0.5 \tag{1}$$

The Bayes error is 2.6%.

In the slides to follow, the classes are renamed from $(1, 2)$ to $(1, -1)$.

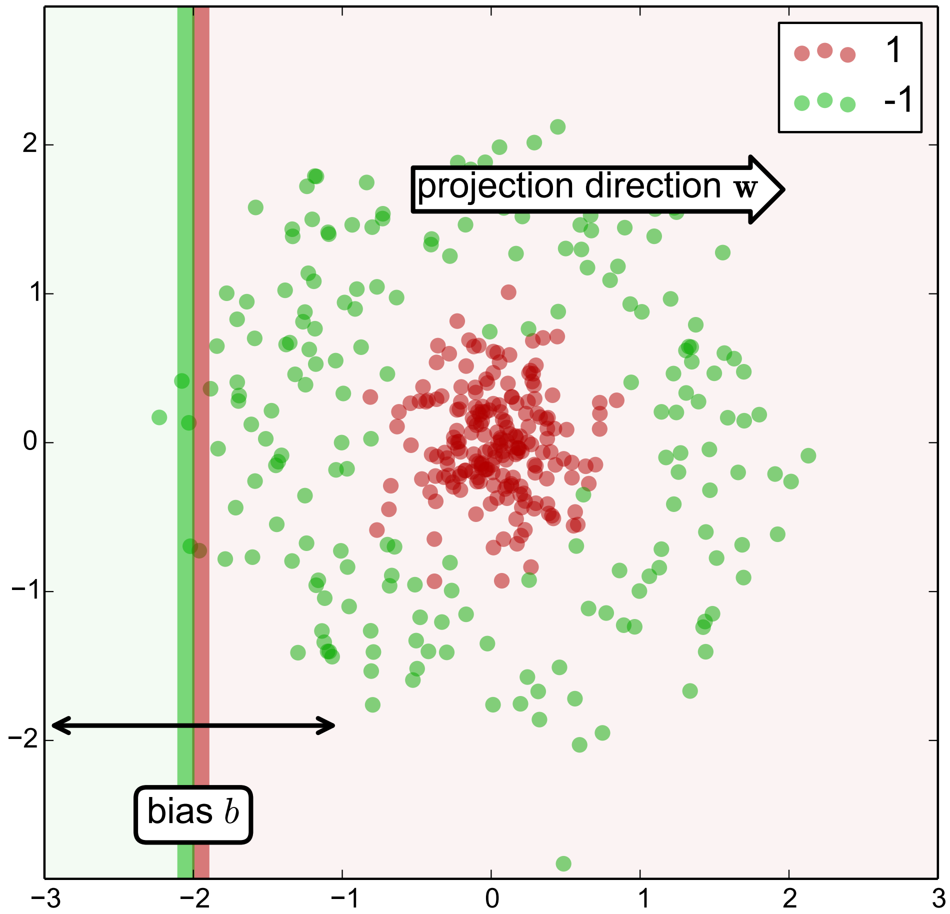
Example 1 – Training Set and Weak Classifier Set



Training set:
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Example 1 – Training Set and Weak Classifier Set



Training set:
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

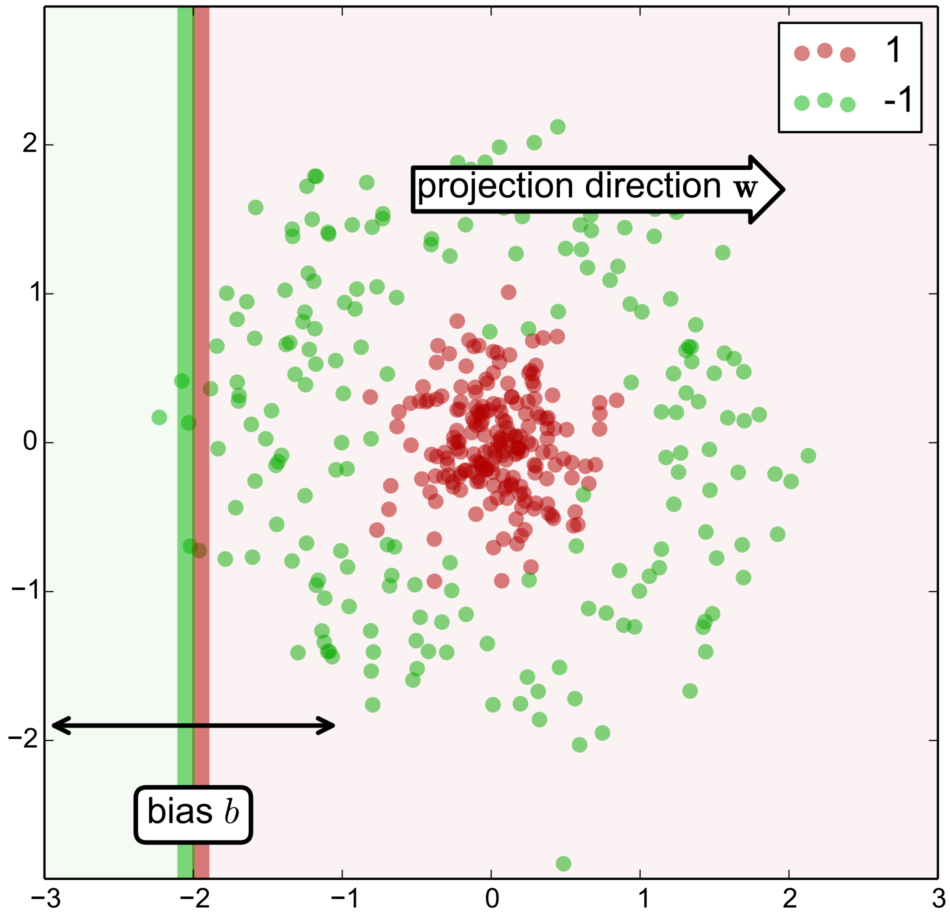
The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sign}(\mathbf{w} \cdot \mathbf{x} + b) ,$$

where $\mathbf{w}$ is the projection direction vector and b is the bias.

Example 1 – Training Set and Weak Classifier Set



Training set:
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sign}(\mathbf{w} \cdot \mathbf{x} + b)\,,$$

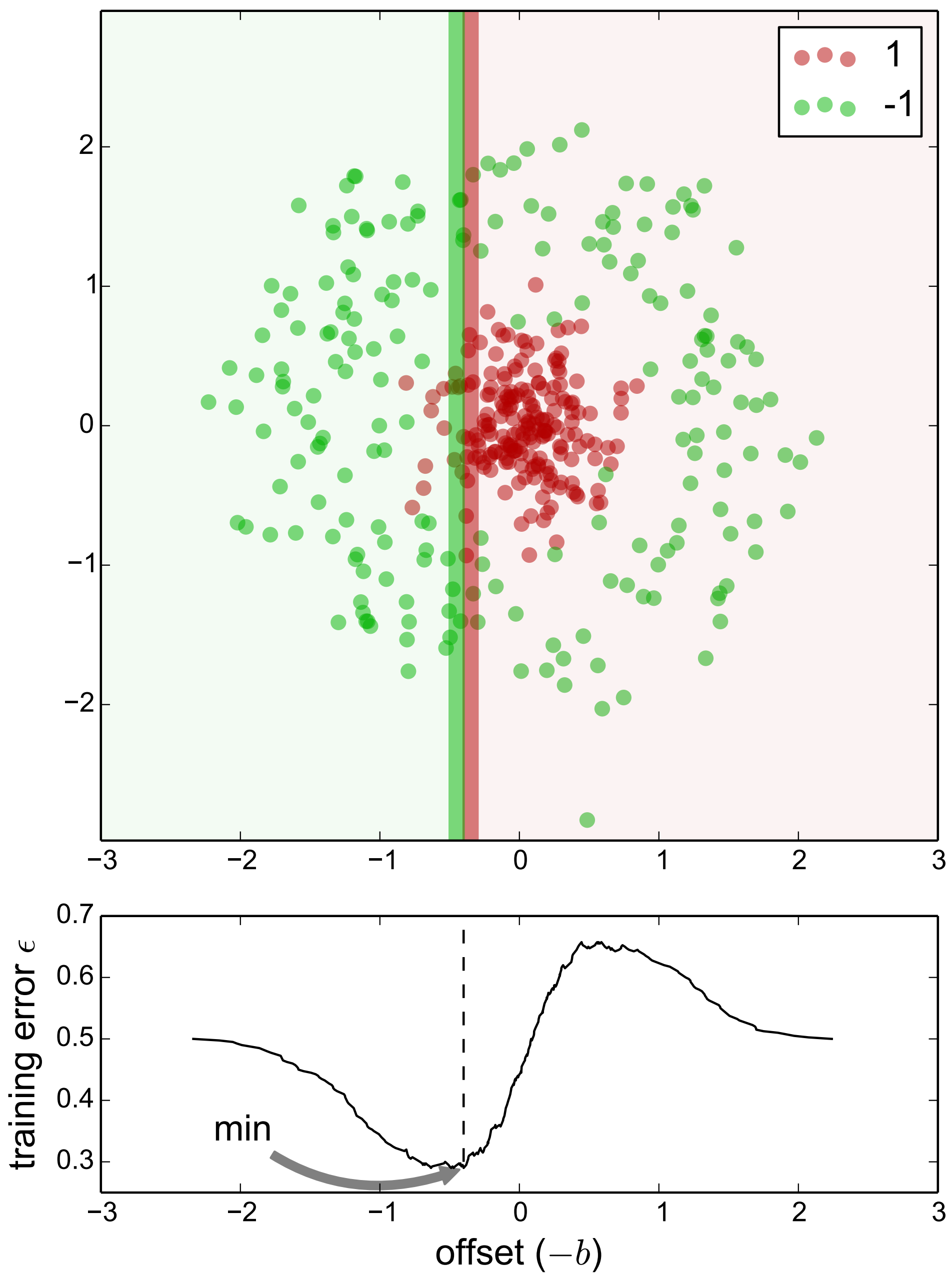
where $\mathbf{w}$ is the projection direction vector and b is the bias.

Weak classifier set $\mathcal{B}$:

$$\{h_{\mathbf{w},b} \mid \mathbf{w} \in \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}, b \in \mathbb{R}\}$$

- N is the number of projection directions used

Example 1 – Training Set and Weak Classifier Set



Training set: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \,,$$

where $\mathbf{w}$ is the projection direction vector and b is the bias.

Weak classifier set $\mathcal{B}$:

$$\{h_{\mathbf{w},b} \mid \mathbf{w} \in \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N\}, b \in \mathbb{R}\}$$

- N is the number of projection directions used
- for each projection direction $\mathbf{w}$, varying bias b results in different training errors ϵ.

AdaBoost Algorithm – Singer & Schapire (1997)

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, 1\}$

Initialize data weights $D_1(i) = 1/L$.

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\quad \epsilon_t = \sum_{i=1}^{L} D_t(i) \, [\![y_i \neq h(\mathbf{x}_i)]\!]$ (*WeakLearn*)

  $\uparrow$

  $[\![\mathrm{true}]\!] \stackrel{\mathrm{def}}{=} 1, \quad [\![\mathrm{false}]\!] \stackrel{\mathrm{def}}{=} 0$

- If $\epsilon_t \geq \frac{1}{2}$ then stop
- Set $\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$ **Note**: $\alpha_t > 0$ because $\epsilon_t < \frac{1}{2}$
- Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}, \qquad Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)},$$

  where Z_t is a normalization factor chosen so that D_{t+1} is a distribution.

Output the final classifier:

$$H(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x})), \qquad f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

Note: Data Reweighting

$D_t(i)$: previous data weights
$D_{t+1}(i)$: new data weights

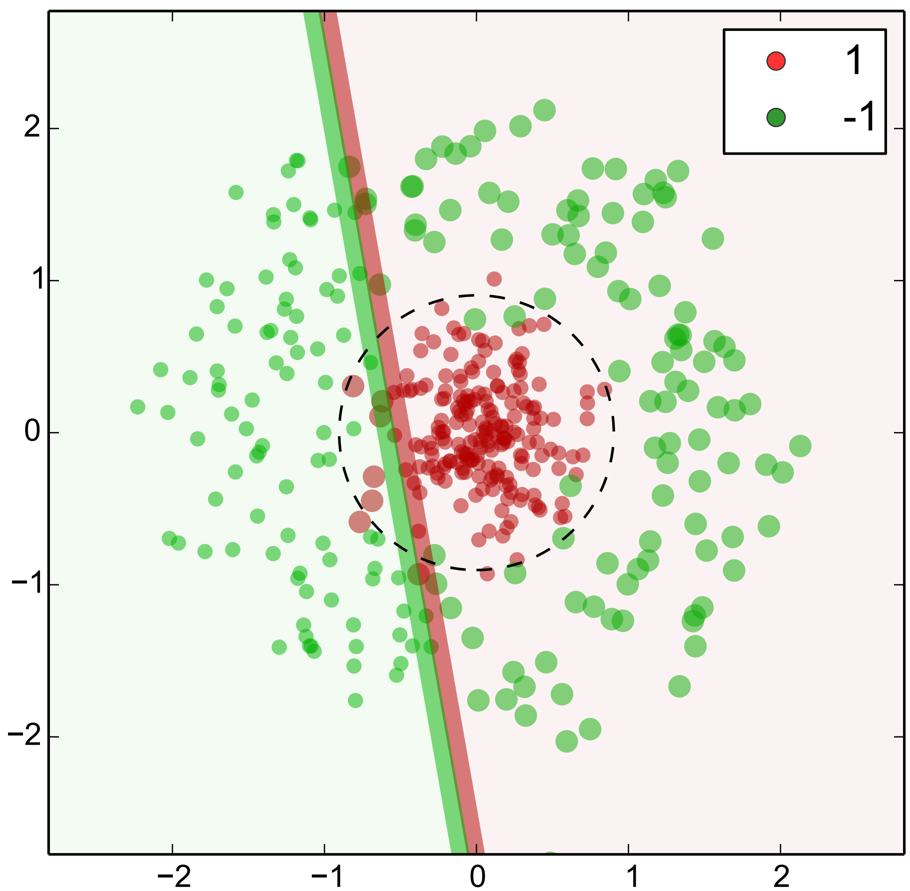
Weight update (recall that $\alpha_t > 0$):

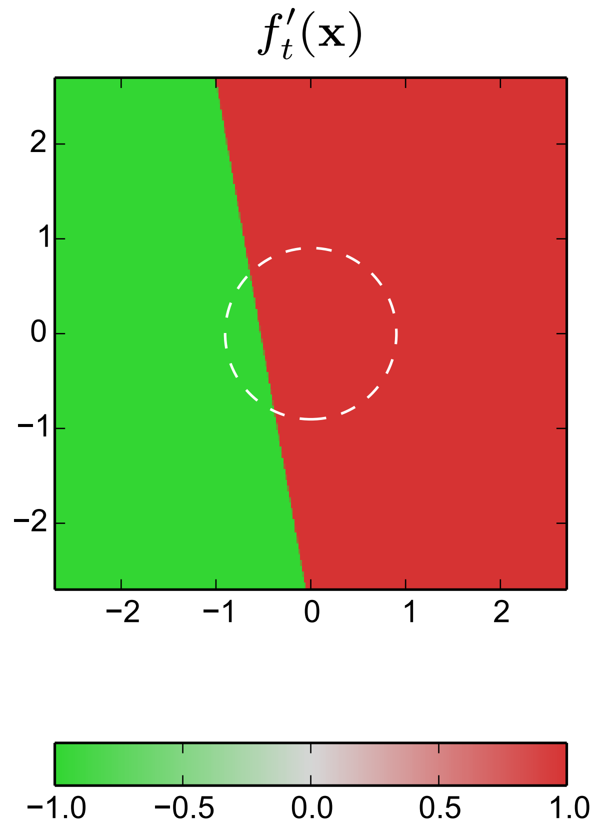
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t} \tag{2}$$

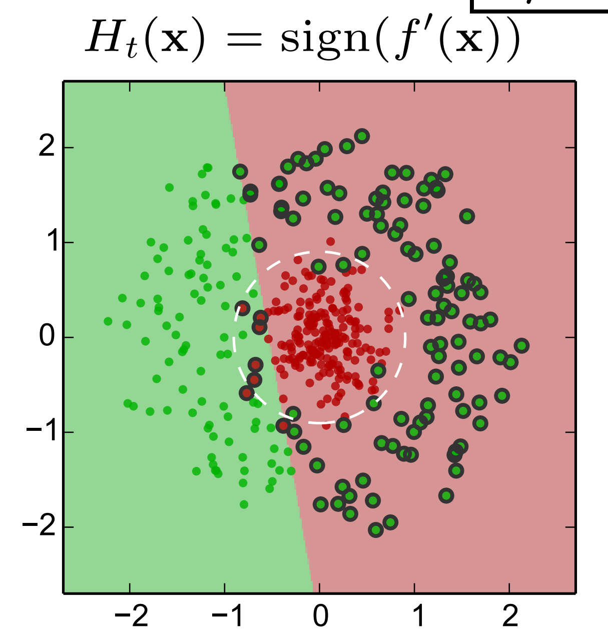
$$e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = \begin{cases} e^{-\alpha_t} & < 1, \quad \text{if } y_i = h_t(\mathbf{x}_i) \\ e^{\alpha_t} & > 1, \quad \text{if } y_i \neq h_t(\mathbf{x}_i) \end{cases} \tag{3}$$

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples.

Example 1 – iteration 1

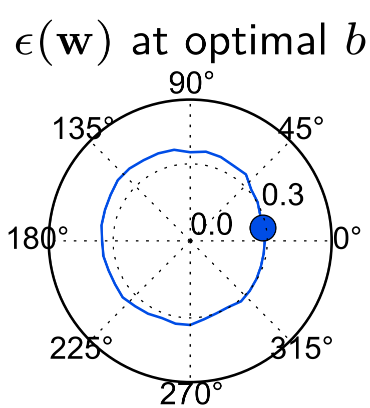


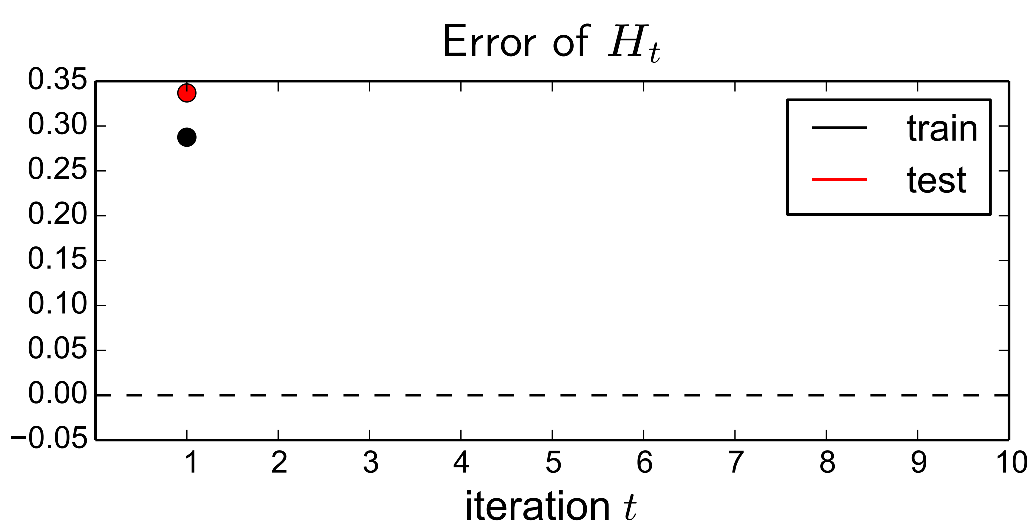




$$\epsilon_t = 28.8\%$$

$$\alpha_t = 0.454$$



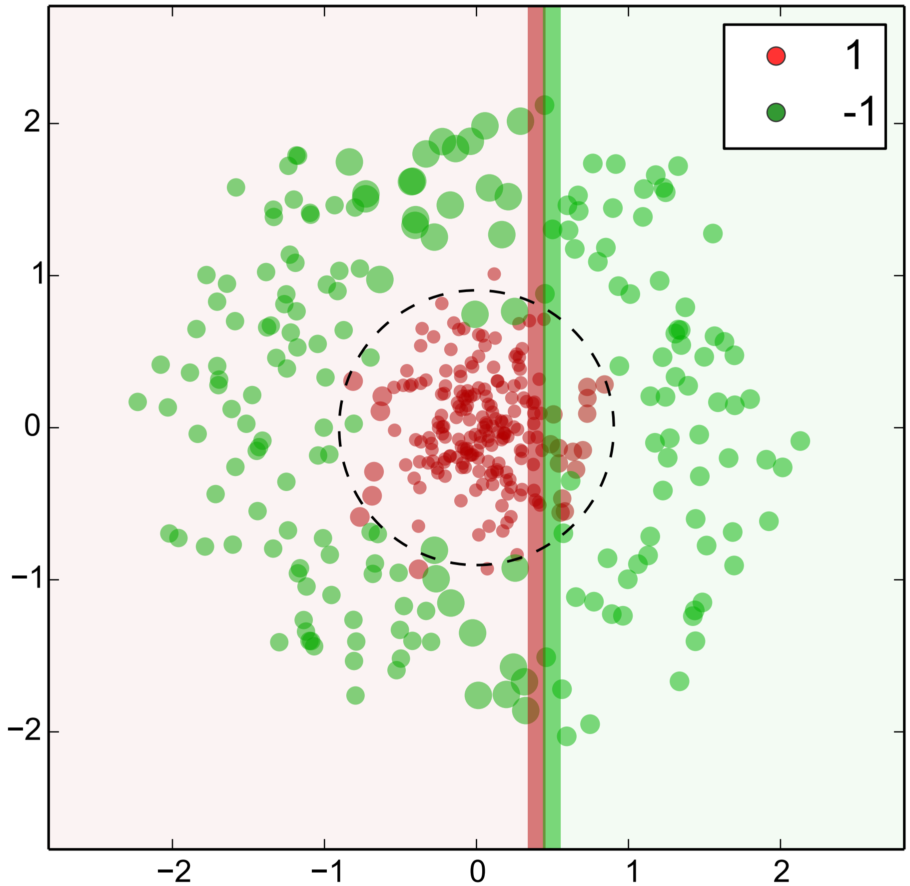


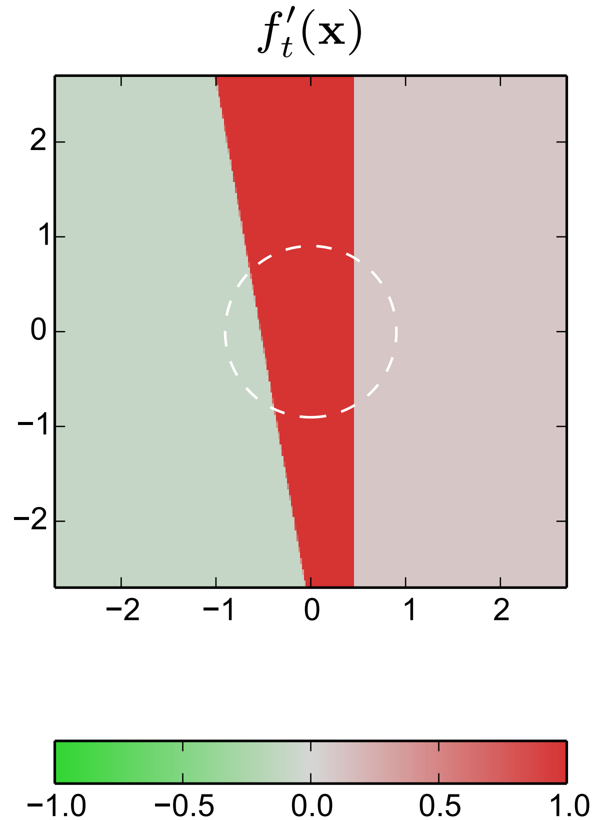
$$\epsilon_{H_t}^{\text{train}} = 28.7\%$$

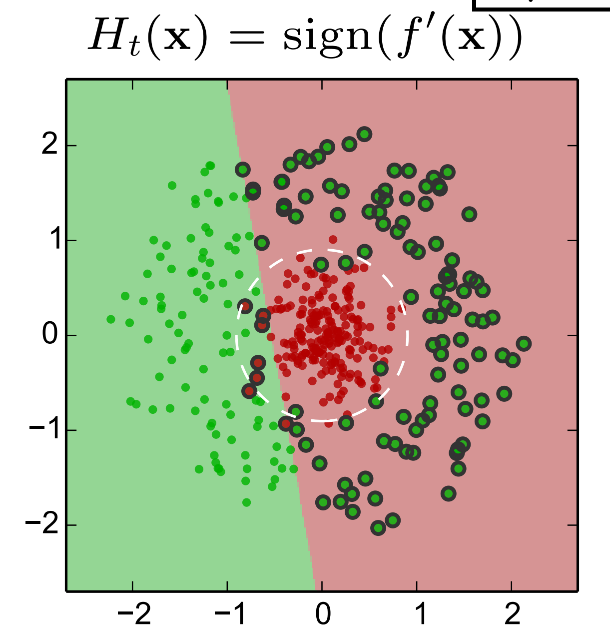
$$\epsilon_{H_t}^{\text{test}} = 33.7\%$$

$$Z_t = 0.905$$

Example 1 – iteration 2

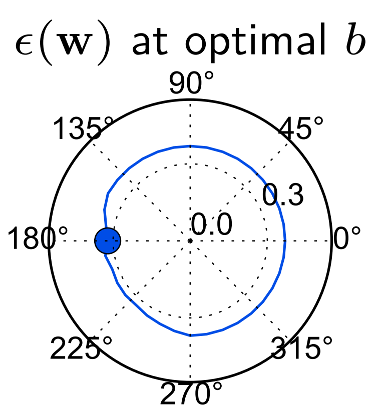


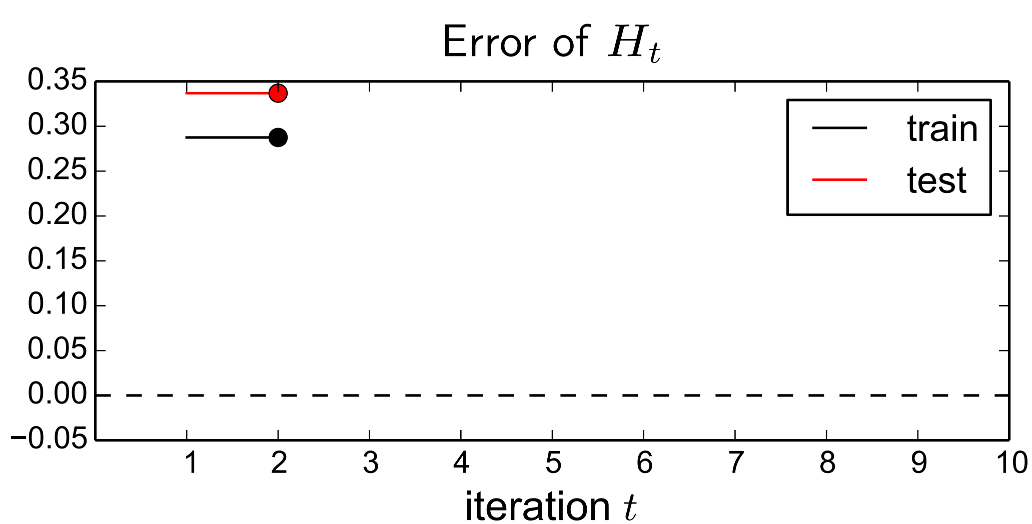




$$\epsilon_t = 32.1\%$$

$$\alpha_t = 0.375$$



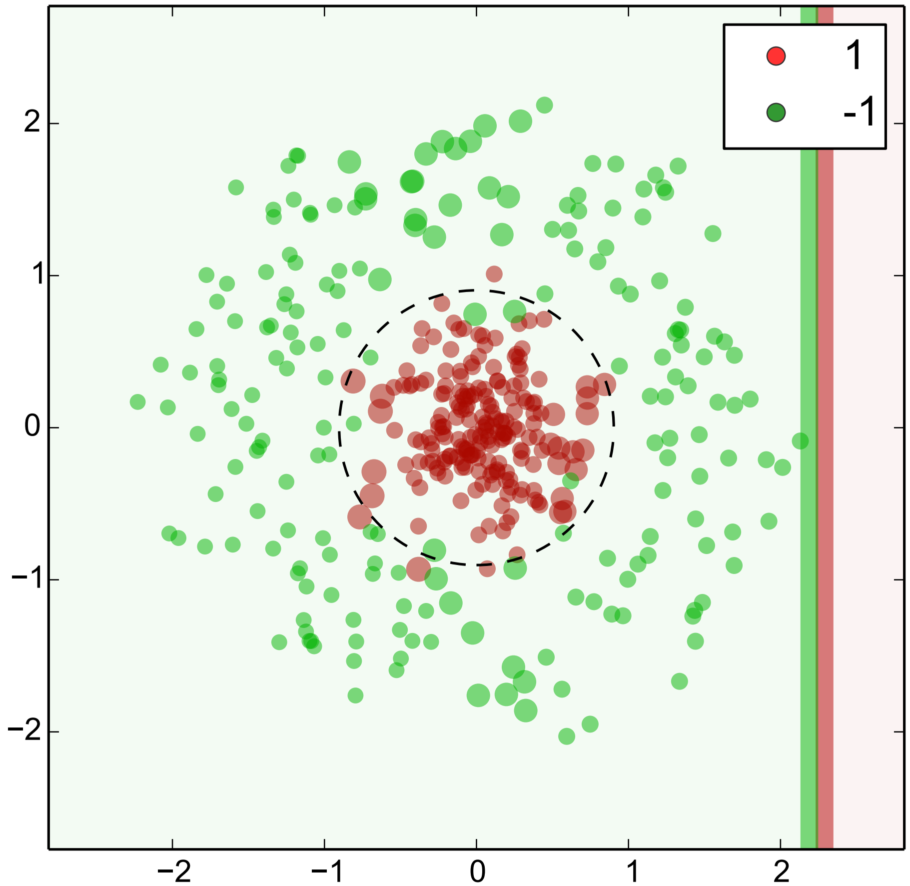


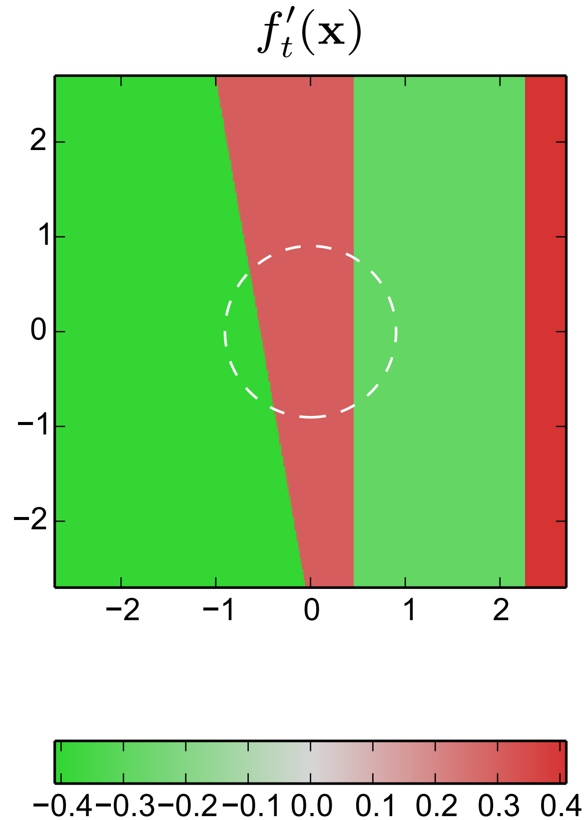
$$\epsilon_{H_t}^{\text{train}} = 28.7\%$$

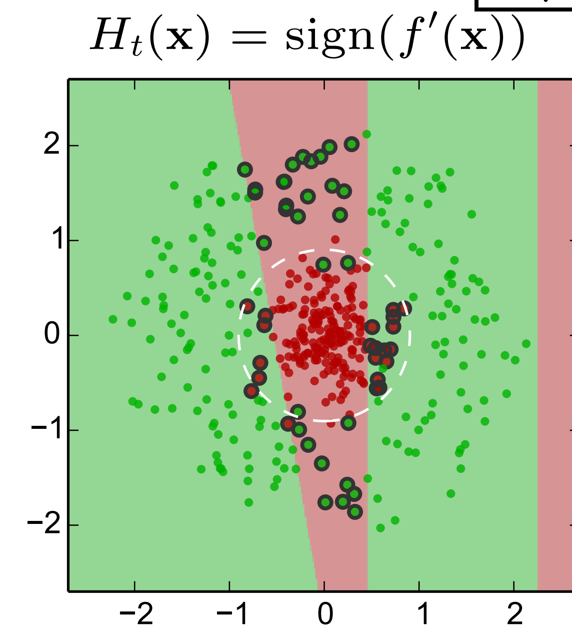
$$\epsilon_{H_t}^{\text{test}} = 33.7\%$$

$$Z_t = 0.934$$

Example 1 – iteration 3

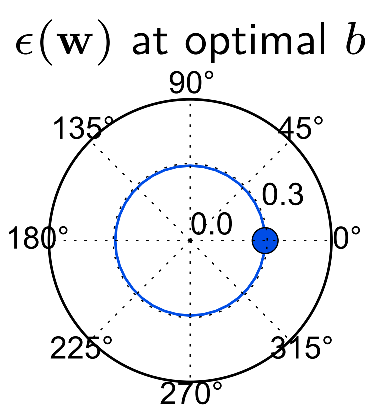


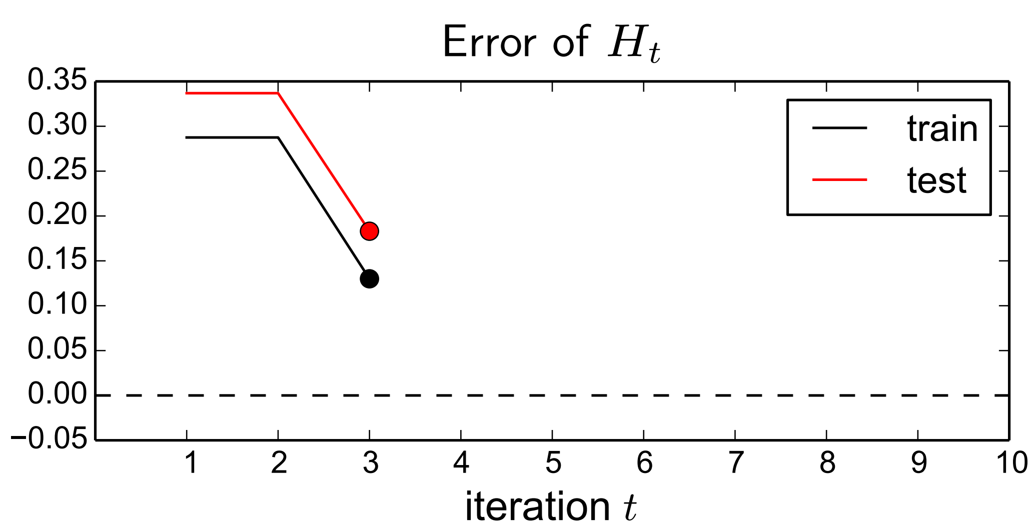




$$\epsilon_t = 29.2\%$$

$$\alpha_t = 0.443$$



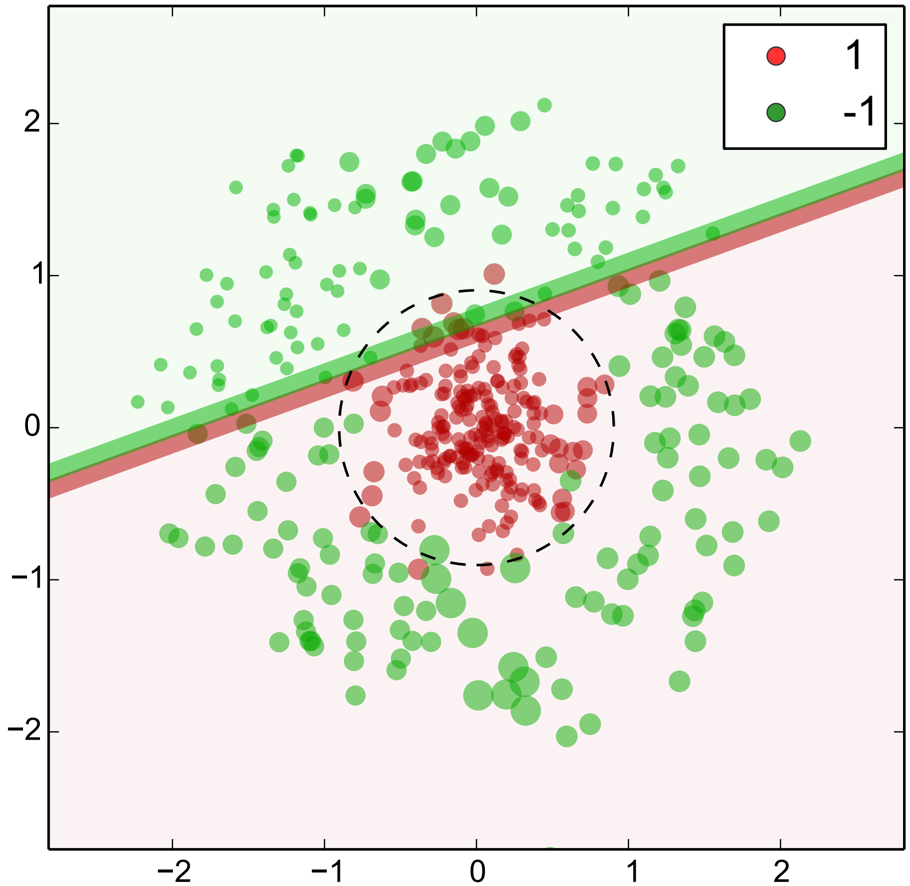


$$\epsilon_{H_t}^{\text{train}} = 13.0\%$$

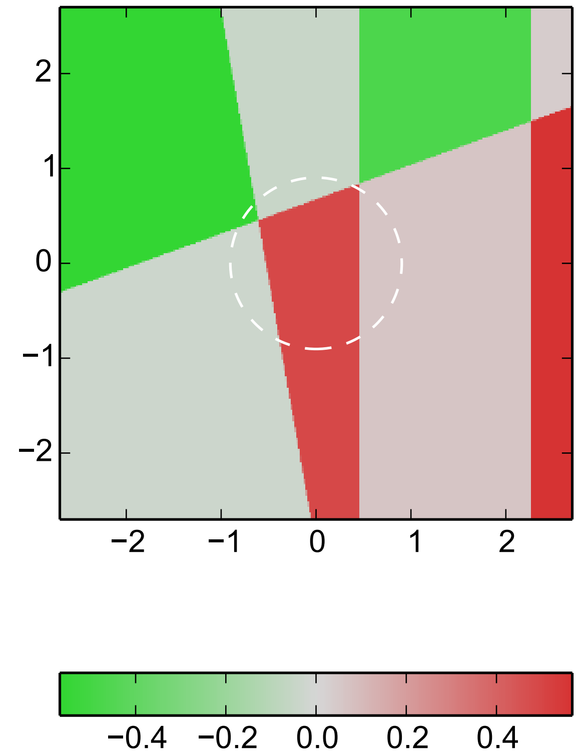
$$\epsilon_{H_t}^{\text{test}} = 18.3\%$$

$$Z_t = 0.909$$

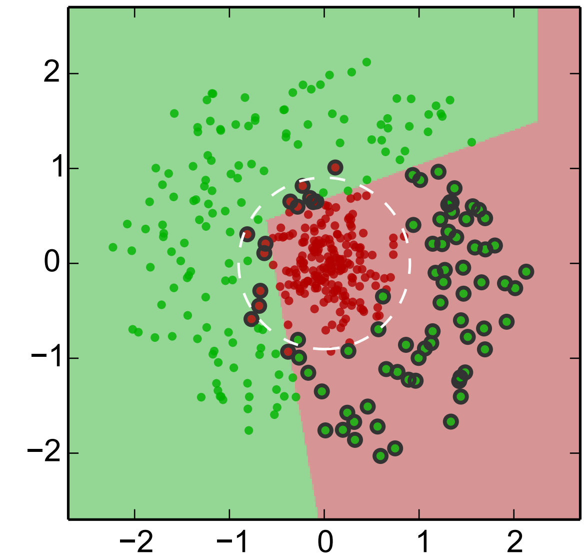
Example 1 – iteration 4



$f'_t(\mathbf{x})$



$H_t(\mathbf{x}) = \mathrm{sign}(f'(\mathbf{x}))$



$\epsilon_t = 28.3\%$

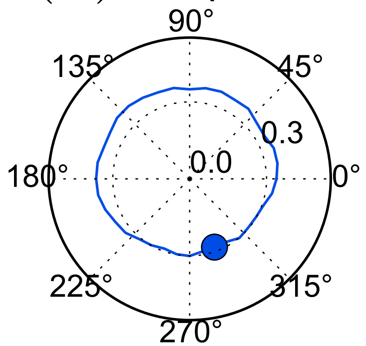
$\alpha_t = 0.465$

$\epsilon_{H_t}^{\text{train}} = 20.2\%$

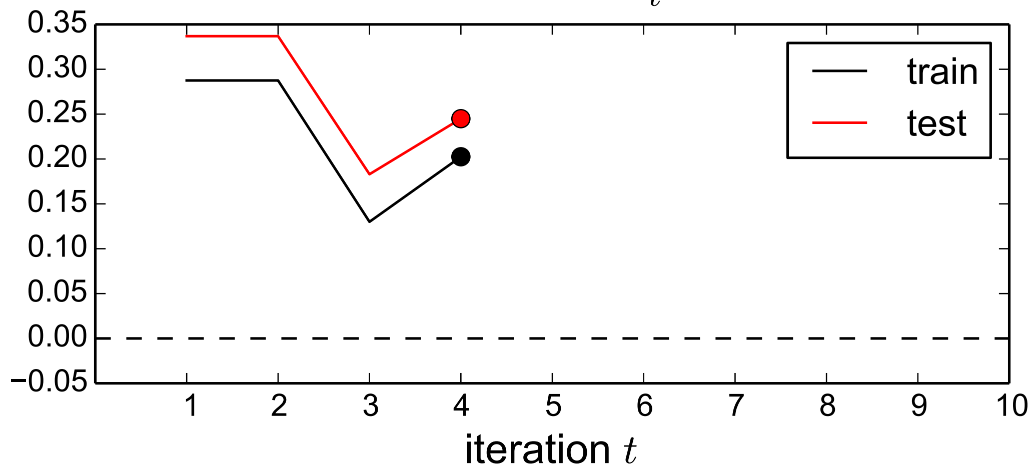
$\epsilon_{H_t}^{\text{test}} = 24.5\%$

$Z_t = 0.901$

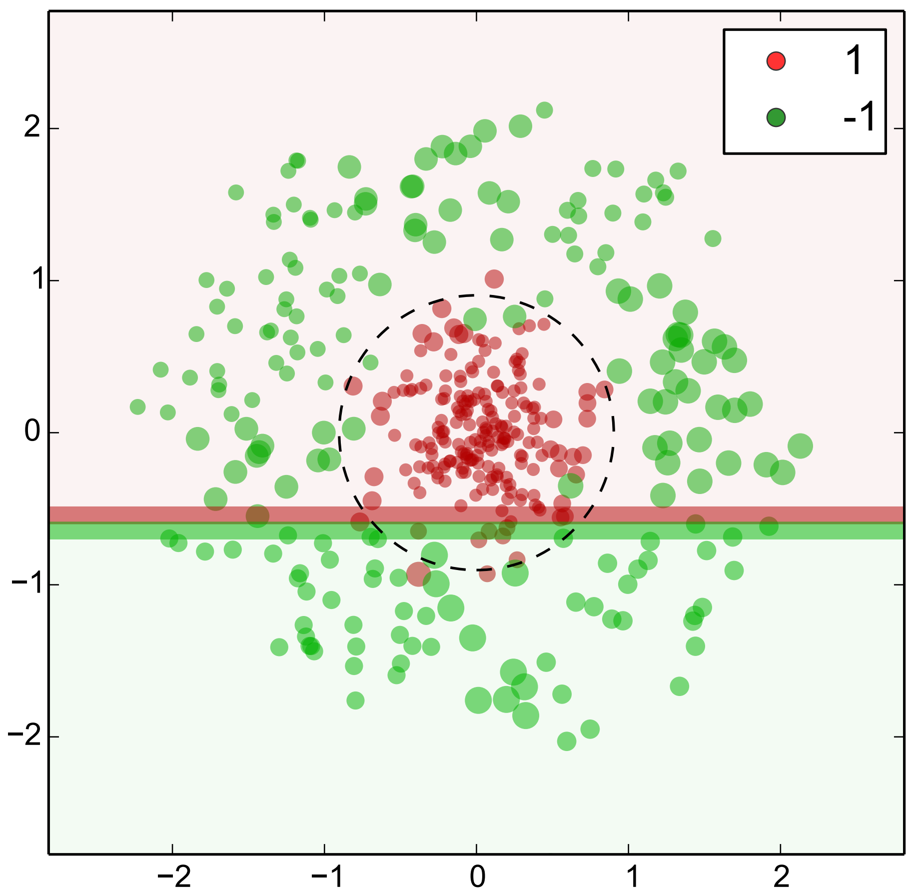
$\epsilon(\mathbf{w})$ at optimal b

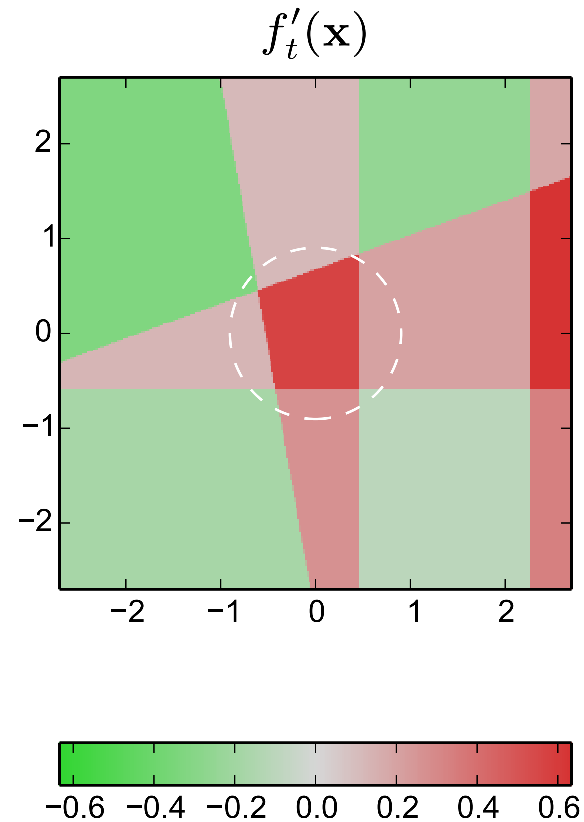


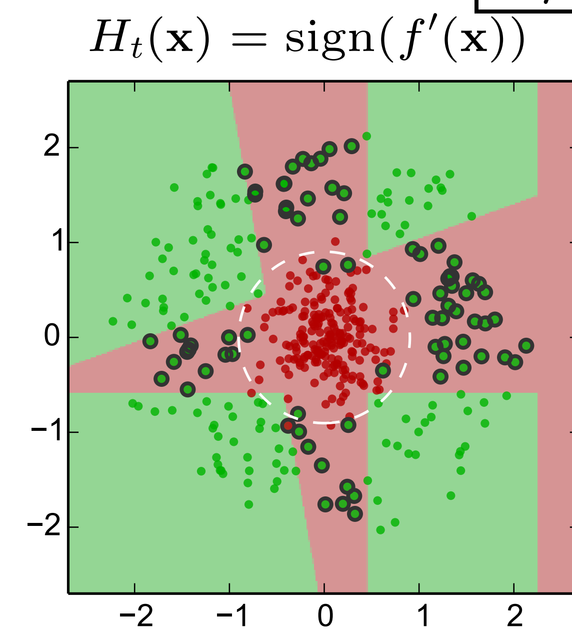
Error of H_t



Example 1 – iteration 5

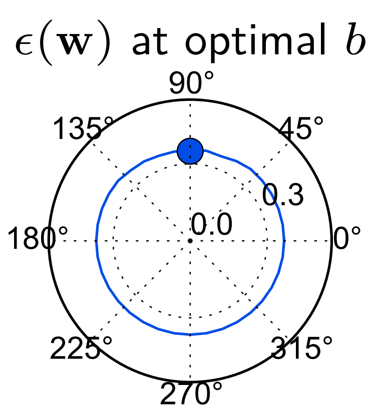


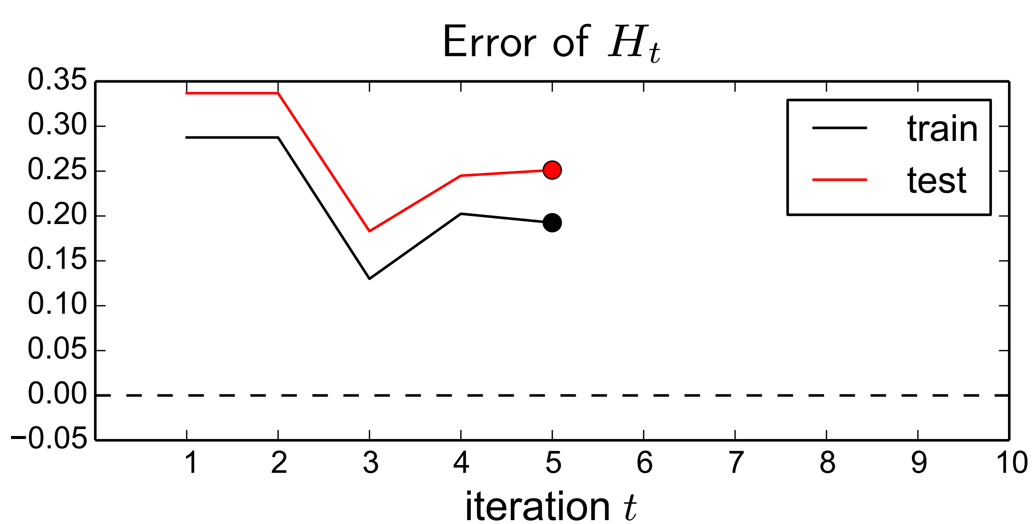




$$\epsilon_t = 34.9\%$$

$$\alpha_t = 0.312$$



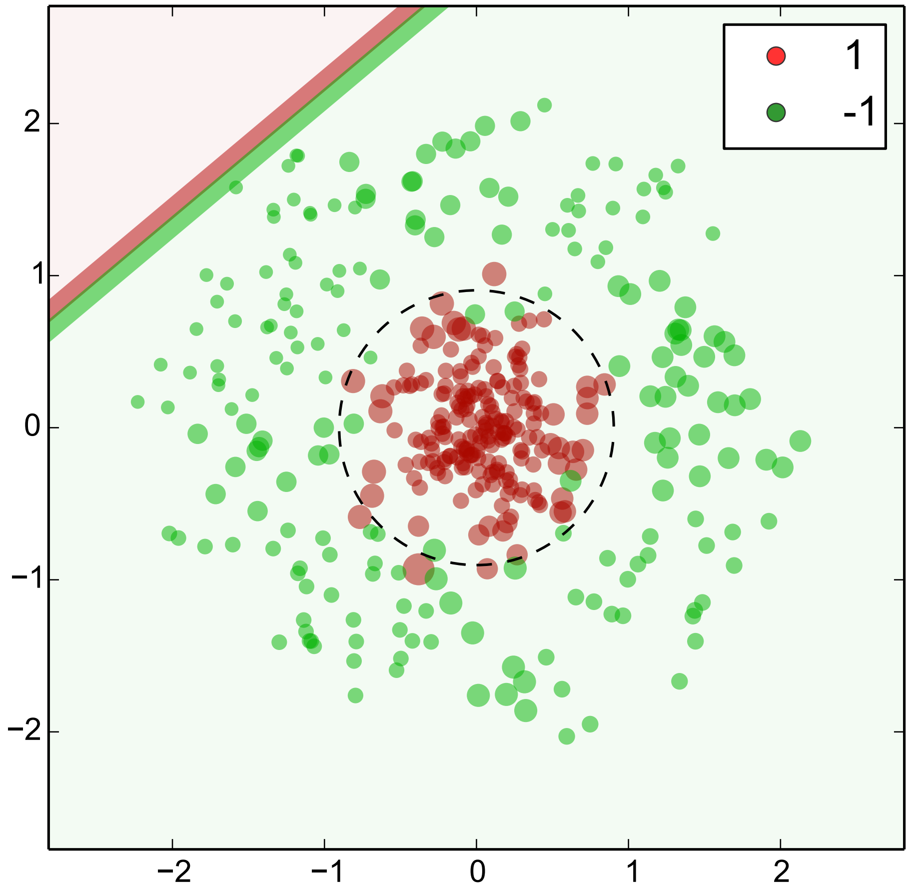


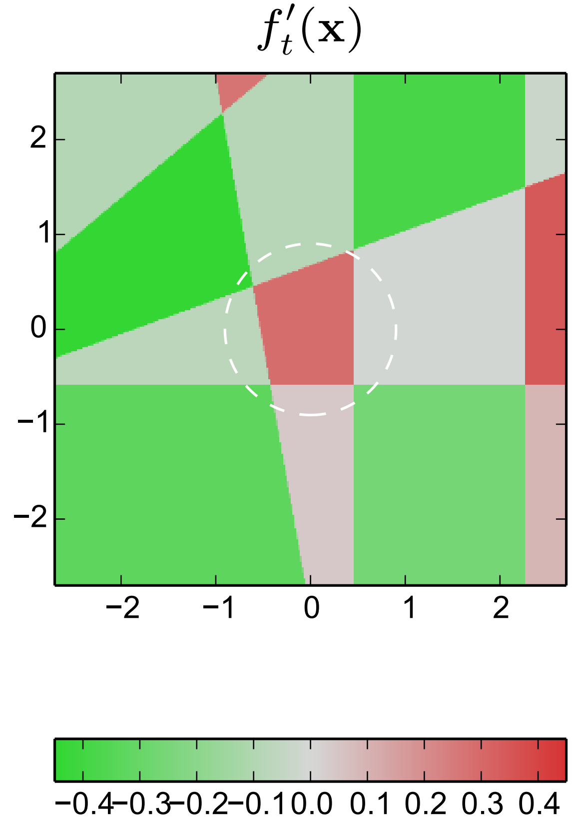
$$\epsilon_{H_t}^{\text{train}} = 19.2\%$$

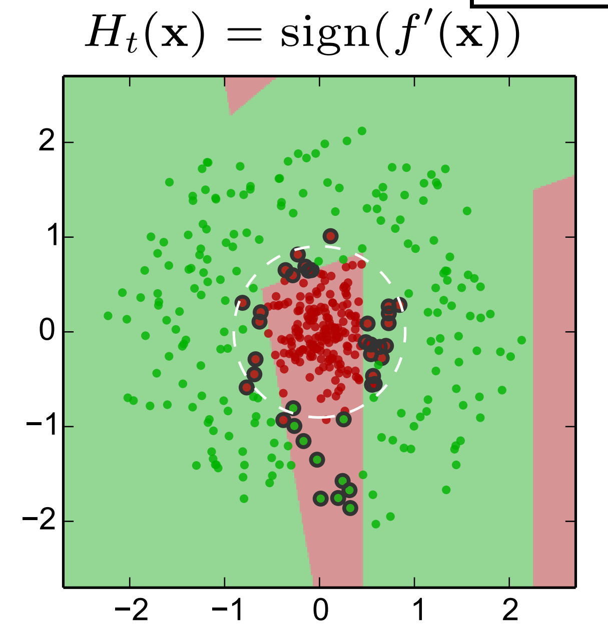
$$\epsilon_{H_t}^{\text{test}} = 25.1\%$$

$$Z_t = 0.953$$

Example 1 – iteration 6

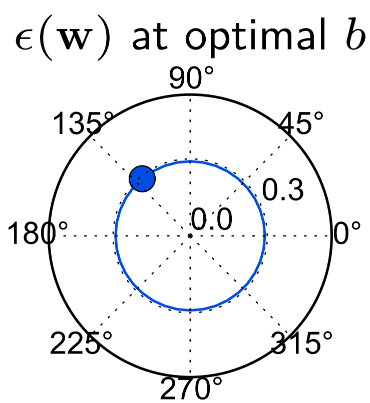


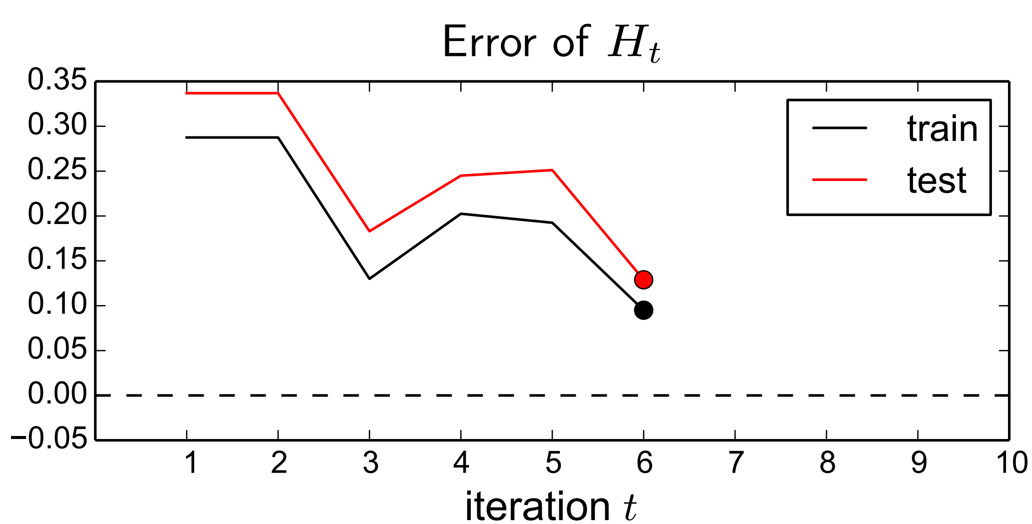




$$\epsilon_t = 29.0\%$$

$$\alpha_t = 0.447$$



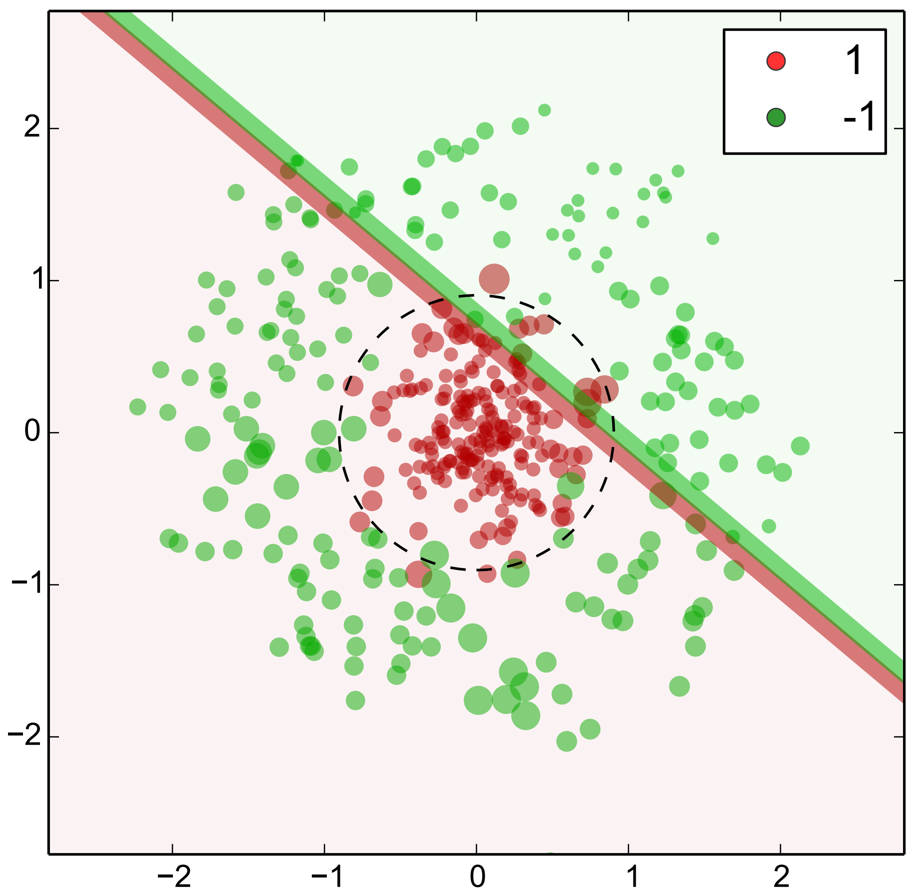


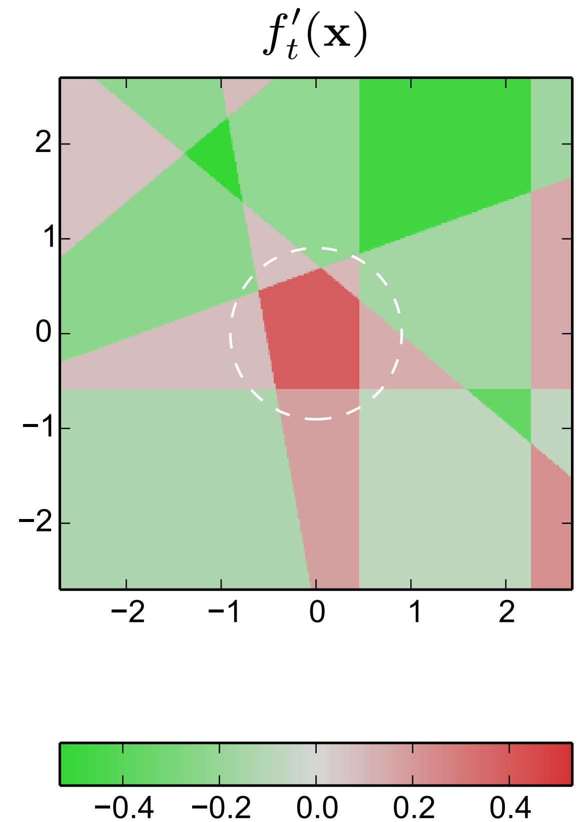
$$\epsilon_{H_t}^{\text{train}} = 9.50\%$$

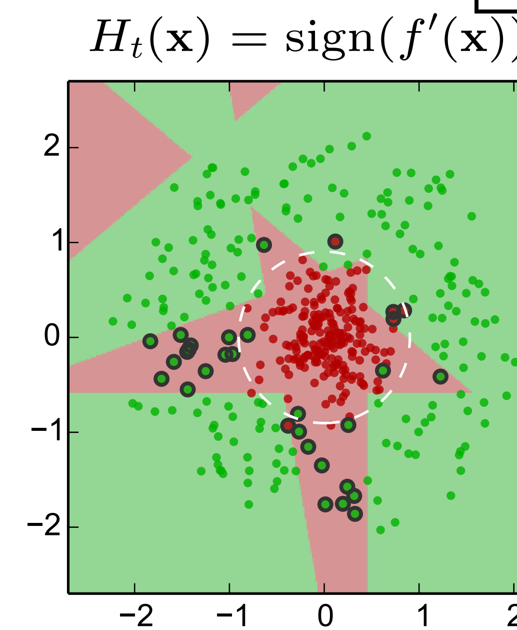
$$\epsilon_{H_t}^{\text{test}} = 12.9\%$$

$$Z_t = 0.908$$

Example 1 – iteration 7

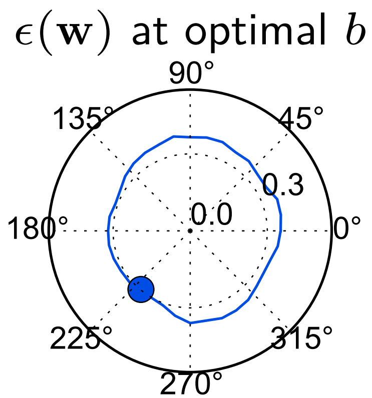


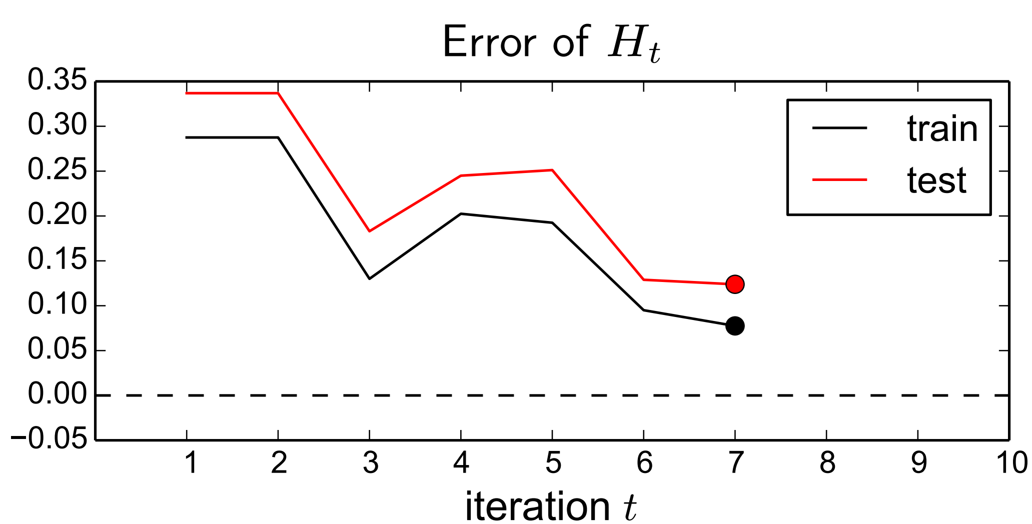




$$\epsilon_t = 29.8\%$$

$$\alpha_t = 0.429$$



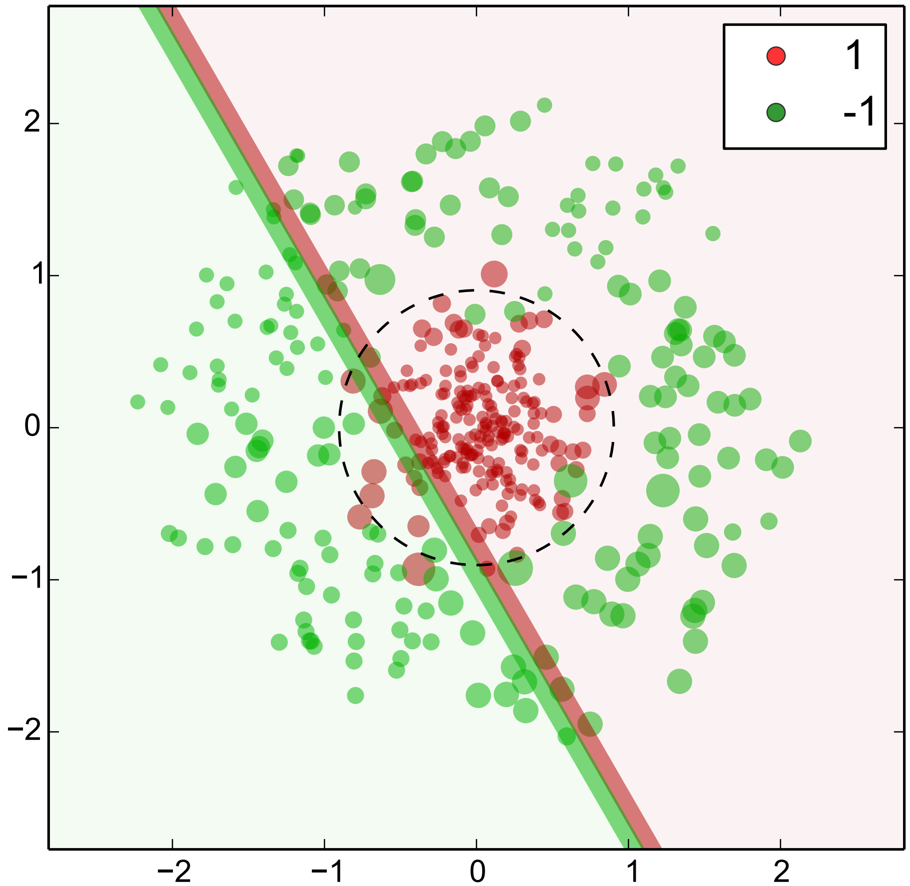


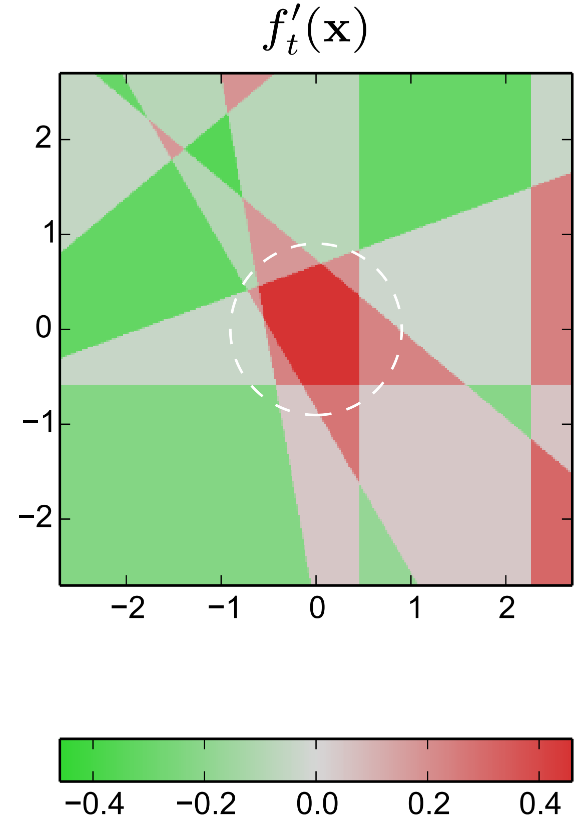
$$\epsilon_{H_t}^{\text{train}} = 7.75\%$$

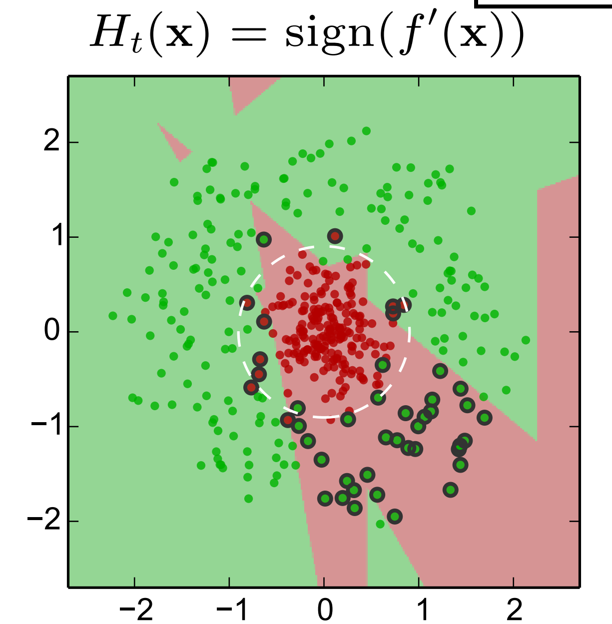
$$\epsilon_{H_t}^{\text{test}} = 12.4\%$$

$$Z_t = 0.915$$

Example 1 – iteration 8

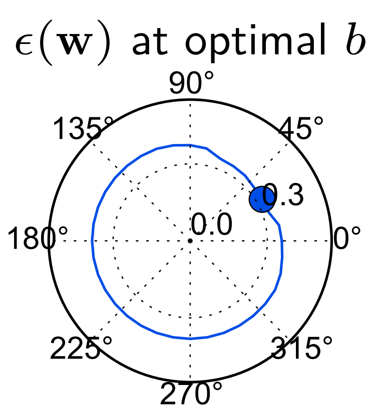


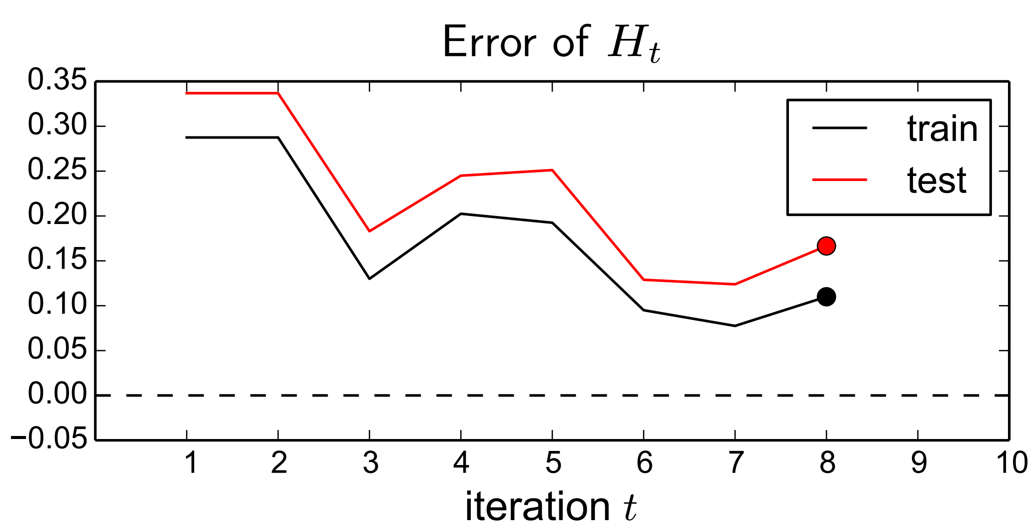




$$\epsilon_t = 32.3\%$$

$$\alpha_t = 0.369$$



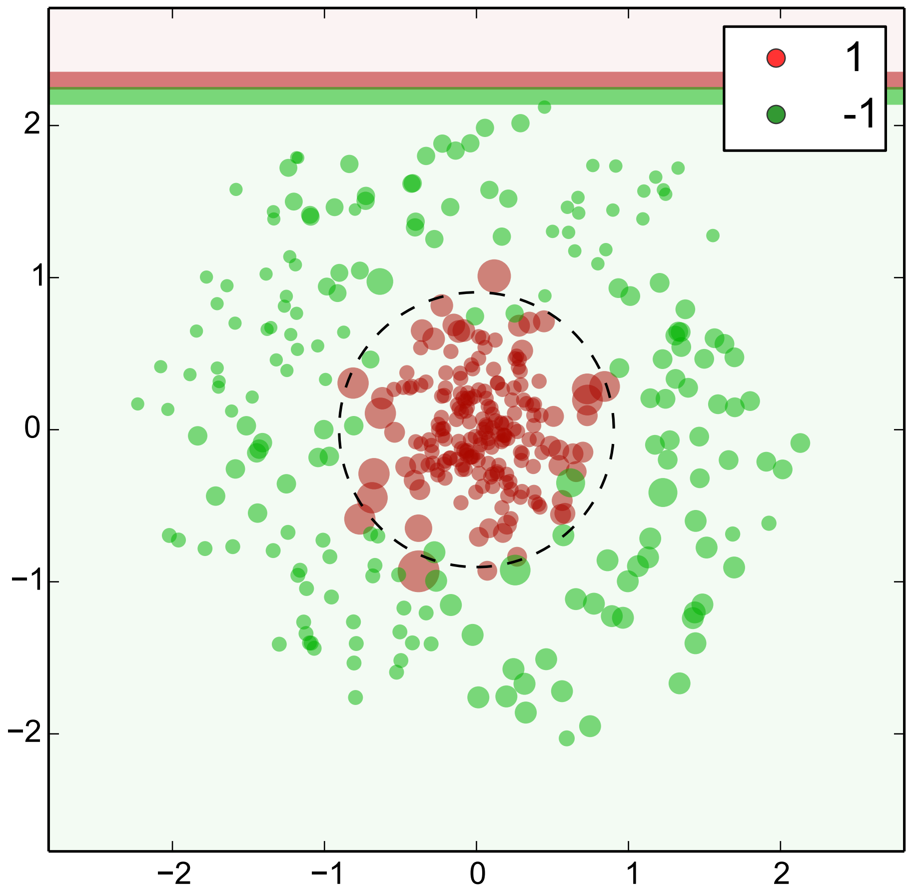


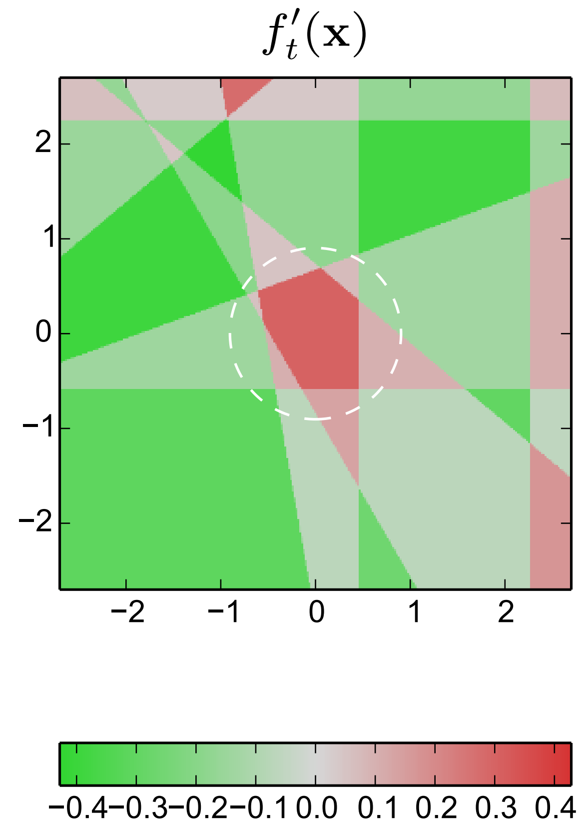
$$\epsilon_{H_t}^{\text{train}} = 11.0\%$$

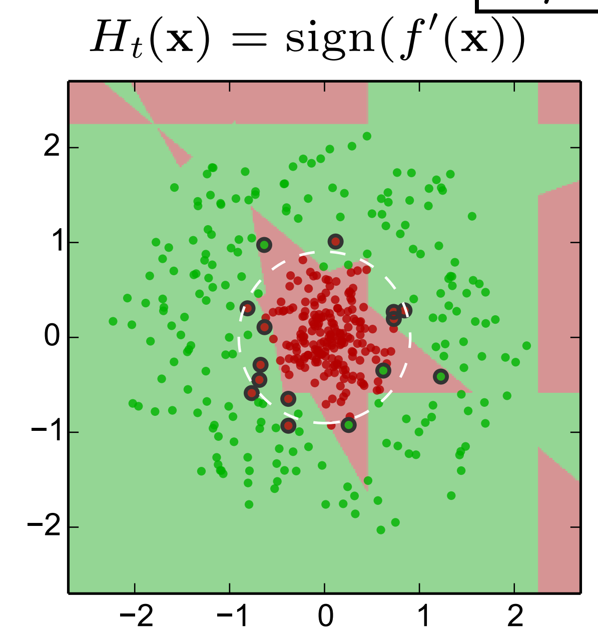
$$\epsilon_{H_t}^{\text{test}} = 16.7\%$$

$$Z_t = 0.935$$

Example 1 – iteration 9

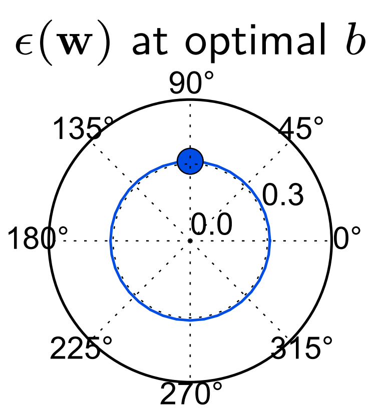


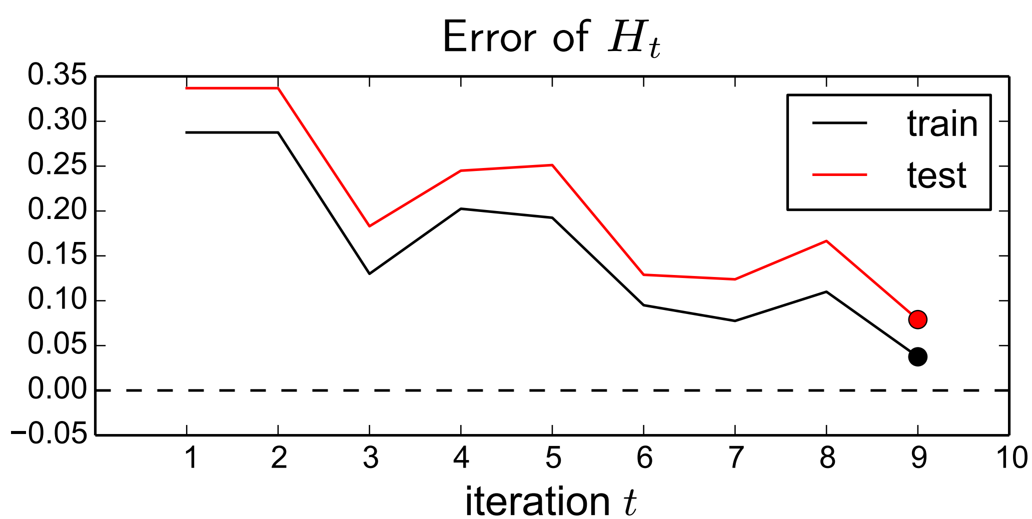




$\epsilon_t = 31.0\%$

$\alpha_t = 0.400$



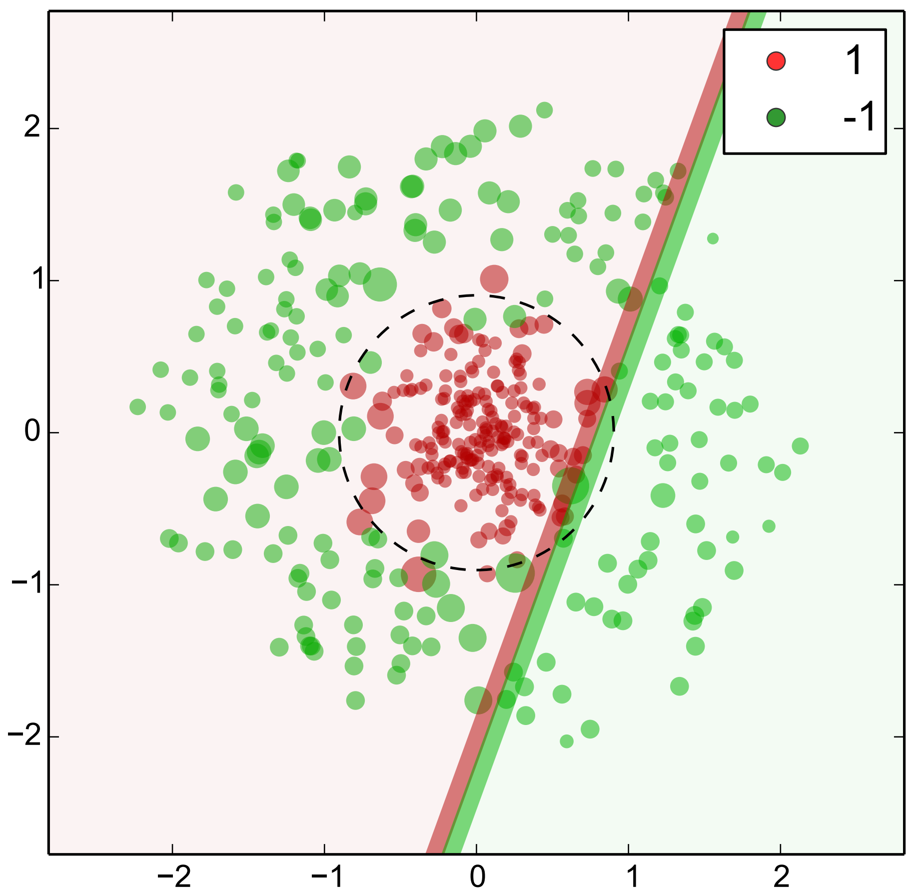


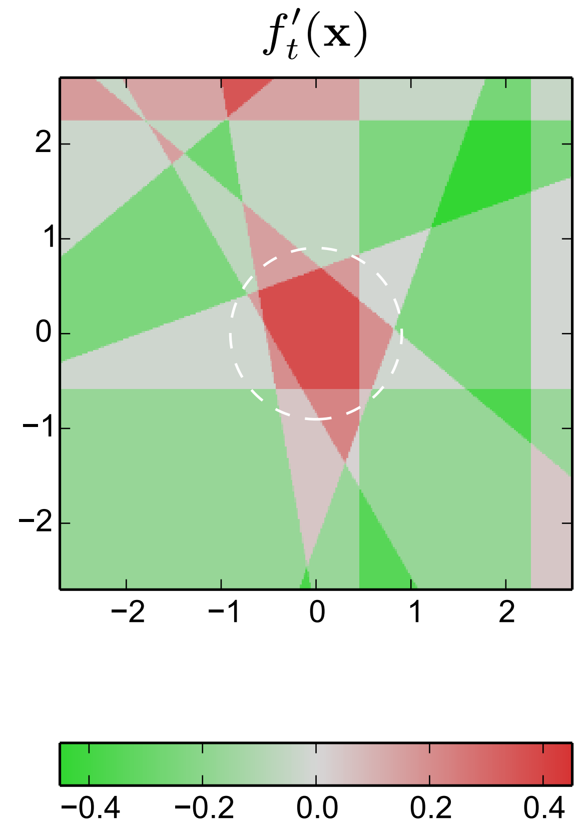
$\epsilon_{H_t}^{\text{train}} = 3.75\%$

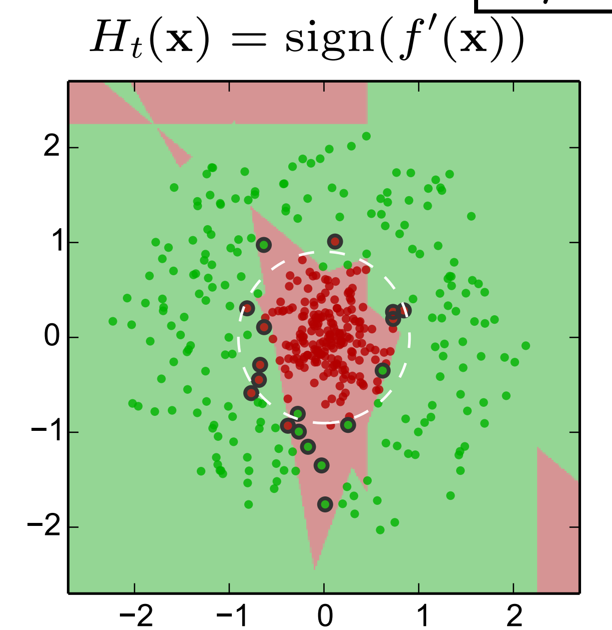
$\epsilon_{H_t}^{\text{test}} = 7.90\%$

$Z_t = 0.925$

Example 1 – iteration 10

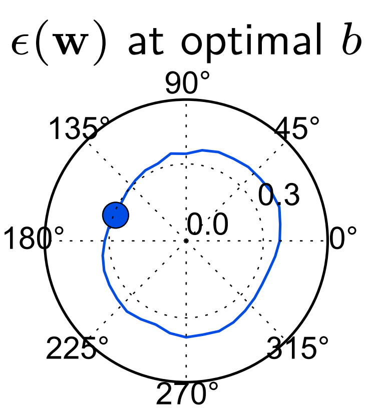


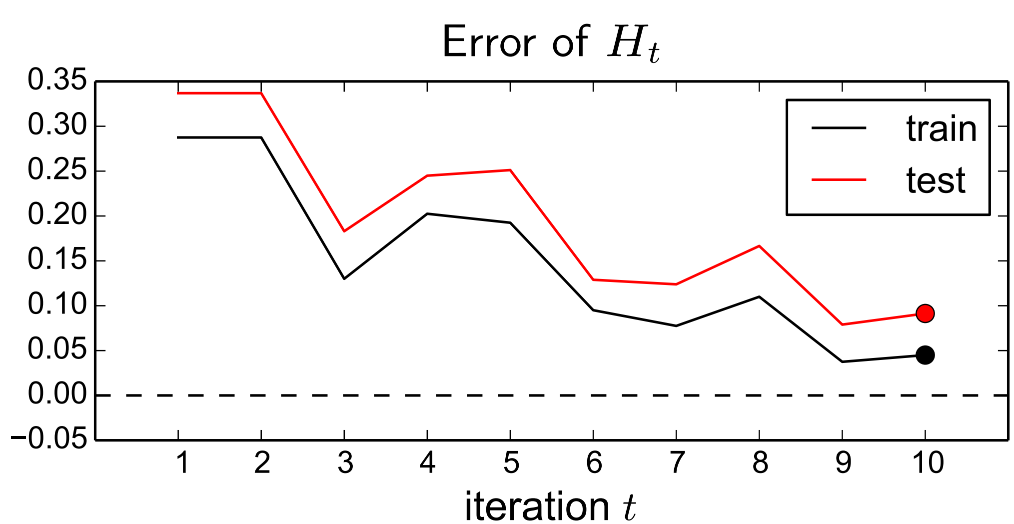




$$\epsilon_t = 29.2\%$$

$$\alpha_t = 0.443$$



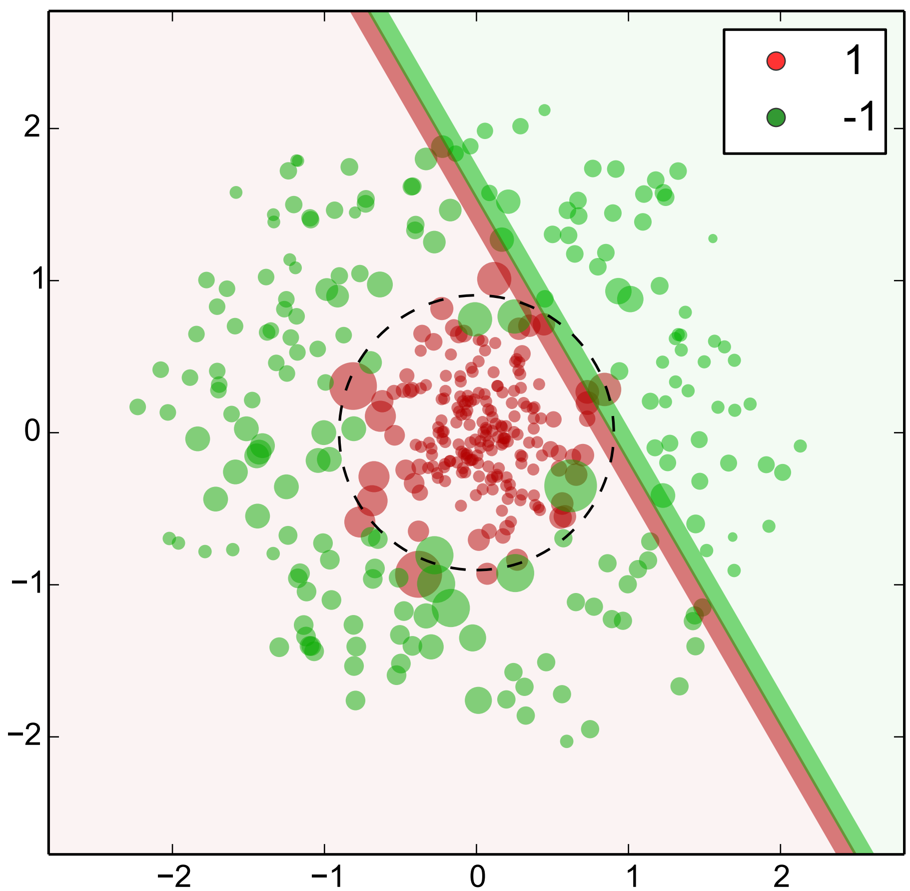


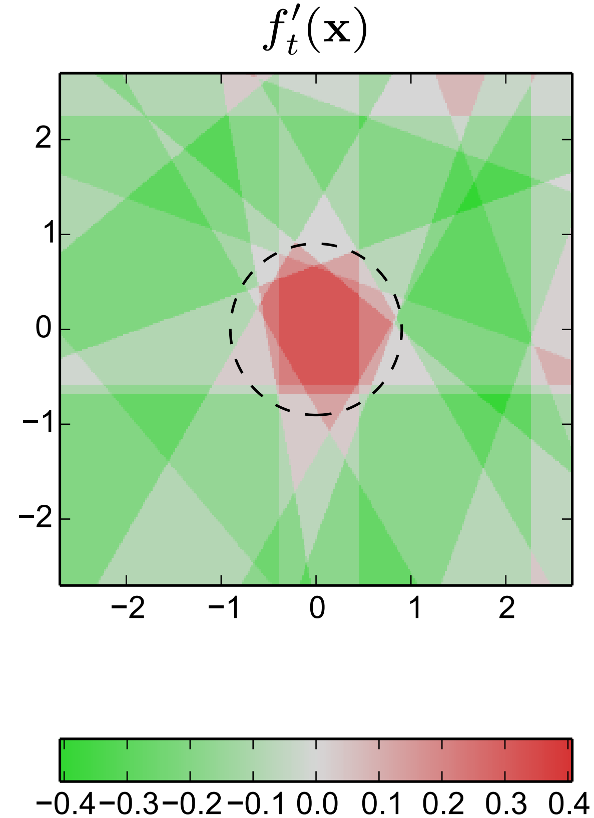
$$\epsilon_{H_t}^{\text{train}} = 4.50\%$$

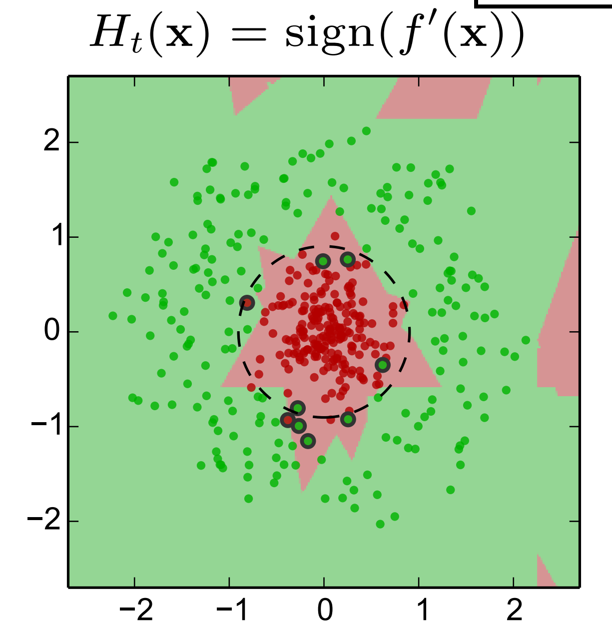
$$\epsilon_{H_t}^{\text{test}} = 9.13\%$$

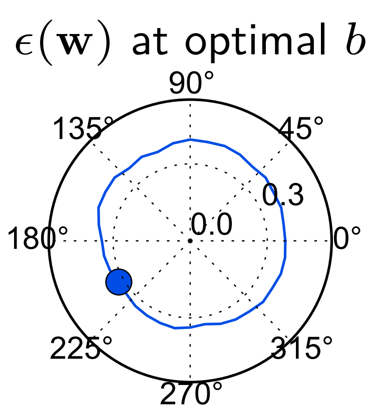
$$Z_t = 0.909$$

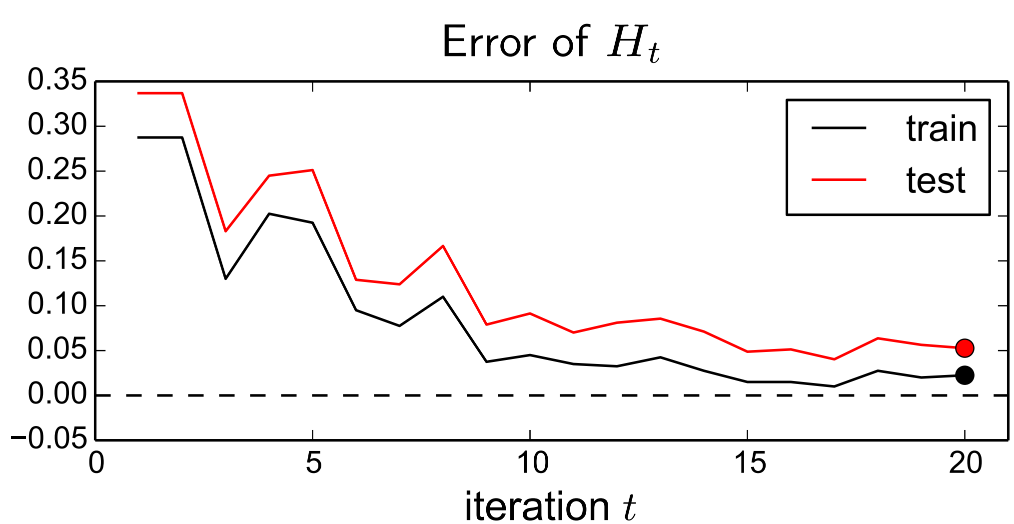
Example 1 – iteration 20











$$\epsilon_t = 32.0\%$$

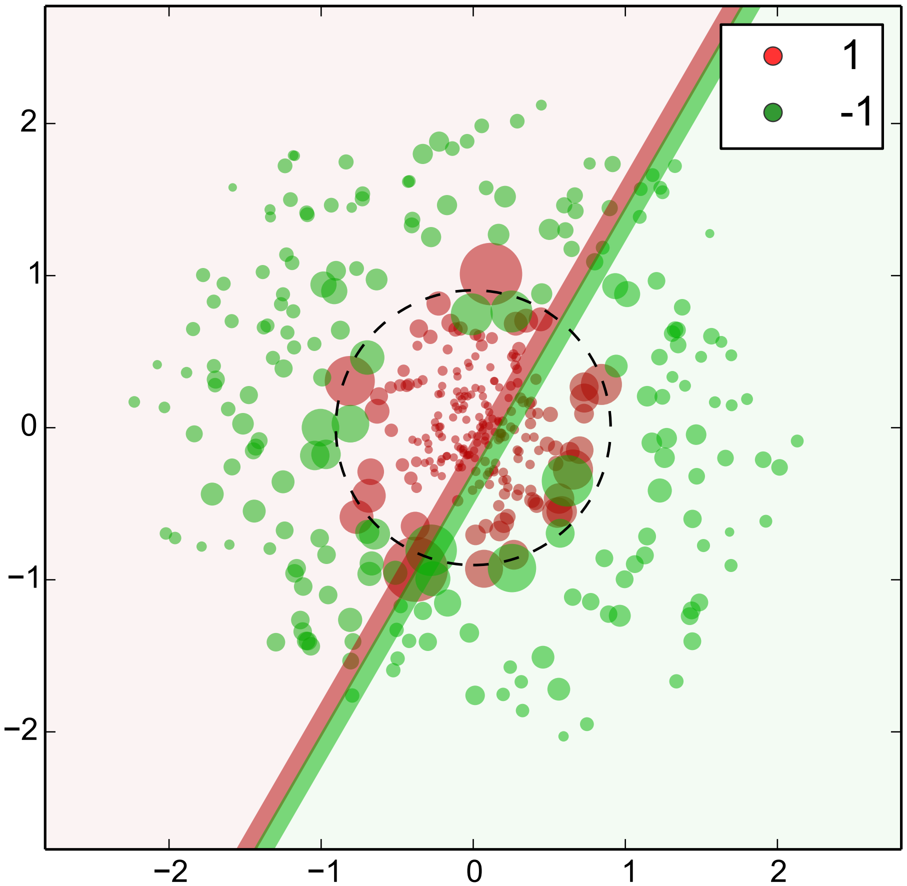
$$\alpha_t = 0.376$$

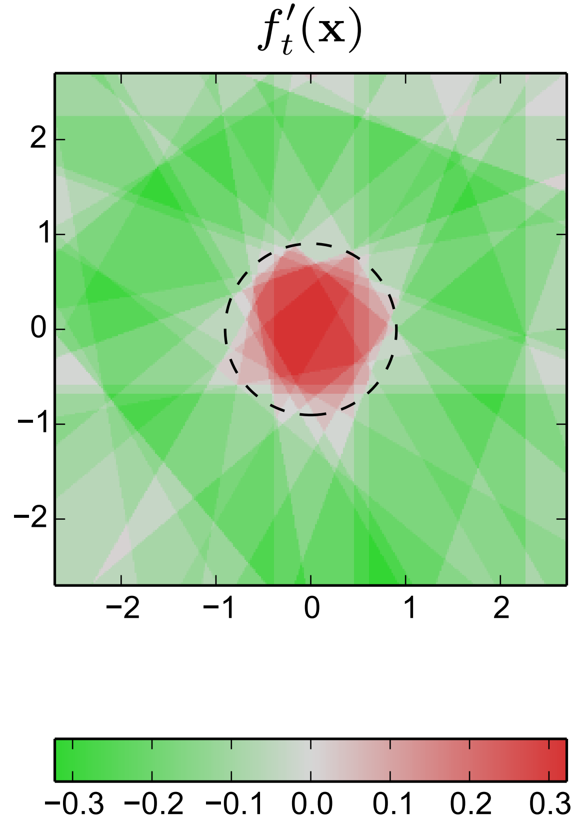
$$\epsilon_{H_t}^{\text{train}} = 2.25\%$$

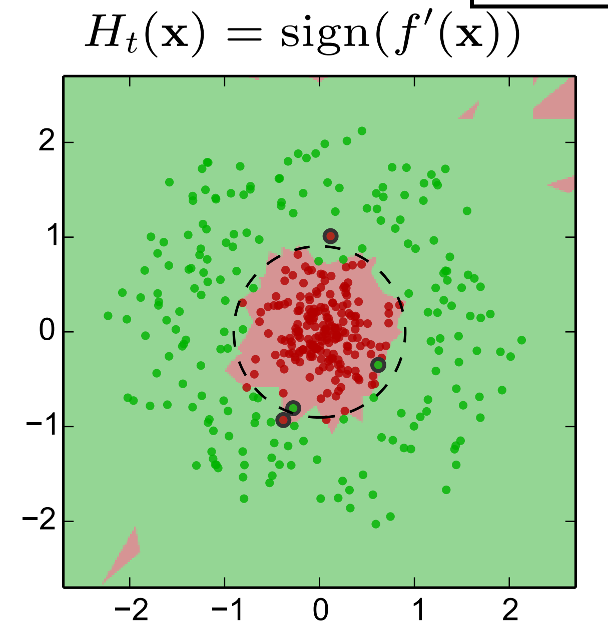
$$\epsilon_{H_t}^{\text{test}} = 5.27\%$$

$$Z_t = 0.933$$

Example 1 – iteration 40

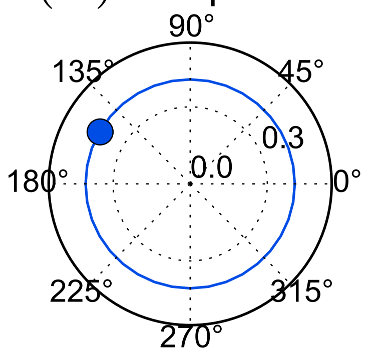


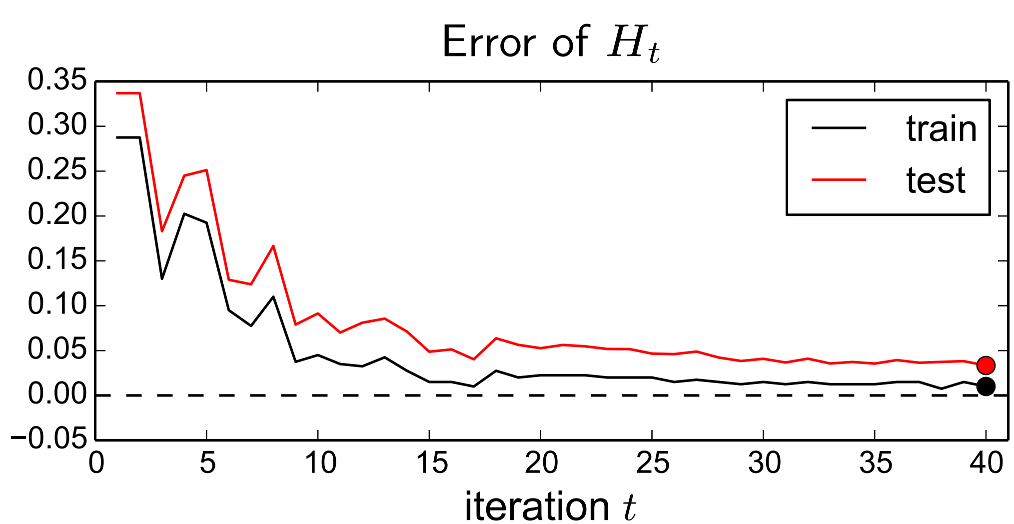




$$\epsilon_t = 40.4\%$$

$$\alpha_t = 0.194$$



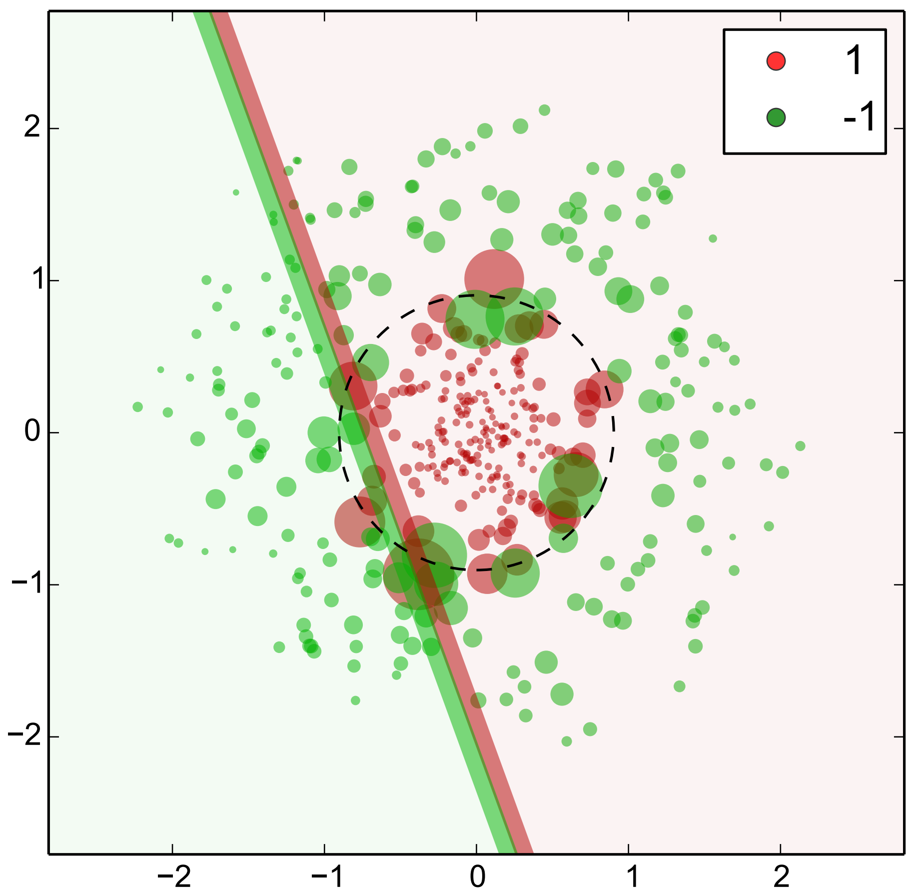


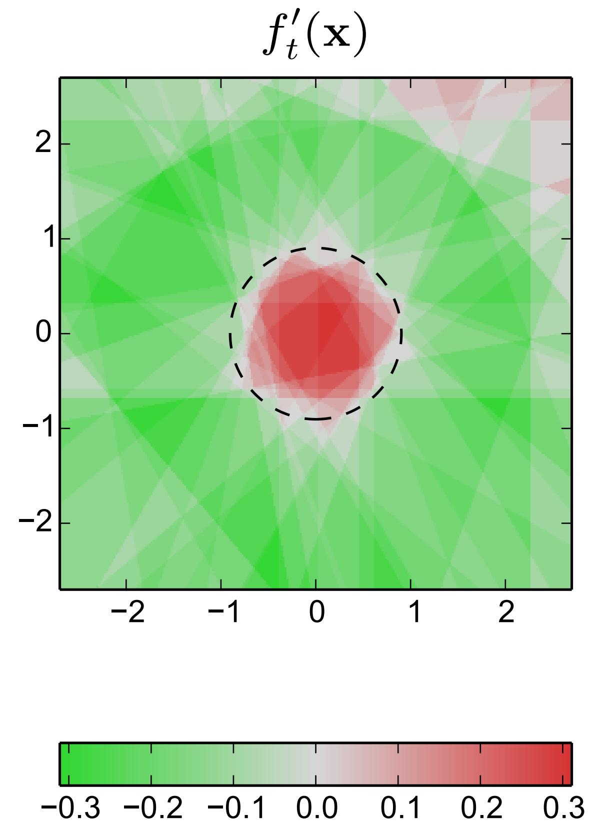
$$\epsilon_{H_t}^{\text{train}} = 1.00\%$$

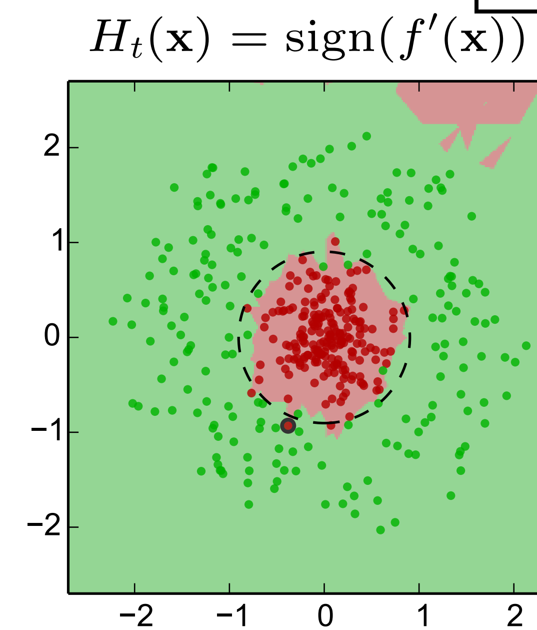
$$\epsilon_{H_t}^{\text{test}} = 3.34\%$$

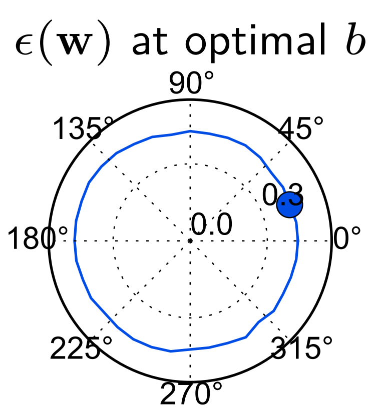
$$Z_t = 0.982$$

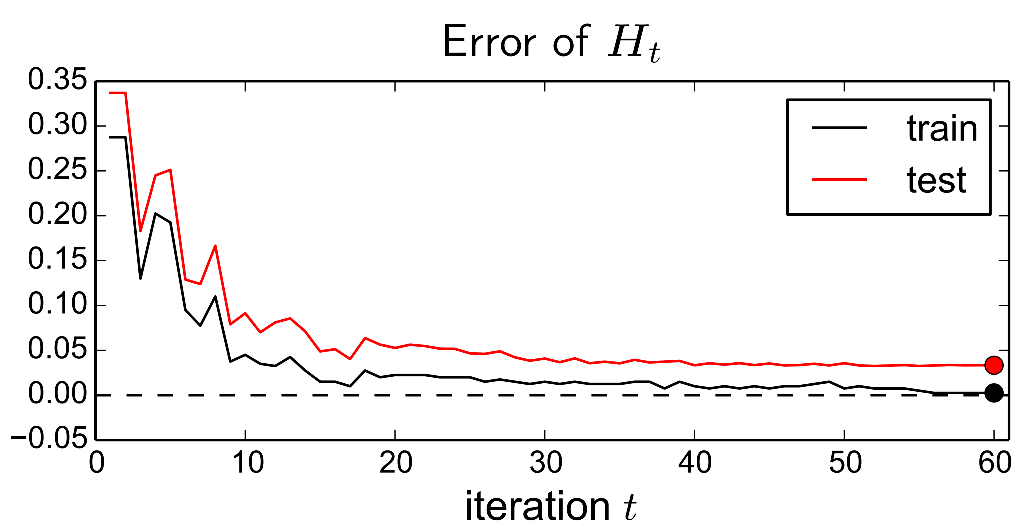
Example 1 – iteration 60











$$\epsilon_t = 41.1\%$$

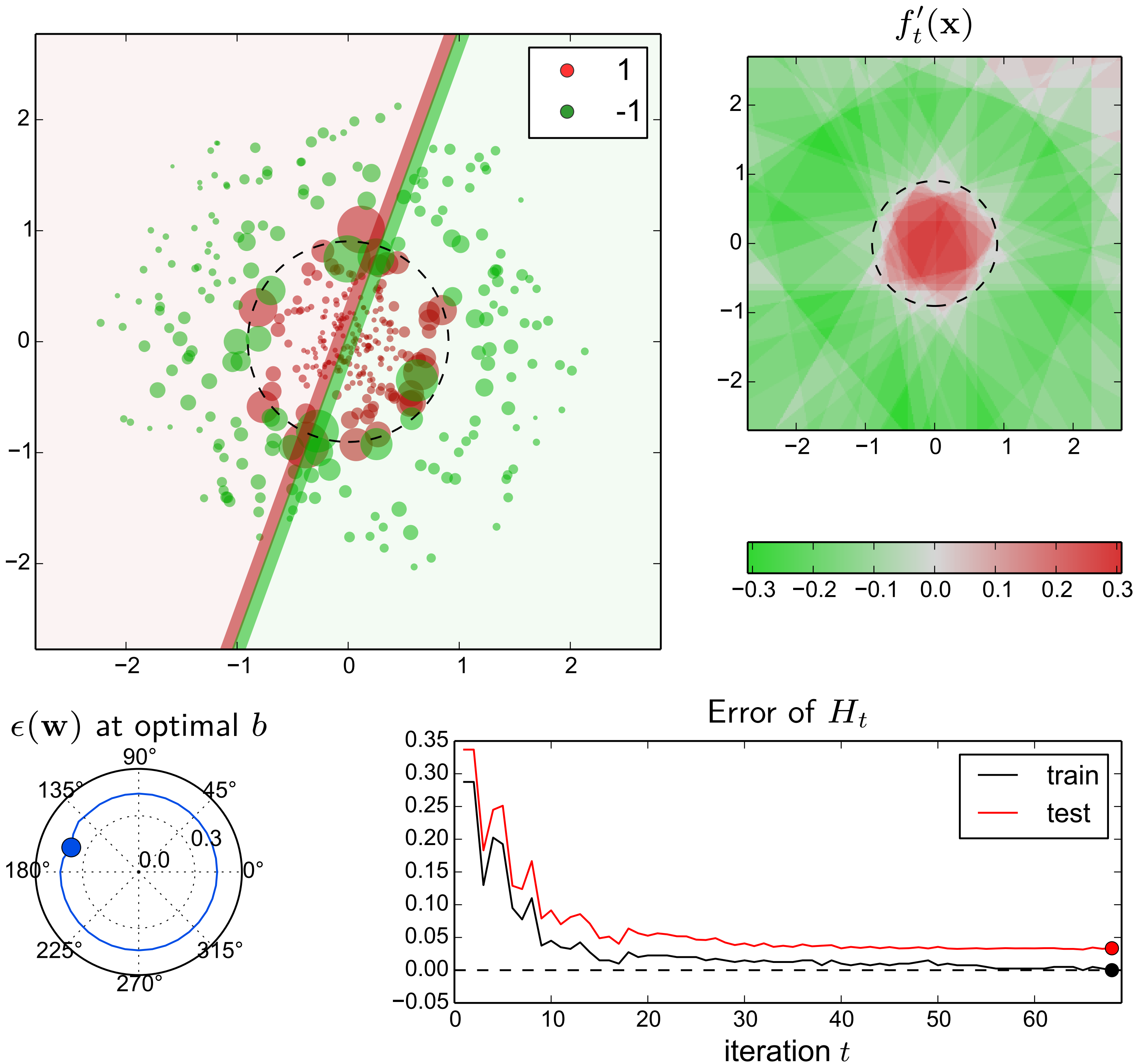
$$\alpha_t = 0.179$$

$$\epsilon_{H_t}^{\text{train}} = 0.250\%$$

$$\epsilon_{H_t}^{\text{test}} = 3.33\%$$

$$Z_t = 0.984$$

Example 1 – iteration 68



$\epsilon_t = 38.3\%$

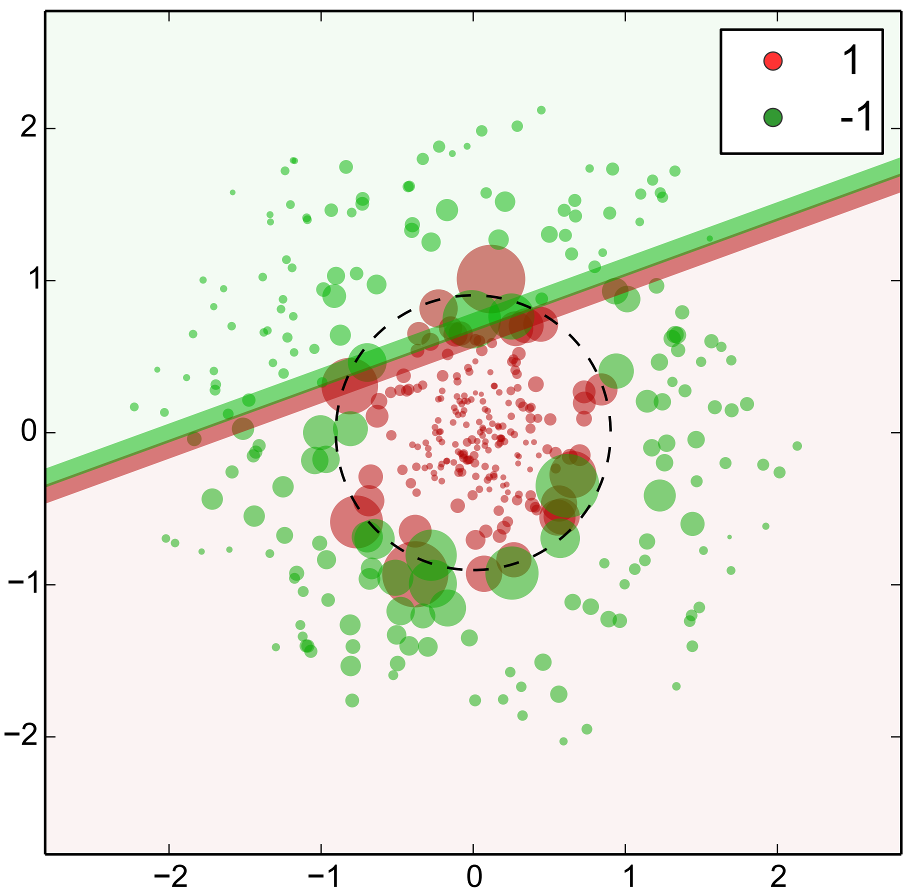
$\alpha_t = 0.239$

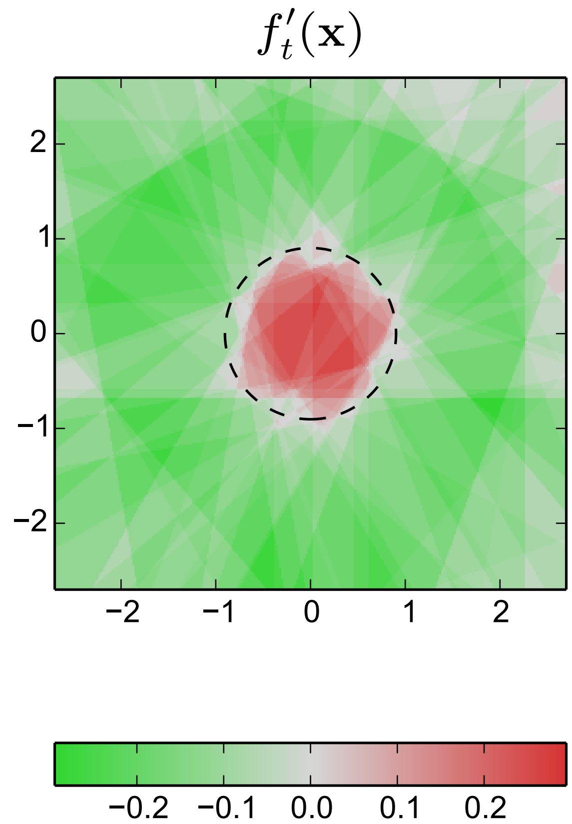
$\epsilon_{H_t}^{\text{train}} = 0.00\%$

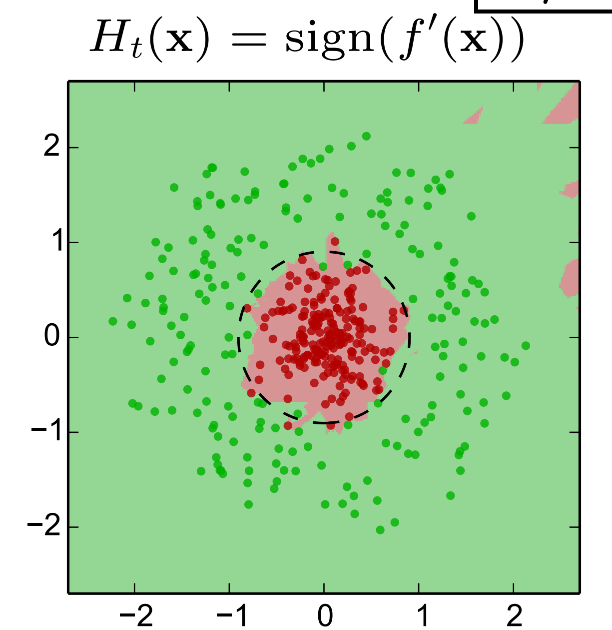
$\epsilon_{H_t}^{\text{test}} = 3.35\%$

$Z_t = 0.972$

Example 1 – iteration 100

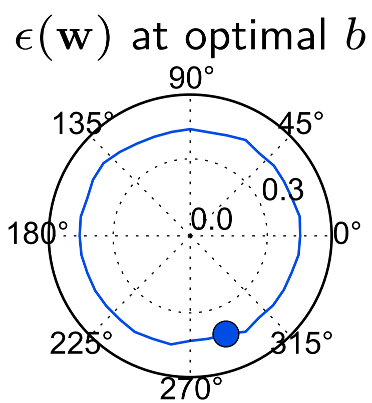


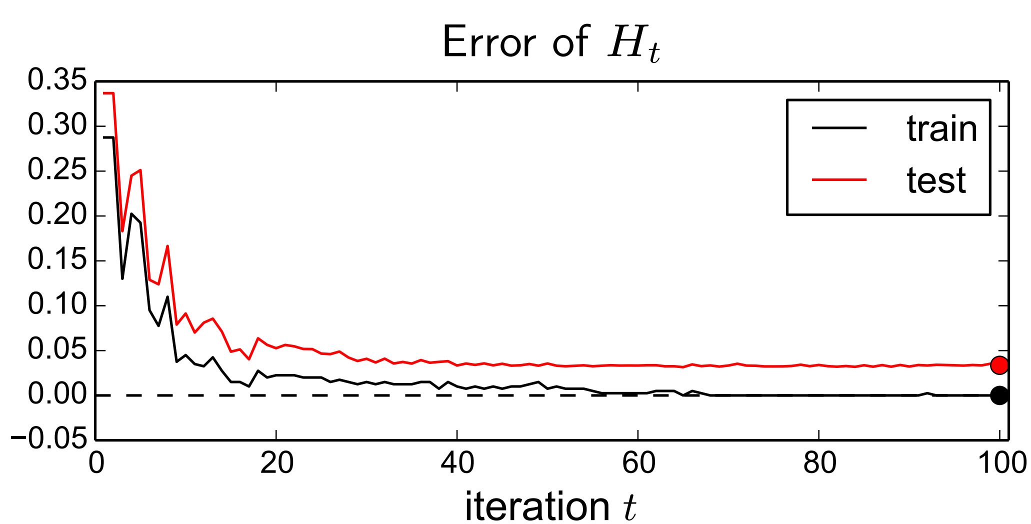




$$\epsilon_t = 40.7\%$$

$$\alpha_t = 0.189$$



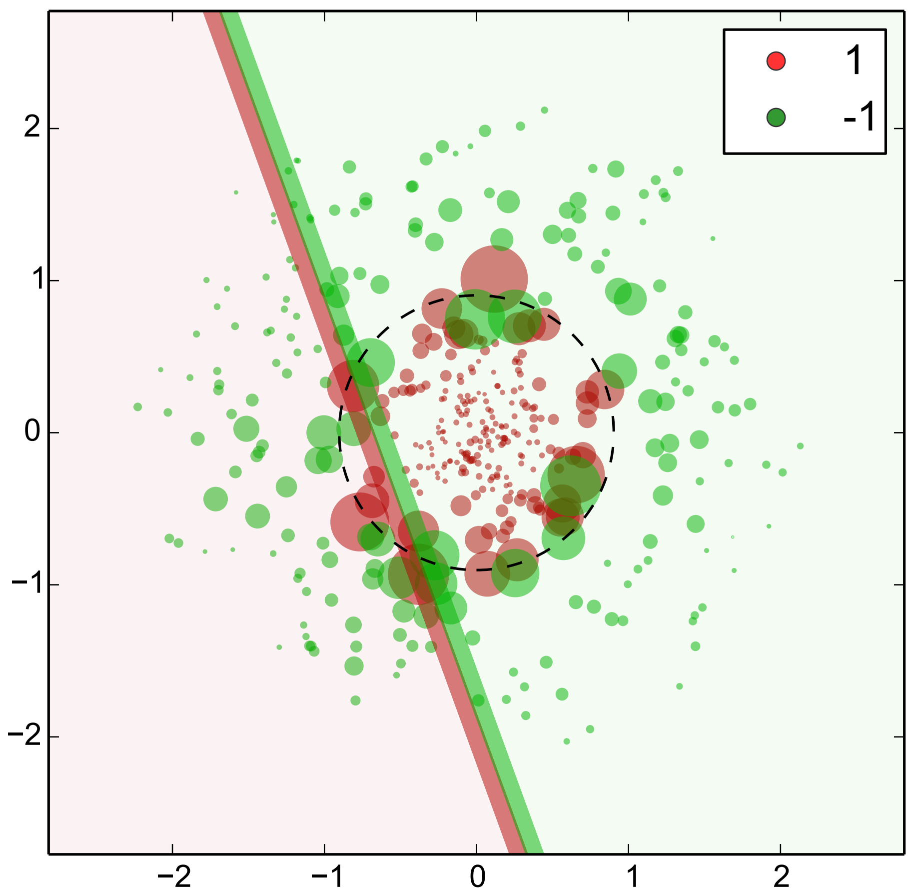


$$\epsilon_{H_t}^{\text{train}} = 0.00\%$$

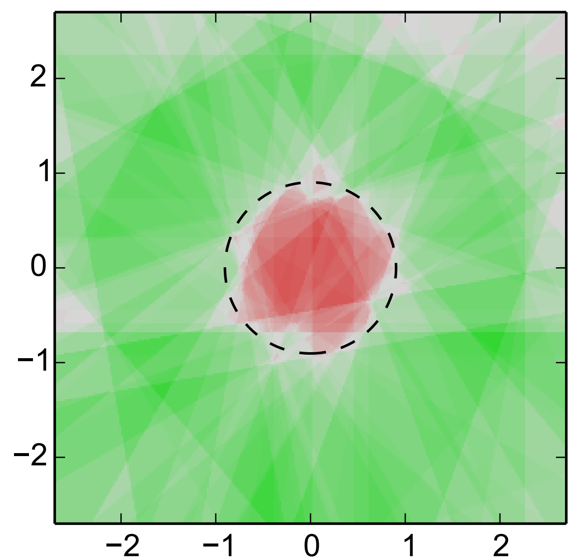
$$\epsilon_{H_t}^{\text{test}} = 3.36\%$$

$$Z_t = 0.982$$

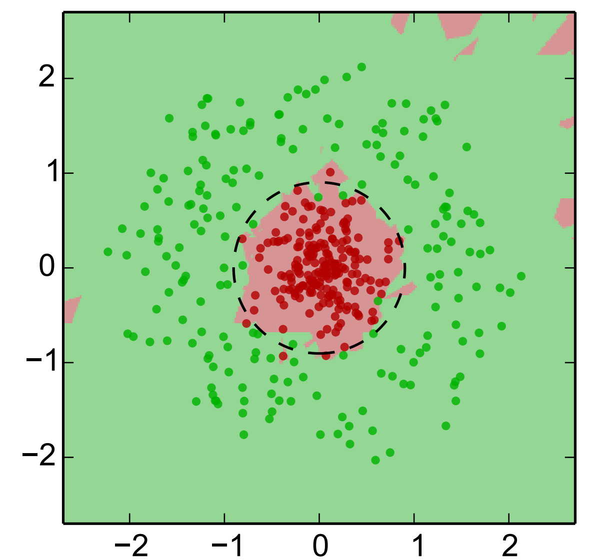
Example 1 – iteration 150

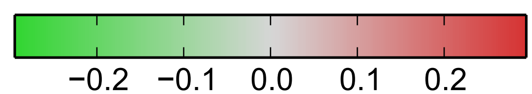


$f'_t(\mathbf{x})$



$H_t(\mathbf{x}) = \mathrm{sign}(f'(\mathbf{x}))$

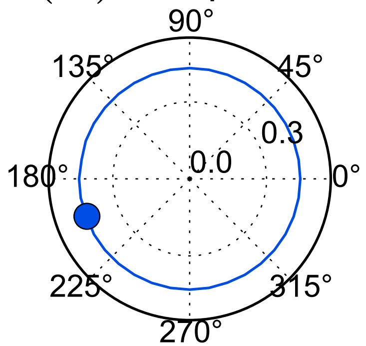




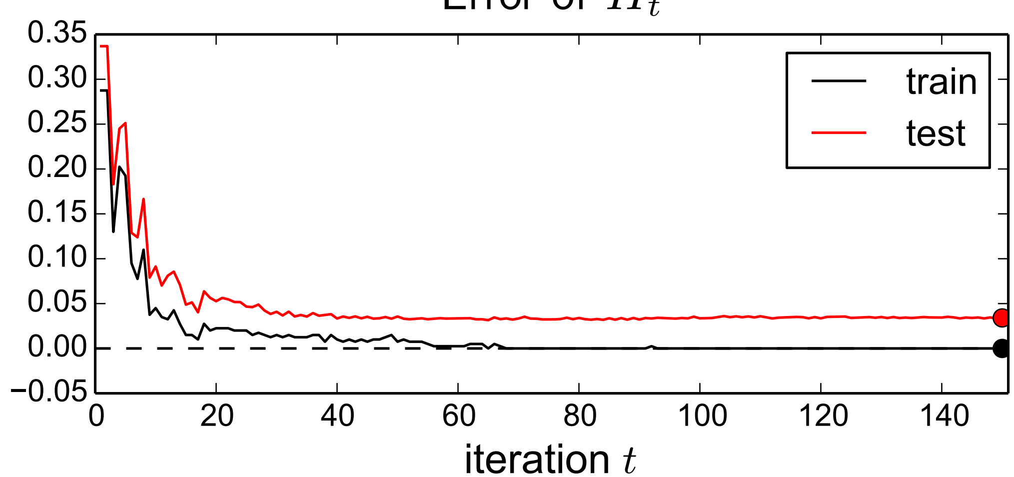
$\epsilon_t = 42.6\%$

$\alpha_t = 0.149$

$\epsilon(\mathbf{w})$ at optimal b



Error of H_t



$\epsilon_{H_t}^{\text{train}} = 0.00\%$

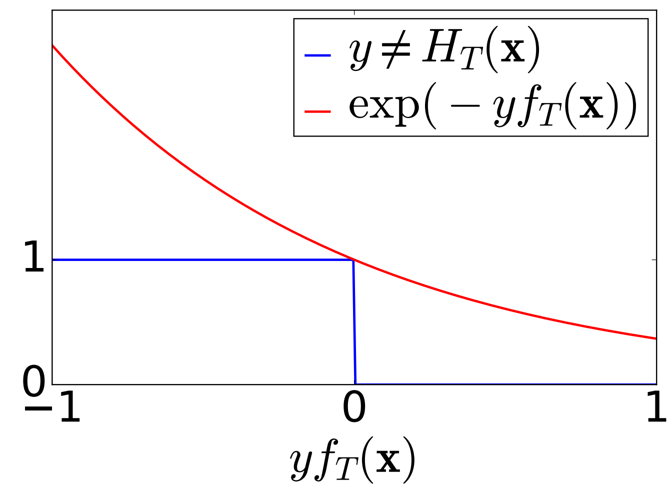
$\epsilon_{H_t}^{\text{test}} = 3.40\%$

$Z_t = 0.989$

Upper bound theorem (1/2)

Theorem: The following upper bound holds, in iteration T, for the training error ϵ of H_T:

$$\epsilon = \frac{1}{L}\sum_{i=1}^{L}[\![y_i \neq H_T(\mathbf{x}_i)]\!] \leq \prod_{t=1}^{T} Z_t \, .$$



Proof: There holds that:

$$[\![H_T(\mathbf{x}_i) \neq y_i]\!] \leq \exp(-y_i f_T(\mathbf{x}_i)) \, , \qquad (4)$$

which can be checked by a simple observation (the inequality follows from the first and last columns):

$[\![H_T(\mathbf{x}) \neq y]\!]$	classification	$yH_T(\mathbf{x})$	$yf_T(\mathbf{x})$	$\exp(-yf_T(\mathbf{x}))$
0	correct	1	> 0	≥ 0
1	incorrect	-1	< 0	≥ 1

Summing over the training dataset and dividing by L, we get

$$\epsilon = \frac{1}{L}\sum_{i}[\![H_T(\mathbf{x}_i) \neq y_i]\!] \leq \frac{1}{L}\sum_{i}\exp(-y_i f_T(\mathbf{x}_i))$$

Upper bound theorem (2/2)

Theorem: The following upper bound holds, in iteration T, for the training error ϵ of H_T:

$$\epsilon = \frac{1}{L}\sum_{i=1}^{L}[\![y_i \neq H_T(\mathbf{x}_i)]\!] \leq \prod_{t=1}^{T} Z_t\,.$$

Proof (contd.):

$$\epsilon = \frac{1}{L}\sum_i [\![H_T(\mathbf{x}_i) \neq y_i]\!] \leq \frac{1}{L}\sum_i \exp(-y_i f_T(\mathbf{x}_i))$$

But from the distribution update rule:

$$D_{T+1}(i) = \frac{\exp(-y_i f_T(\mathbf{x}_i))}{L\prod_{t=1}^{T} Z_t}$$

we have that

$$\frac{1}{L}\sum_i \exp(-y_i f_T(\mathbf{x}_i)) = \left(\prod_{t=1}^{T} Z_t\right)\underbrace{\left(\sum_i D_{T+1}(i)\right)}_{=1},$$

which completes the proof.

AdaBoost as a Minimiser of the Upper Bound on the Empirical Error

- The main objective is to minimize $\epsilon = \frac{1}{L}\sum_{i=1}^{L}[\![y_i \neq H_T(\mathbf{x}_i)]\!]$ (plus maximize the margin).
- ϵ has just been shown to be upperbounded: $\epsilon(H_T) \leq \Pi_{t=1}^{T} Z_t$.
- Adaboost is minimizing this upper bound.
- It does so by greedily minimizing Z_t in each iteration.
- Recall that

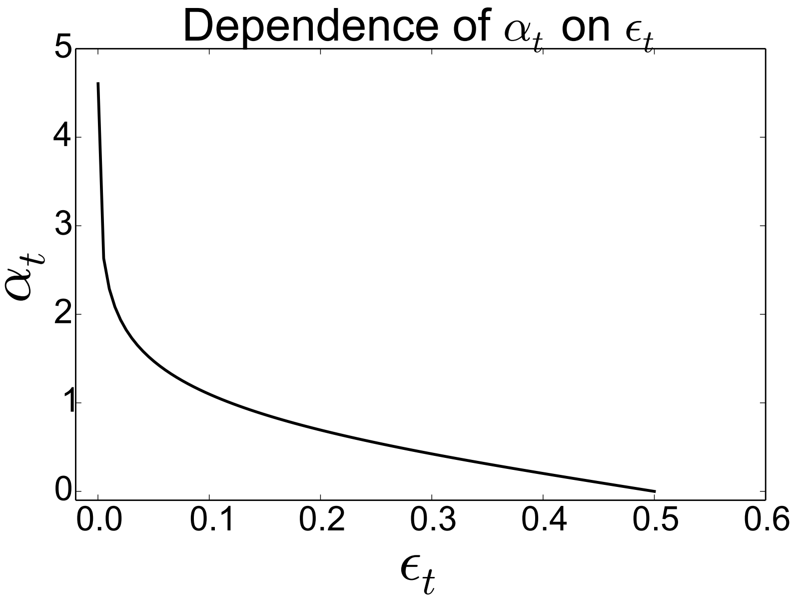
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \;;$$

given the dataset $\{(\mathbf{x}_i, y_i)\}$ and the distribution D_t in iteration t, the variables to minimize Z_t over are α_t and h_t.

Choosing α_t

Let us minimize $Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$ with respect to α_t:

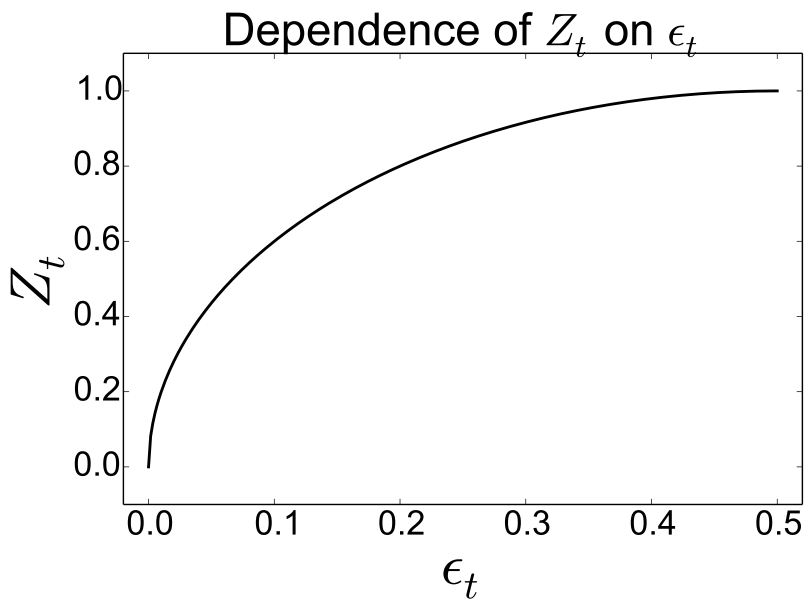
$$\begin{aligned}
\frac{\mathrm{d}Z}{\mathrm{d}\alpha_t} = -\sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} y_i h_t(\mathbf{x}_i) &= 0 \\
-\underbrace{\sum_{i:y_i=h_t(\mathbf{x}_i)} D_t(i)}_{1-\epsilon_t} e^{-\alpha_t} + \underbrace{\sum_{i:y_i\neq h_t(\mathbf{x}_i)} D(i)}_{\epsilon_t} e^{\alpha_t} &= 0 \\
-e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t &= 0 \\
\alpha_t + \log \epsilon_t &= -\alpha_t + \log(1-\epsilon_t) \\
\alpha_t &= \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}
\end{aligned}$$



Choosing h_t

Let us substitute $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$ into Z_t:

$$
\begin{aligned}
Z_t &= \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \\
&= \sum_{i:y_i=h_t(\mathbf{x}_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i\neq h_t(\mathbf{x}_i)} D_t(i) e^{\alpha_t} \\
&= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}
$$



⇒ Z_t is minimised by selecting h_t with minimal weighted error ϵ_t.

Weak classifier examples

- Decision tree, Perceptron – $\mathcal{B}$ *infinite*
- Selecting the best one from a given *finite* set $\mathcal{B}$

Minimization of an Upper Bound on the Empirical Error - Recapitulation

Choosing α_t and h_t

- For any weak classifier h_t with error ϵ_t, $Z_t(\alpha)$ is a convex differentiable function with a single minimum at α_t:

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

- $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)} \leq 1$ for optimal α_t $\Rightarrow$ Z_t is minimized by h_t with minimal ϵ_t.

Comments

- The process of selecting α_t and $h_t(\mathbf{x})$ can be interpreted as a single optimisation step minimising the upper bound on the empirical error. Improvement of the bound is guaranteed, provided that $\epsilon < 1/2$.
- The process can be interpreted as a component-wise local optimisation (Gauss-Southwell iteration) in the (possibly infinite dimensional!) space of $\vec{\alpha} = (\alpha_1, \alpha_2, \dots)$ starting from $\vec{\alpha}_0 = (0, 0, \dots)$.

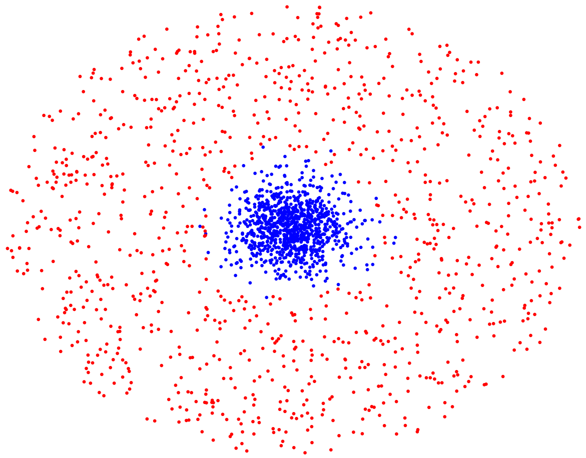
Summary of the Algorithm

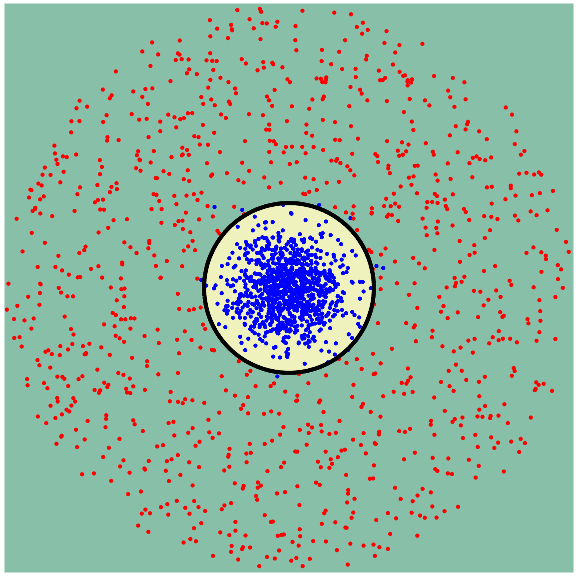
Initialization ...

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:



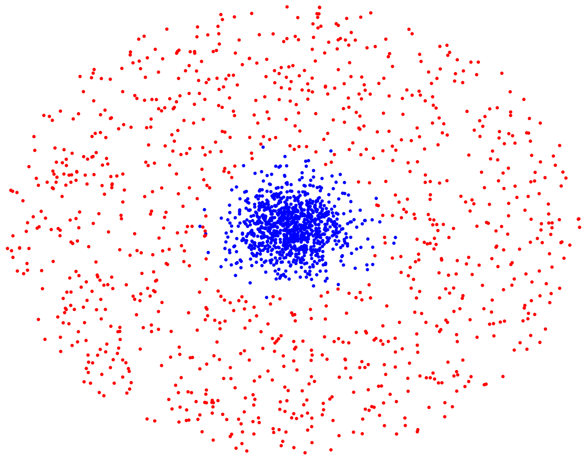


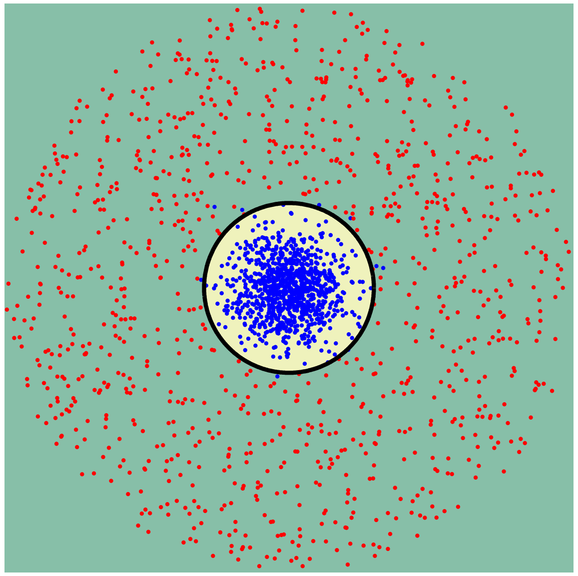
Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t;\ \epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(\mathbf{x}_i)]\!]$





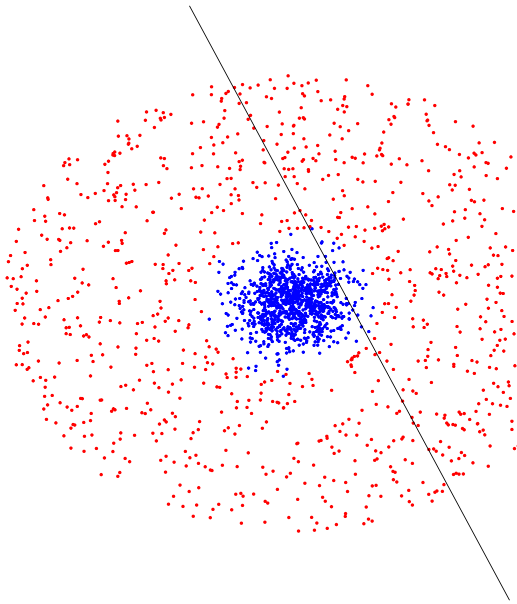
Summary of the Algorithm

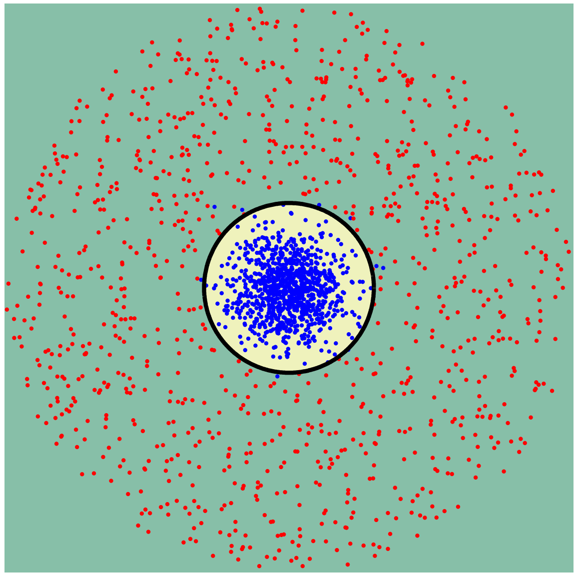
Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop

$t = 1$





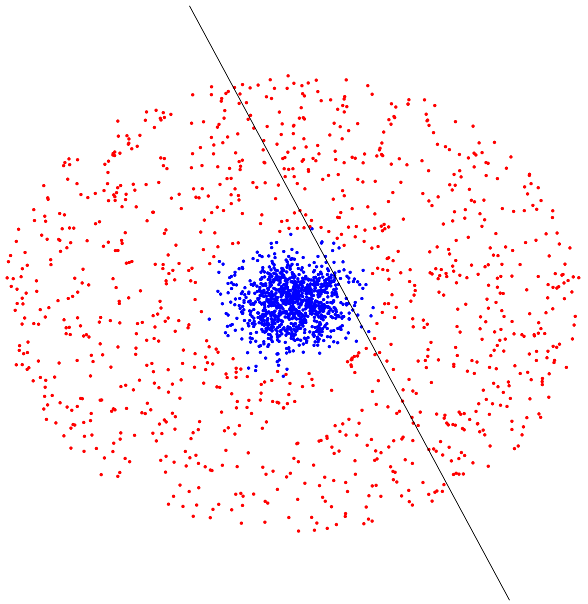
Summary of the Algorithm

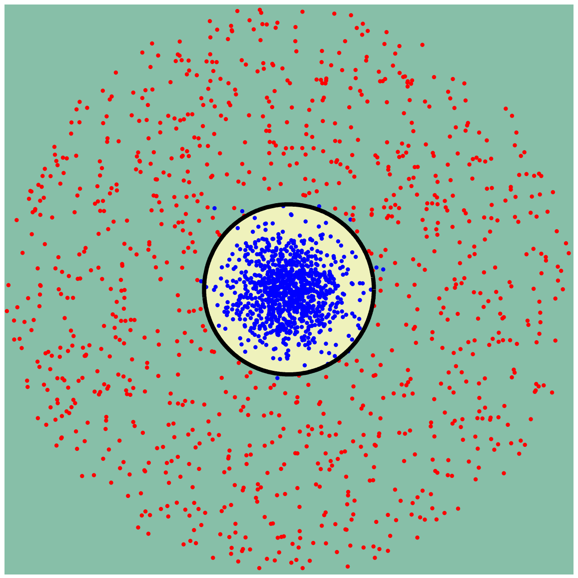
Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$

$t = 1$





Summary of the Algorithm

Initialization ...

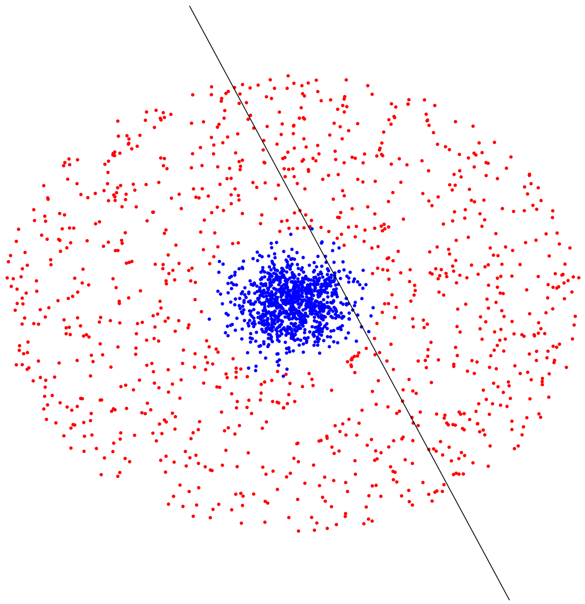
For $t = 1, ..., T$:

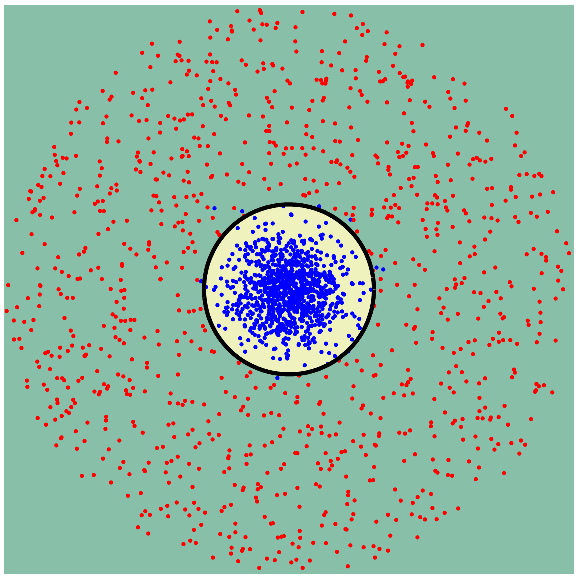
- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t; \;\; \epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

$$Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

$t = 1$





Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t;\ \ \epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

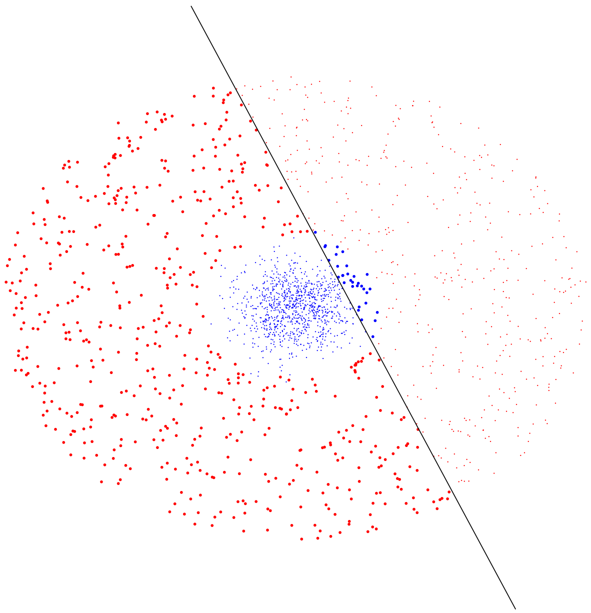
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

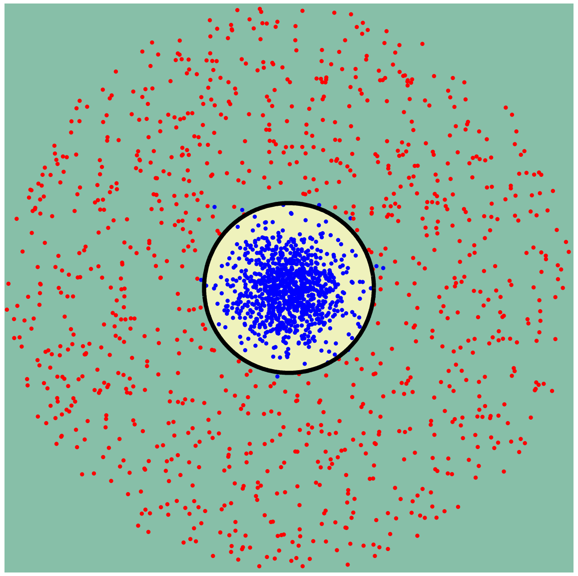
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

$t = 1$





Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t; \;\; \epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

$$Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

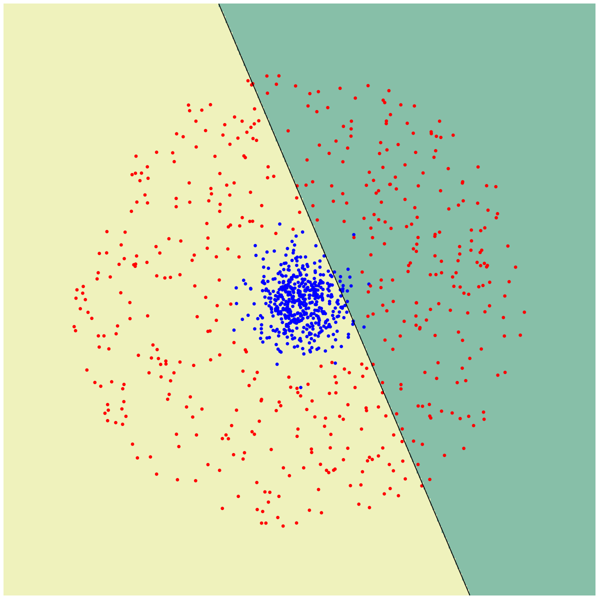
Output the final classifier:

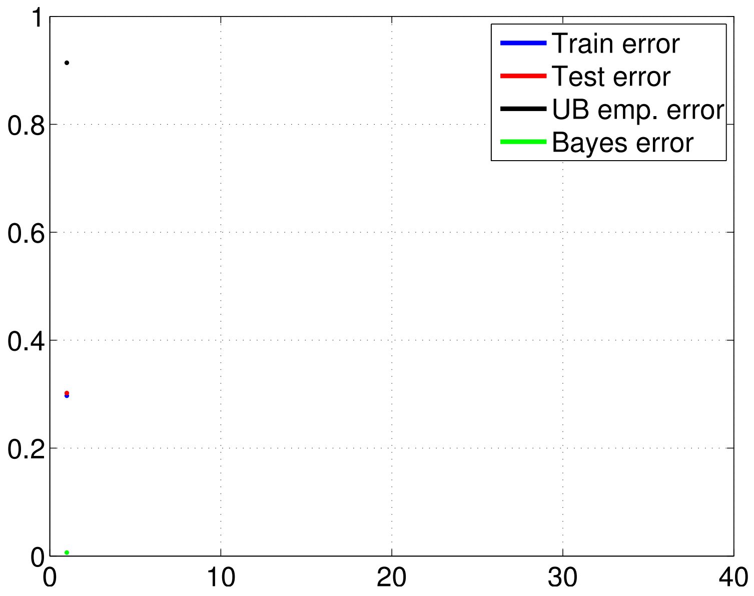
$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

$t = 1$





Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t; \ \epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

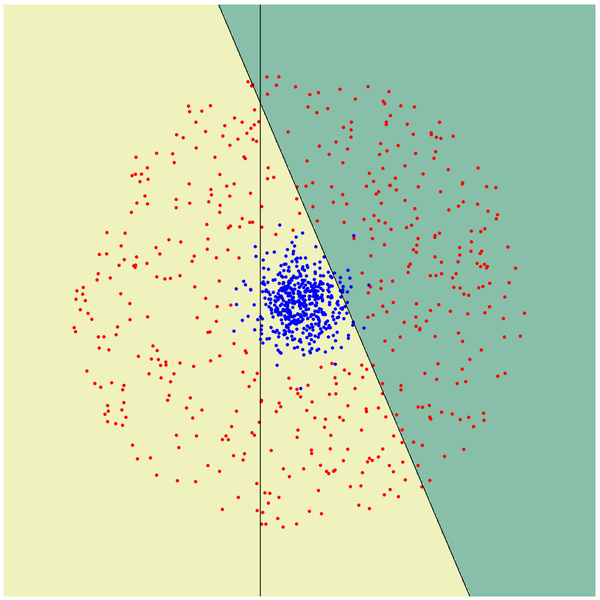
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

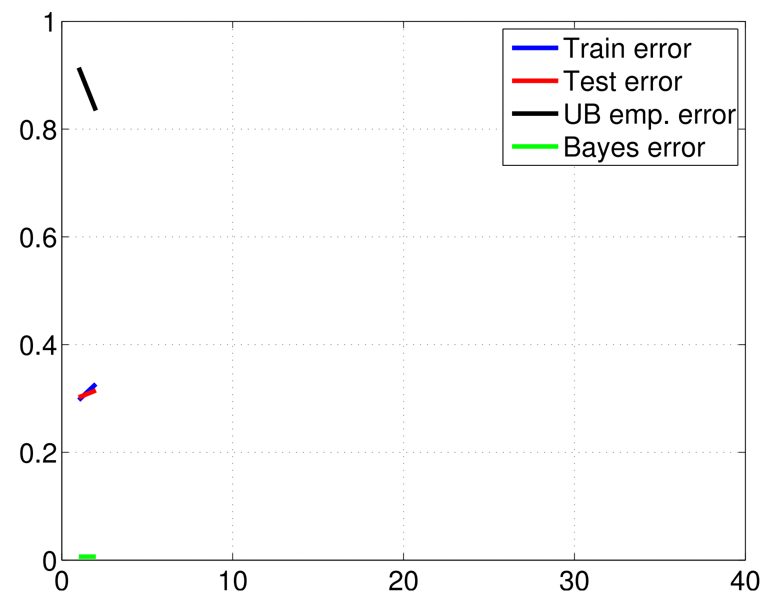
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

$t = 2$





Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

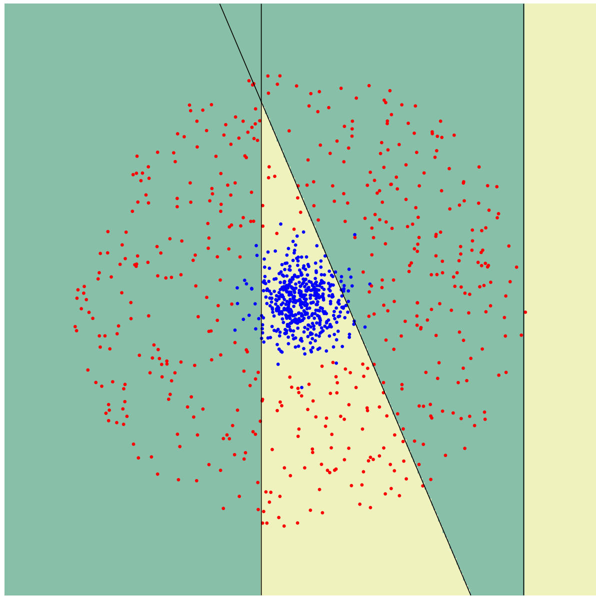
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

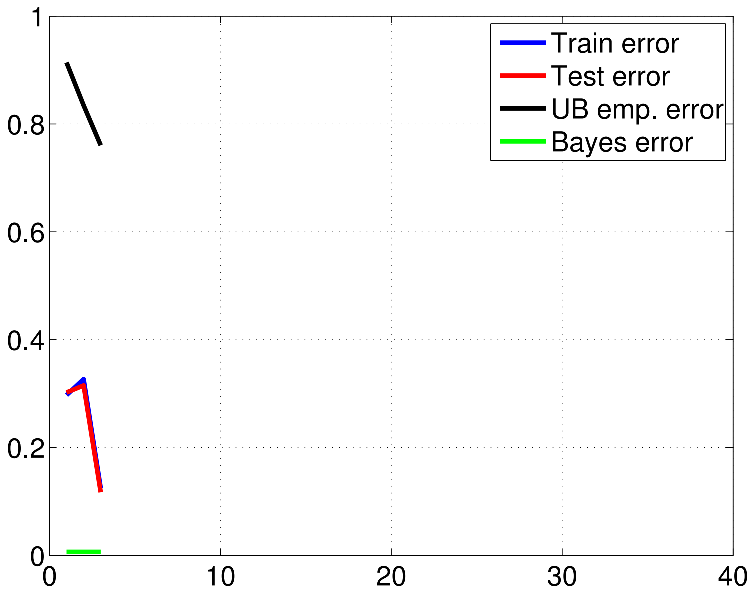
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

$t = 3$





Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t; \;\; \epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

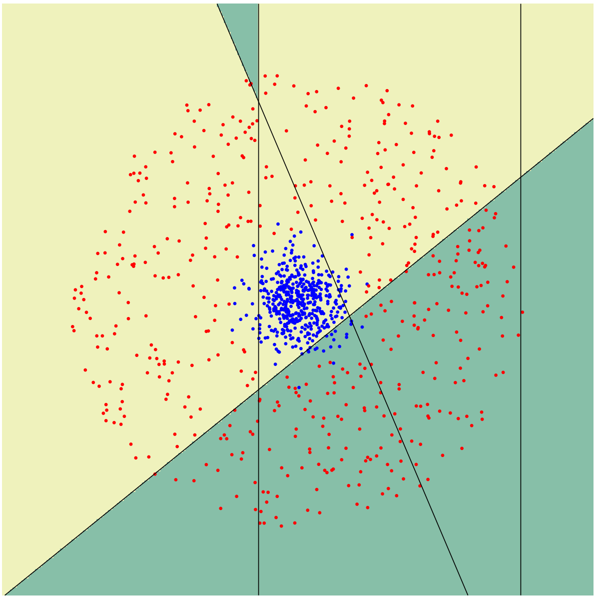
Output the final classifier:

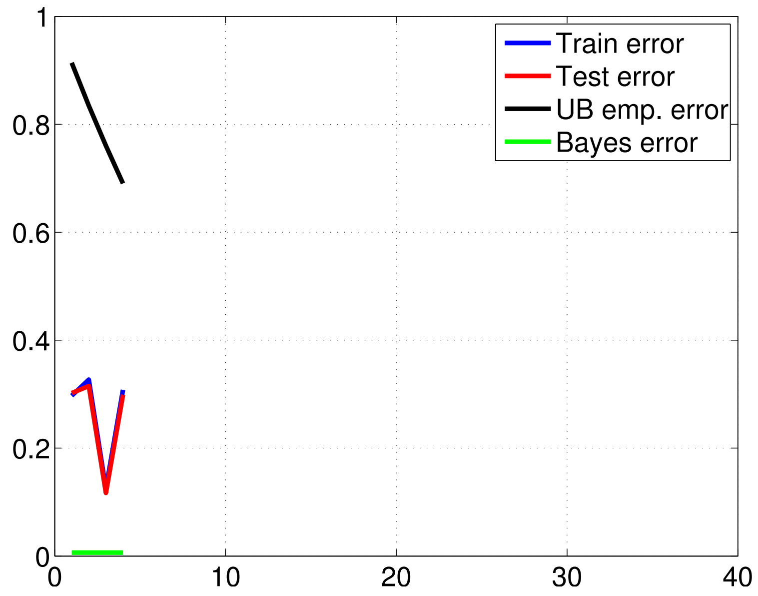
$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

$t = 4$





Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t; \; \epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

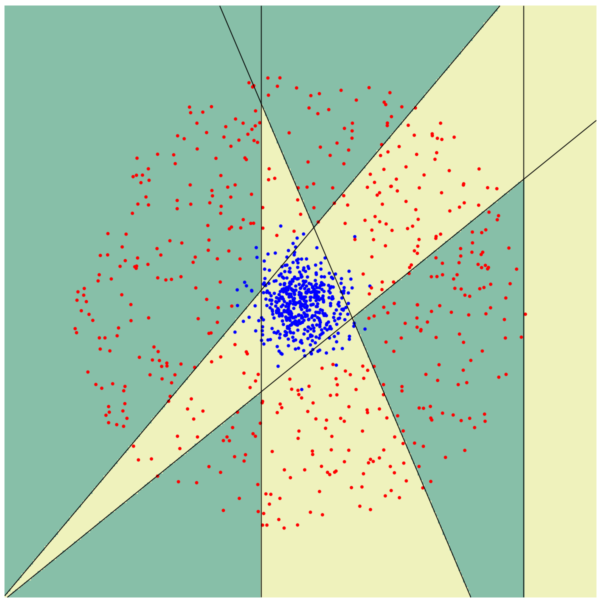
Output the final classifier:

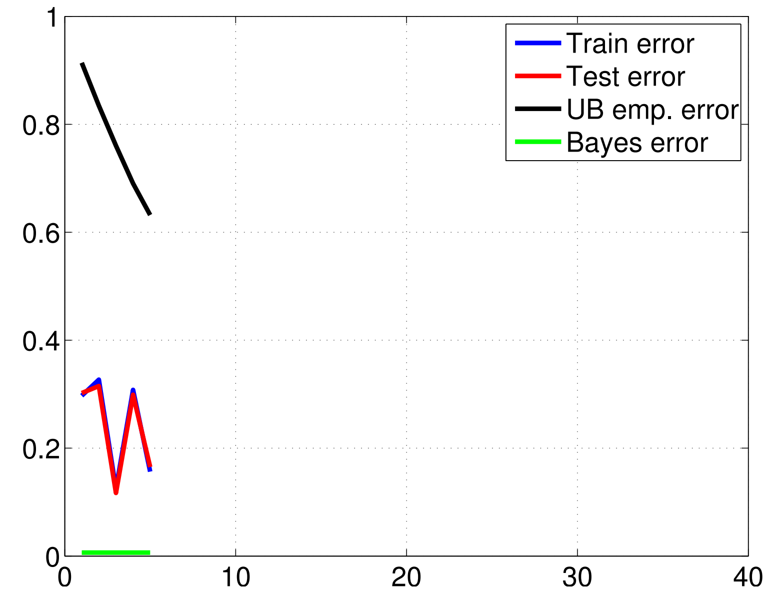
$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected “features” is captured in D_t

$t = 5$





Summary of the Algorithm

Initialization ...

For $t = 1, \ldots, T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

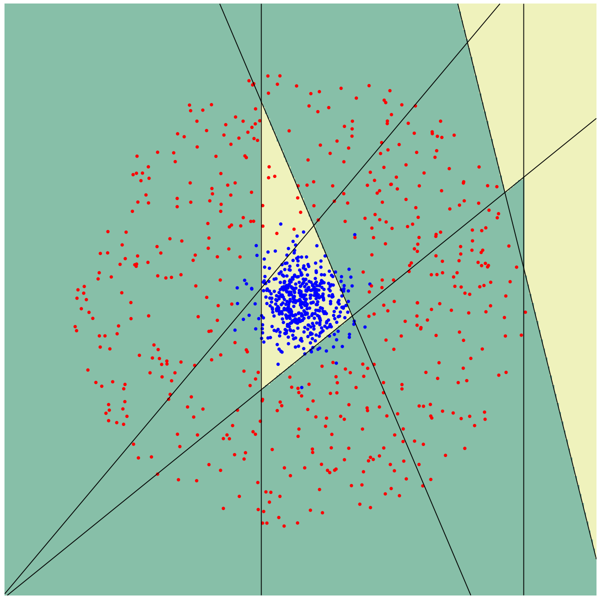
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

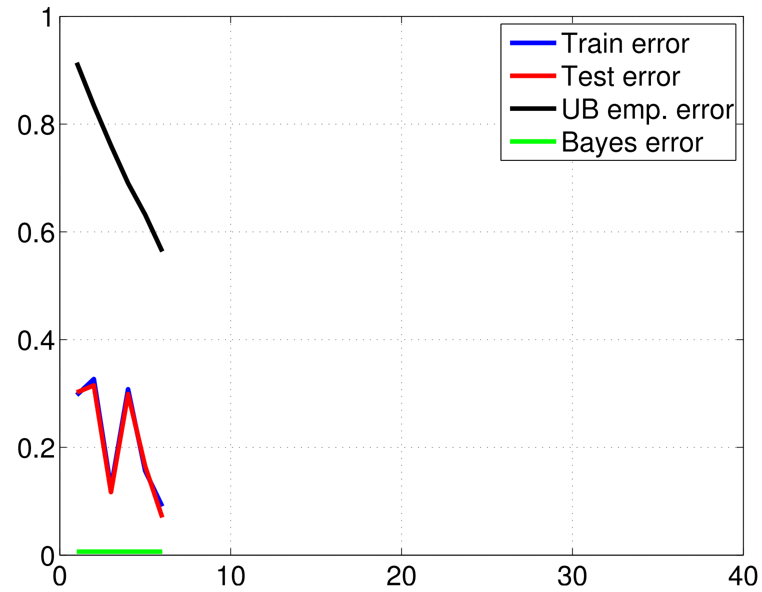
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

$t = 6$





Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

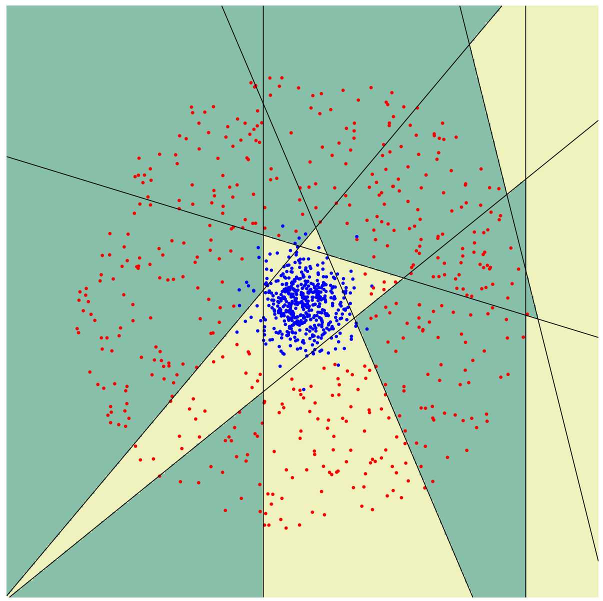
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

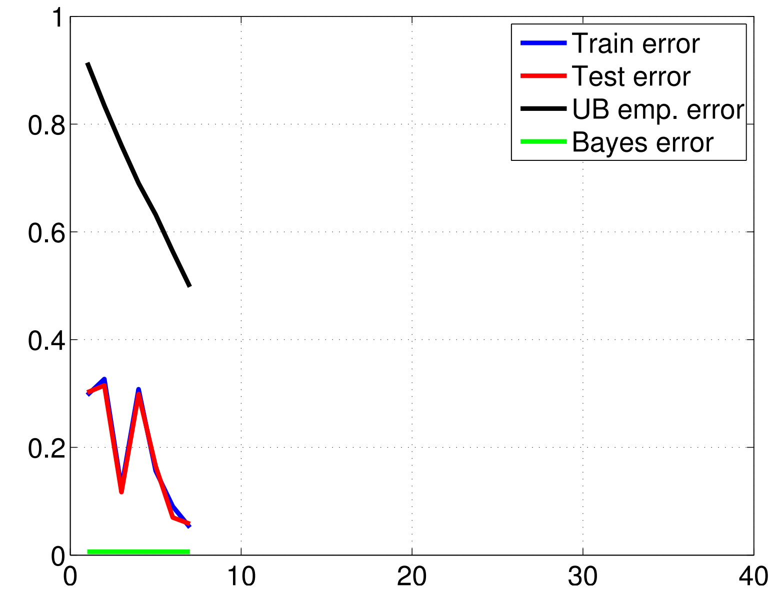
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

$t = 7$





Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected "features" is captured in D_t

Summary of the Algorithm

Initialization ...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h(\mathbf{x}_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

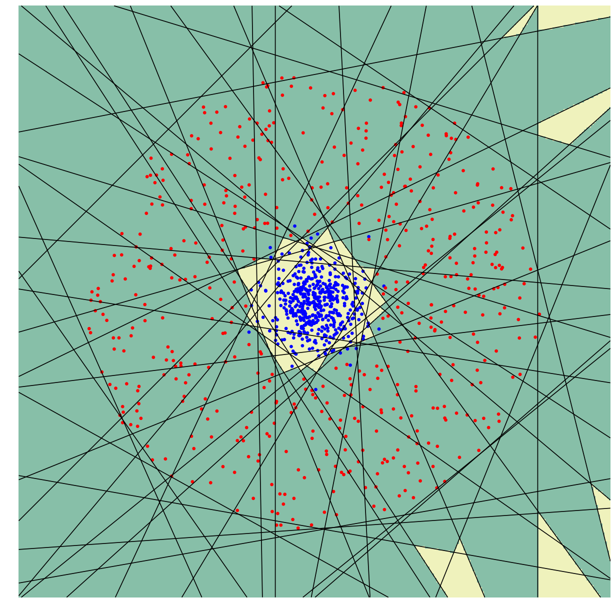
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

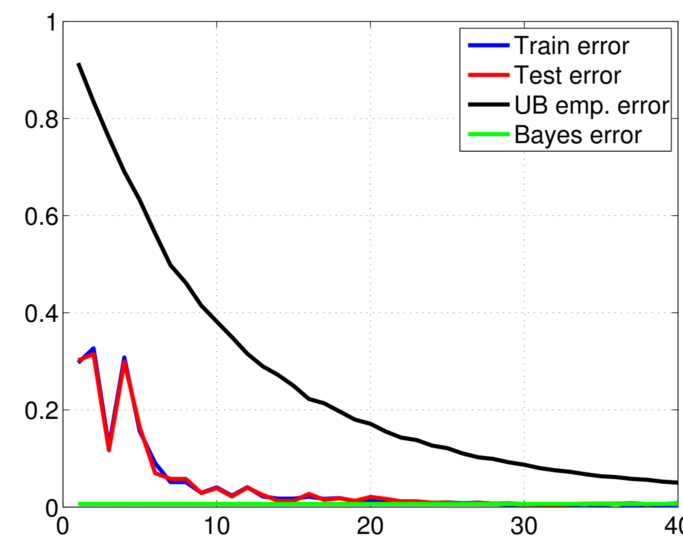
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

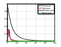
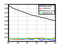
Output the final classifier:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$$

$t = 40$





100 steps  detail 

Comments

- The computational complexity of selecting h_t is independent of t
- All information about previously selected “features” is captured in D_t

Does AdaBoost generalise?

Margins in SVM

$$\max \min_{(\mathbf{x},y)\in\mathcal{T}} \frac{y(\vec{\alpha}\cdot\vec{h}(\mathbf{x}))}{\|\vec{\alpha}\|_2}$$

Margins in AdaBoost

$$\max \min_{(\mathbf{x},y)\in\mathcal{T}} \frac{y(\vec{\alpha}\cdot\vec{h}(\mathbf{x}))}{\|\vec{\alpha}\|_1}$$

Maximising margins in AdaBoost

$$P_S[yf(\mathbf{x}) \leq \theta] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}} \qquad \text{where } f(\mathbf{x}) = \frac{\vec{\alpha}\cdot\vec{h}(\mathbf{x})}{\|\vec{\alpha}\|_1}$$

Upper bounds based on margin

$$P_{\mathcal{D}}[yf(\mathbf{x}) \leq 0] \leq P_S[yf(\mathbf{x}) \leq \theta] + \mathcal{O}\left(\frac{1}{\sqrt{L}}\left(\frac{d\log^2(L/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right)$$

Pros and cons of AdaBoost

Advantages

- Very simple to implement
- Feature selection on very large sets of features
- Fairly good generalisation
- Linear classifier with all its desirable properties.
- Output converges to the logarithm of likelihood ratio.
- Feature selector with a principled strategy (minimisation of upper bound on empirical error)
- Close to sequential decision making (it produces a sequence of gradually more complex classifiers).

Disadvantages

- Suboptimal solution for $\vec{\alpha}$
- Can overfit in the presence of noise

AdaBoost variants

Freund & Schapire 1995

- Discrete ($h : \mathcal{X} \to \{0, 1\}$)
- Multiclass AdaBoost.M1 ($h : \mathcal{X} \to \{0, 1, ..., k\}$)
- Multiclass AdaBoost.M2 ($h : \mathcal{X} \to [0, 1]^k$)
- Real valued AdaBoost.R ($Y = [0, 1]$, $h : \mathcal{X} \to [0, 1]$)

Schapire & Singer 1997

- Confidence rated prediction ($h : \mathcal{X} \to R$, two-class)
- Multilabel AdaBoost.MR, AdaBoost.MH (different formulation of minimised loss)

... Many other modifications since then (WaldBoost, cascaded AB, online AB, ...)