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

14.11.2014

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

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

where

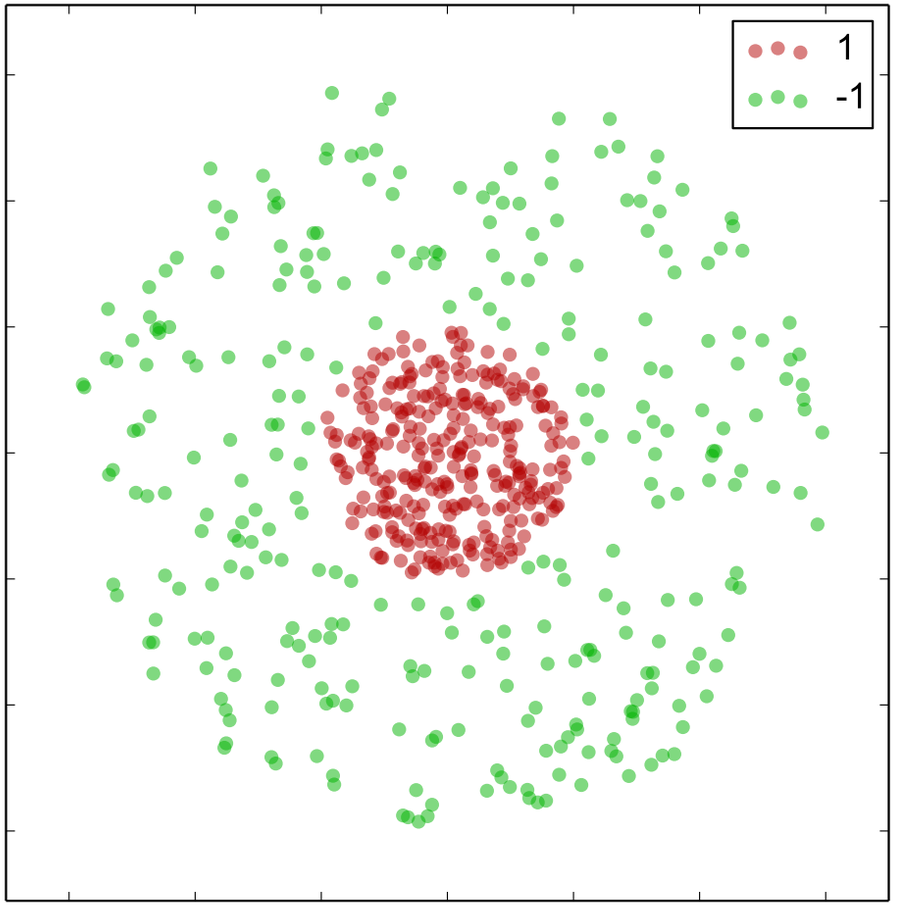
$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

is a linear combination of weak classifiers $h_t(x)$ 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 $h_t(x) \in \mathcal{B}$, and with choosing α_t, for gradually increasing t.

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

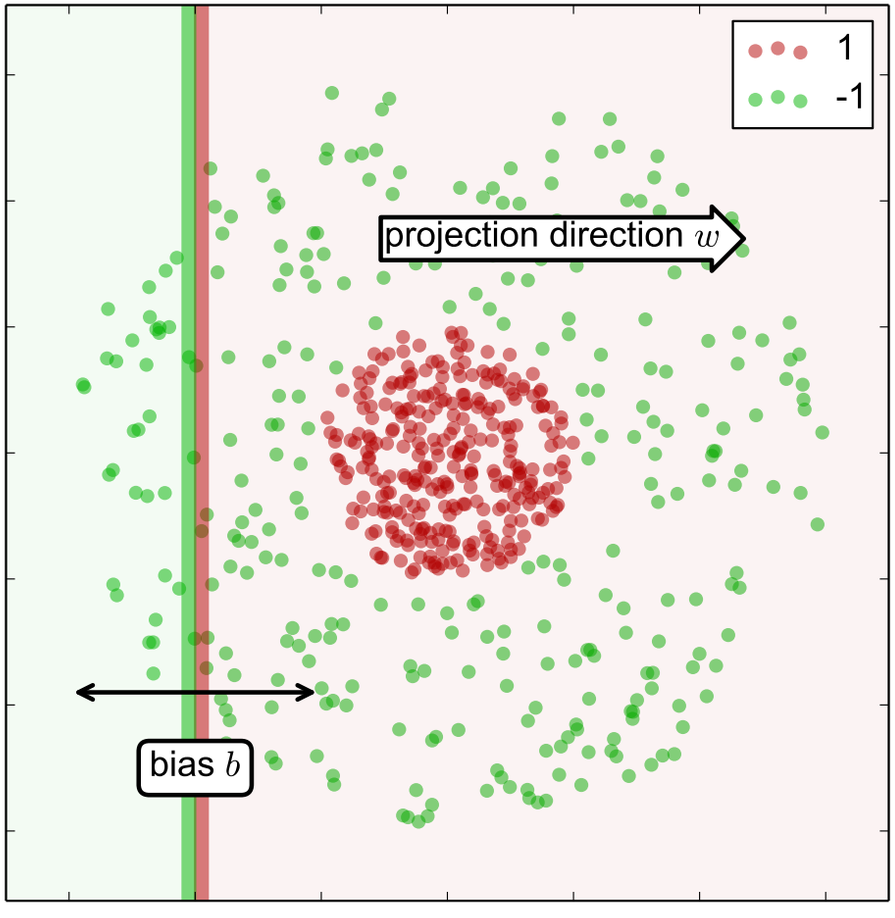
Example 1 – Dataset and Weak Classifier Set



Dataset: $(x_1, y_1), \ldots, (x_L, y_L)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Example 1 – Dataset and Weak Classifier Set



Dataset: $(x_1, y_1), \ldots, (x_L, y_L)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.

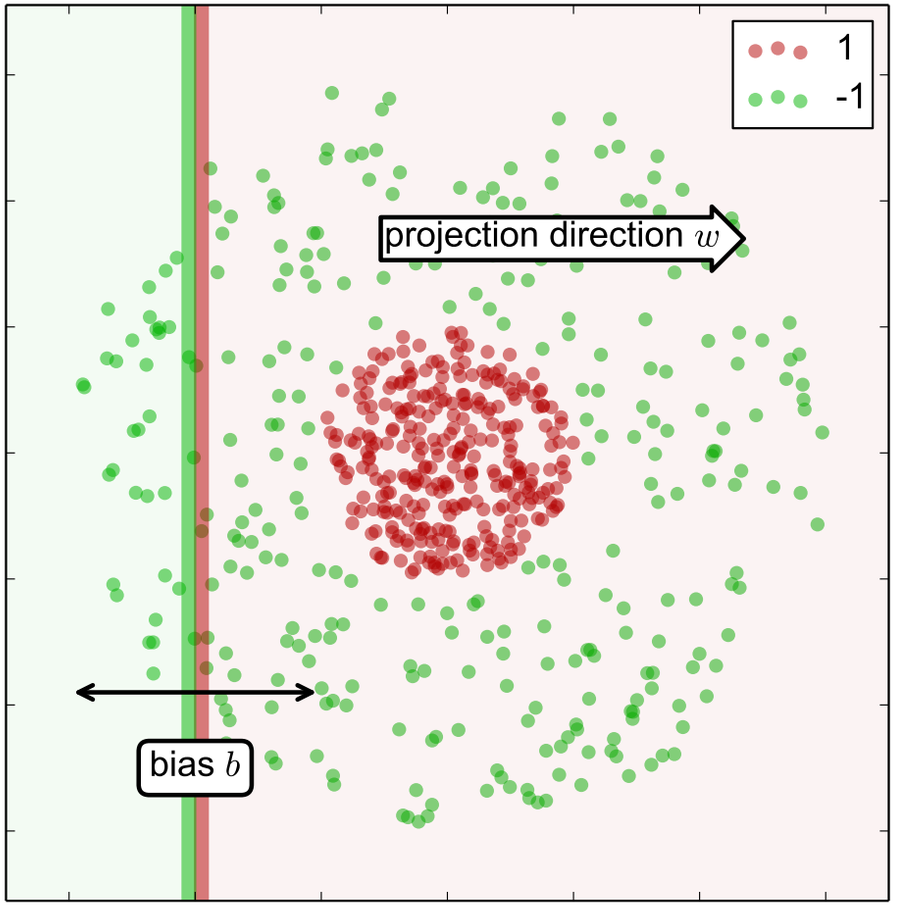
The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

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

where w is the projection direction vector and b is the bias.

Example 1 – Dataset and Weak Classifier Set



Dataset: $(x_1, y_1), \ldots, (x_L, y_L)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

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

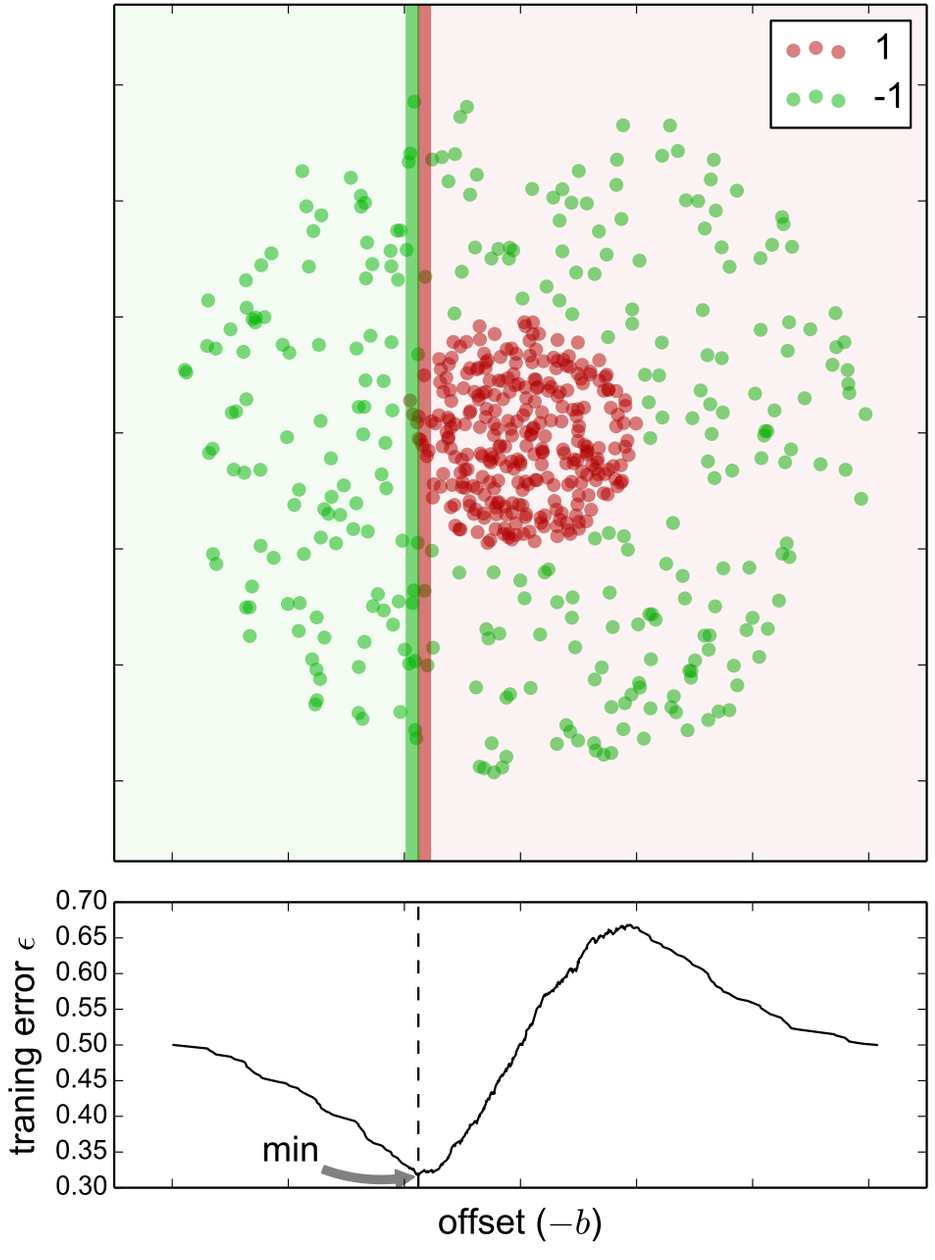
where w is the projection direction vector and b is the bias.

Weak classifier set $\mathcal{B}$:

$$\{h_{w,b} \mid w \in \{w_1, w_2, ..., w_N\}, b \in \mathbb{R}\}$$

- N is the number of projection directions used

Example 1 – Dataset and Weak Classifier Set



Dataset: $(x_1, y_1), \ldots, (x_L, y_L)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.

The two class distributions do not overlap (Bayes error is 0). The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

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

where w is the projection direction vector and b is the bias.

Weak classifier set $\mathcal{B}$:

$$\{h_{w,b} \mid w \in \{w_1, w_2, ..., w_N\}, b \in \mathbb{R}\}$$

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

AdaBoost Algorithm – Singer & Schapire (1997)

Input: $(x_1, y_1), \ldots, (x_L, y_L)$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$

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

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

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

  $[\![\text{true}]\!] \stackrel{\text{def}}{=} 1, \quad [\![\text{false}]\!] \stackrel{\text{def}}{=} 0$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Update

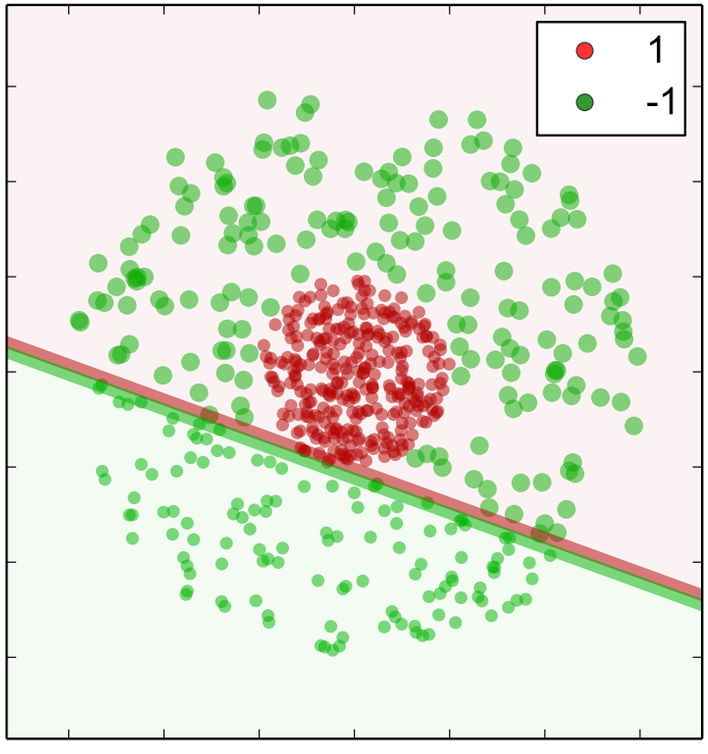
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}, \qquad Z_t = \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(x_i)},$$

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

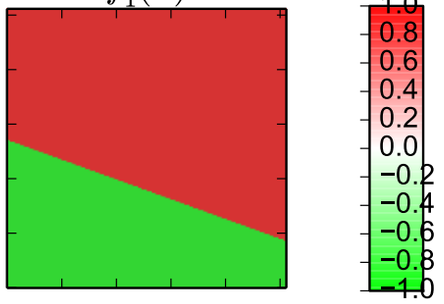
Output the final classifier:

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

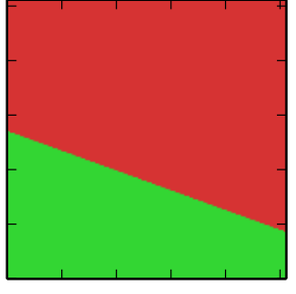
Example 1 – iteration 1



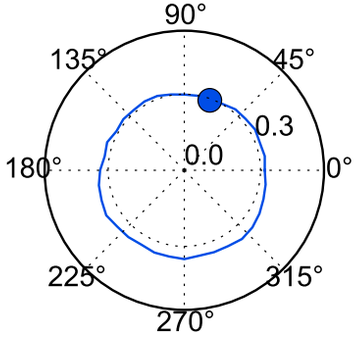
$f_1'(x)$

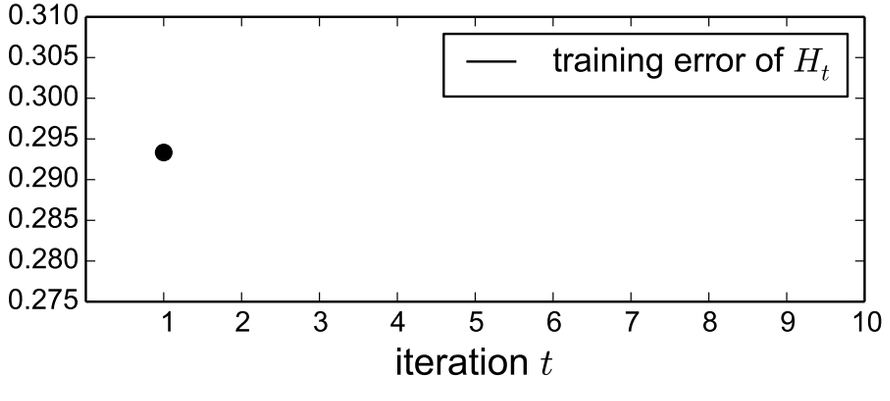


$H_1(x) = \text{sign}(f_1(x))$



$\epsilon_1(w)$
at optimal b

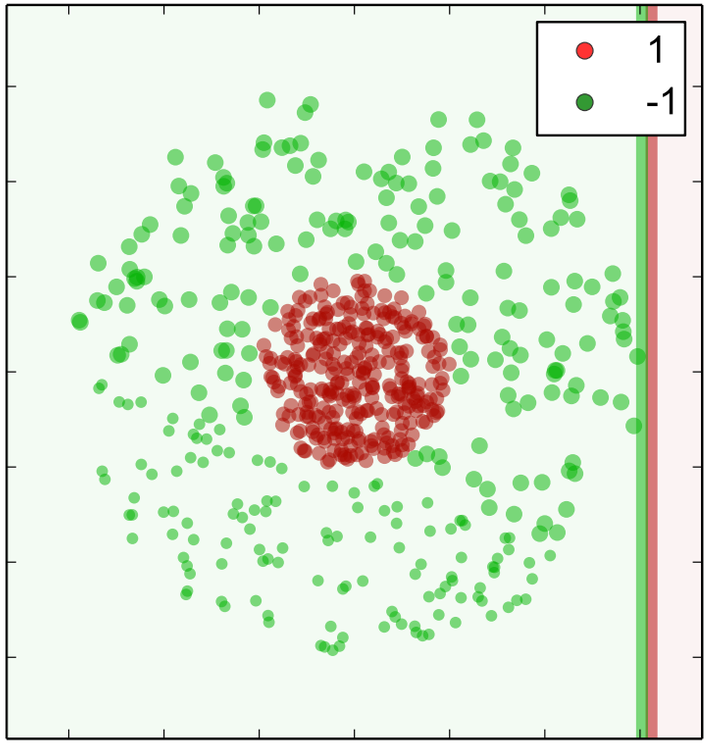




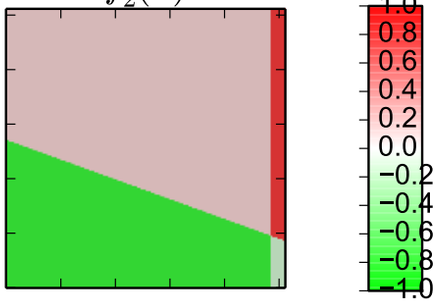
$t = 1$

- h_1 selected (note • in the polar plot). $N = 36$ directions w are used.
- $\epsilon_1 < 0.5$, continue
- $\alpha_1 = \frac{1}{2}\log(\frac{1-\epsilon_1}{\epsilon_1})$
- re-weighting D puts more weight to mis-classified samples in the • class
- $f_1(x) = \alpha_1 h_1(x)$
- $f_1'(x) = f_1(x)/\alpha_1$
- $H_1(x) = \text{sign}(f_1(x))$

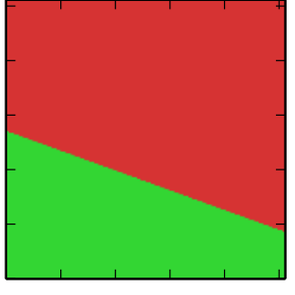
Example 1 – iteration 2



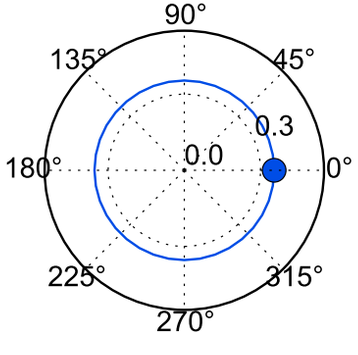
$f_2'(x)$

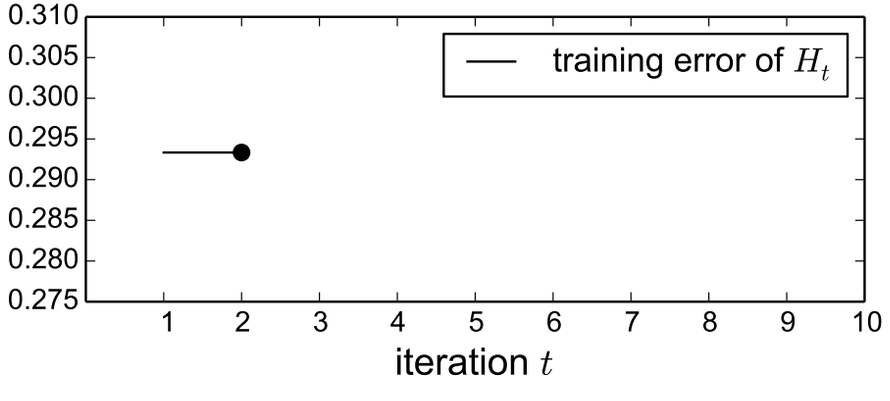


$H_2(x) = \text{sign}(f_2(x))$



$\epsilon_2(w)$
at optimal b





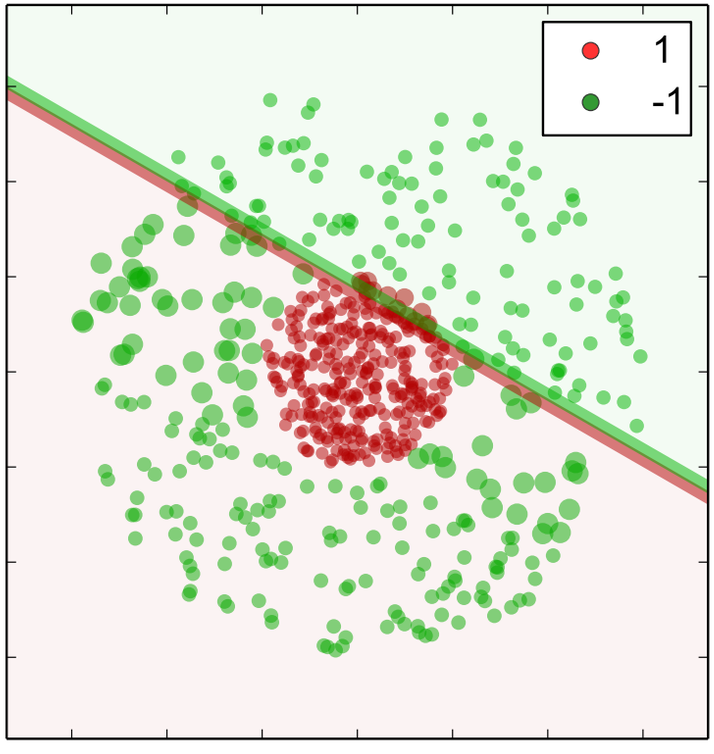
$t = 2$

- minimum errors $\epsilon_2(w)$ for all weak classifier directions w are equal. Everything is classified as -1 (●)
- this essentially re-weights the classes; gives more weight to class 1 (●)
- $f_2(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$
- $f_2'(x) = \frac{f_2(x)}{\alpha_1 + \alpha_2}$
- $H_2(x) = \text{sign}(f_2(x))$

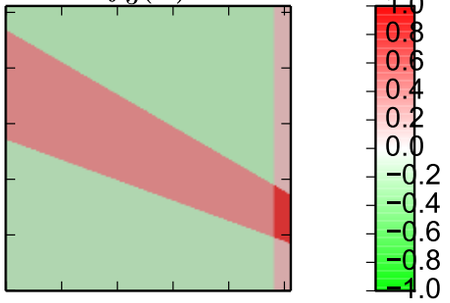
Quiz question:

- What is the difference between $f_2(x)$ and the previous $f_1(x)$? (Note that all points are classified as -1)

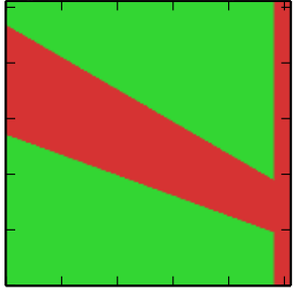
Example 1 – iteration 3



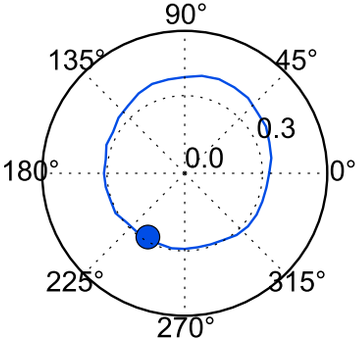
$f'_3(x)$

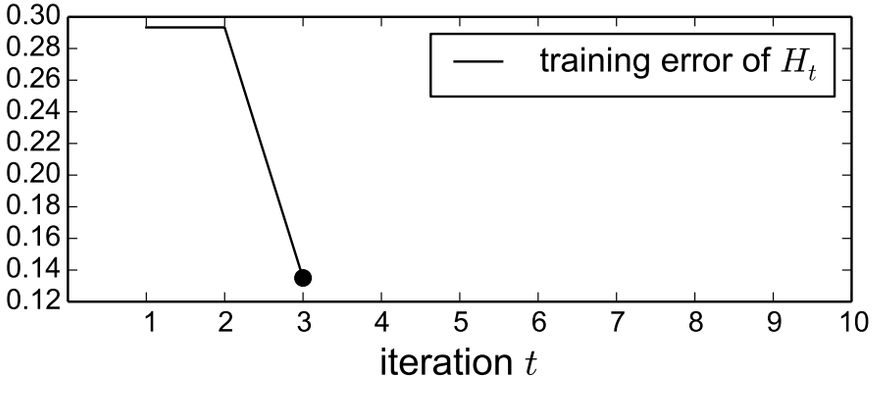


$H_3(x) = \text{sign}(f_3(x))$



$\epsilon_3(w)$
at optimal b

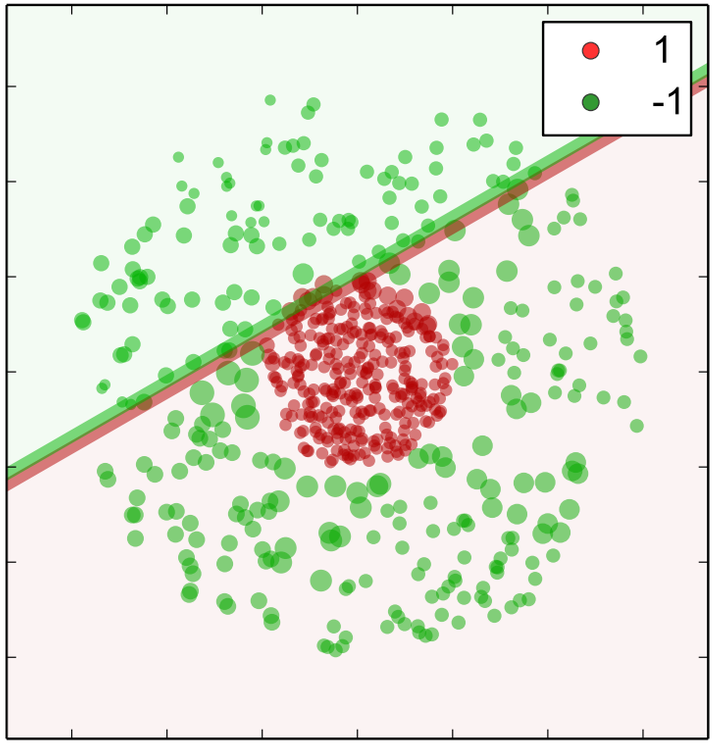




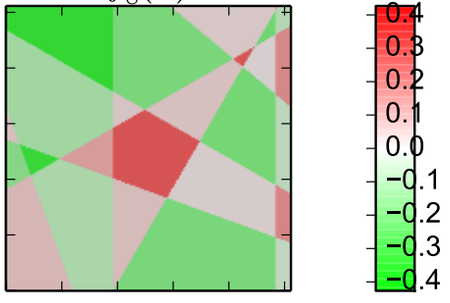
$t = 3$

- h_3 selected which minimizes $\epsilon_3(w)$ (note • in the polar plot).
- $\alpha_3 = \frac{1}{2}\log(\frac{1-\epsilon_3}{\epsilon_3})$
- distribution re-weighted
- $f_3(x) = \sum_{q=1}^{3} \alpha_q h_q(x)$
- $f'_3(x) = \frac{f(x)}{\sum_{q=1}^{3} \alpha_q}$
- $H_3(x) = \text{sign}(f_3(x))$

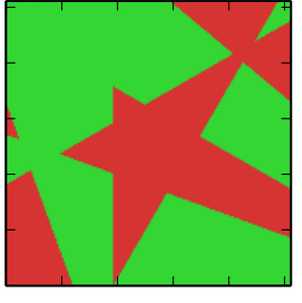
Example 1 – iteration 8



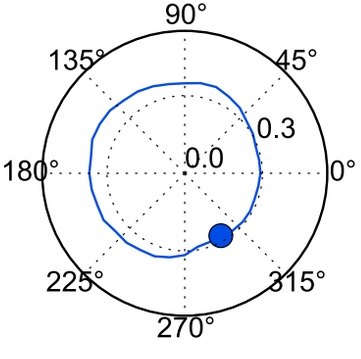
$f'_8(x)$

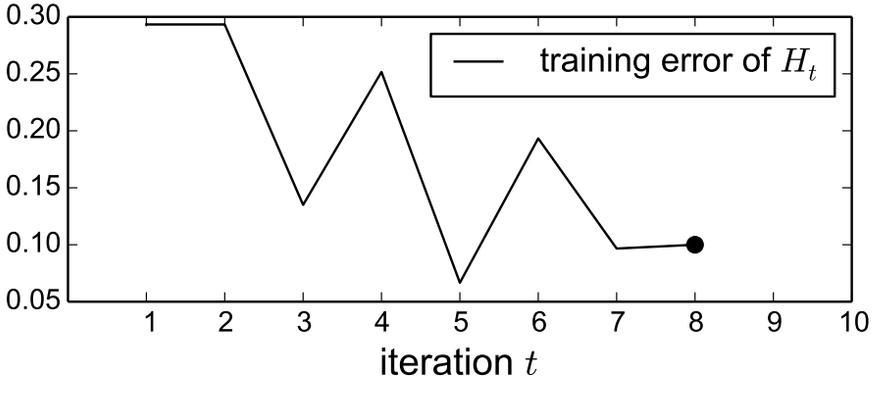


$H_8(x) = \text{sign}(f_8(x))$



$\epsilon_8(w)$
at optimal b

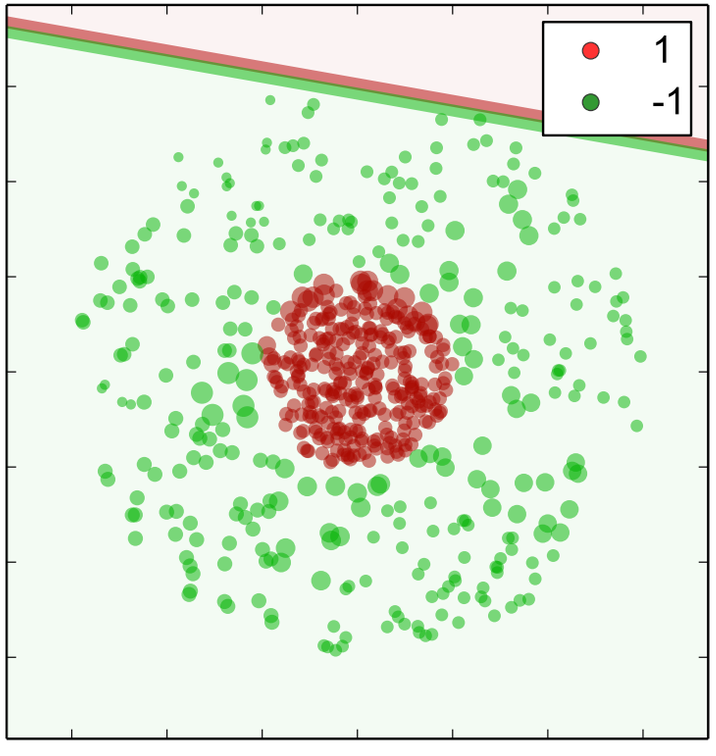




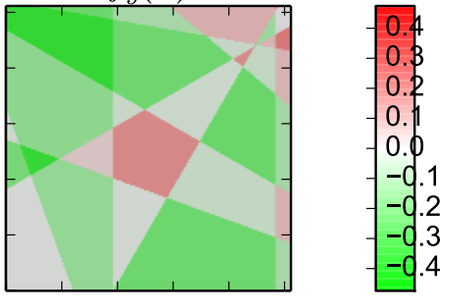
$t = 8$

- h_8 selected which minimizes $\epsilon_8(w)$ (note • in the polar plot).
- $\alpha_8 = \frac{1}{2}\log(\frac{1-\epsilon_8}{\epsilon_8})$
- distribution re-weighted
- $f_8(x) = \sum_{q=1}^{8} \alpha_q h_q(x)$
- $f'_8(x) = \frac{f(x)}{\sum_{q=1}^{8} \alpha_q}$
- $H_8(x) = \text{sign}(f_8(x))$

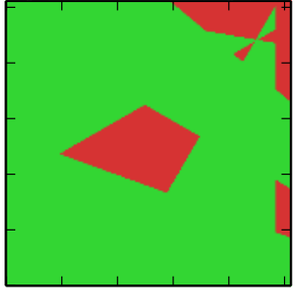
Example 1 – iteration 9



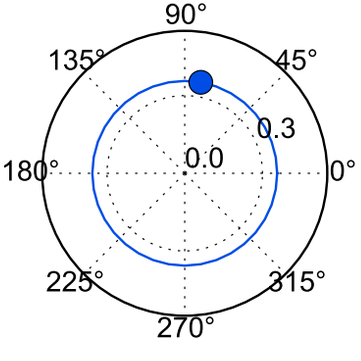
$f'_9(x)$

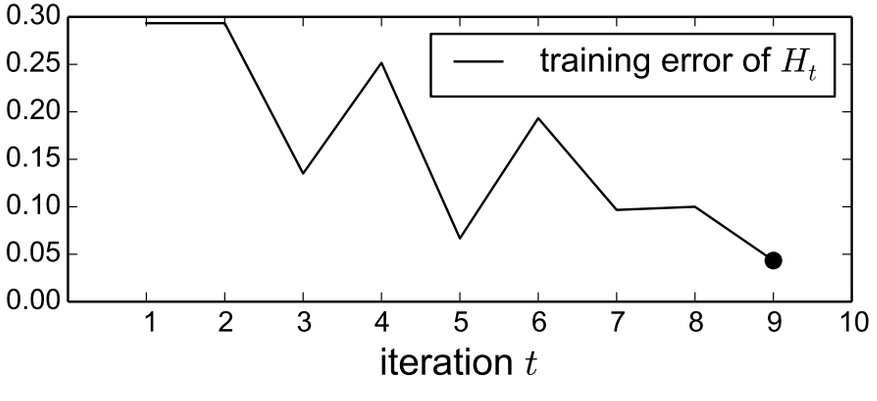


$H_9(x) = \text{sign}(f_9(x))$



$\epsilon_9(w)$
at optimal b

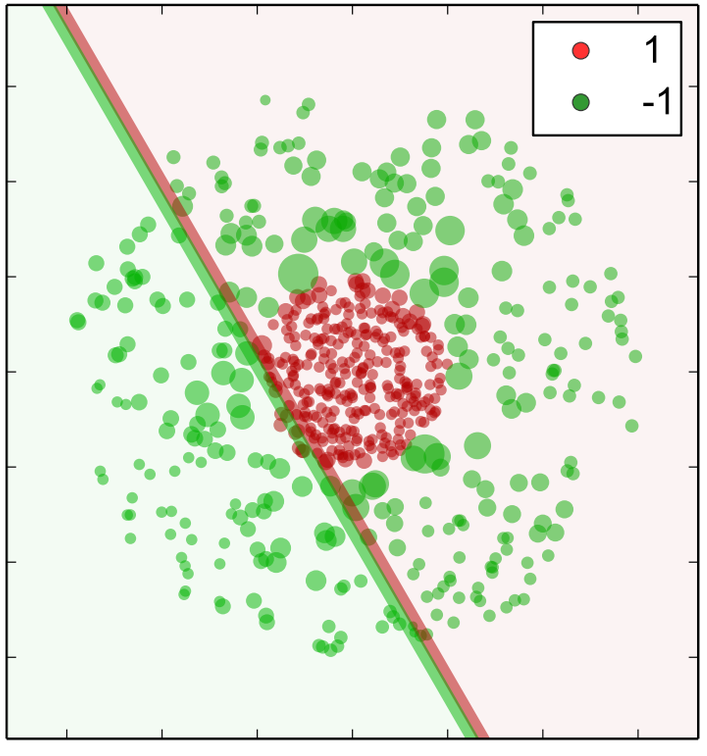




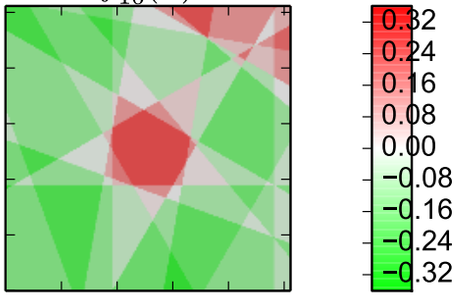
$t = 9$

- h_9 selected which minimizes $\epsilon_9(w)$ (note • in the polar plot).
- $\alpha_9 = \frac{1}{2}\log(\frac{1-\epsilon_9}{\epsilon_9})$
- distribution re-weighted
- $f_9(x) = \sum_{q=1}^{9} \alpha_q h_q(x)$
- $f'_9(x) = \frac{f(x)}{\sum_{q=1}^{9} \alpha_q}$
- $H_9(x) = \text{sign}(f_9(x))$

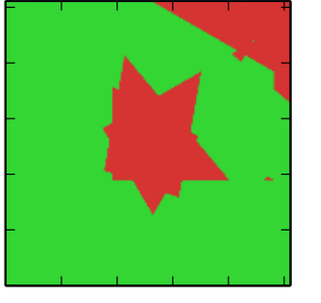
Example 1 – iteration 15



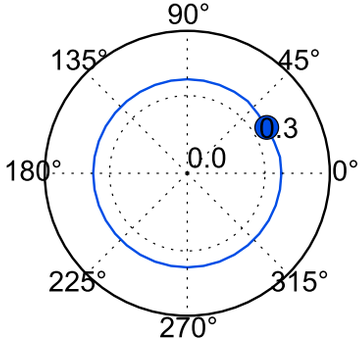
$f'_{16}(x)$

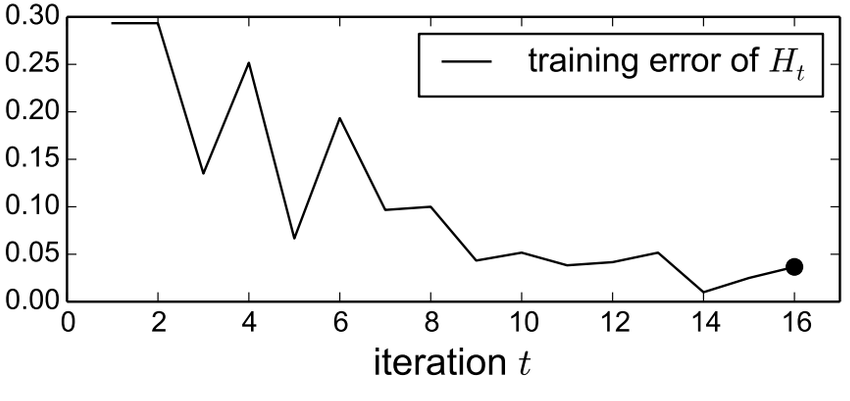


$H_{16}(x) = \text{sign}(f_{16}(x))$



$\epsilon_{16}(w)$
at optimal b

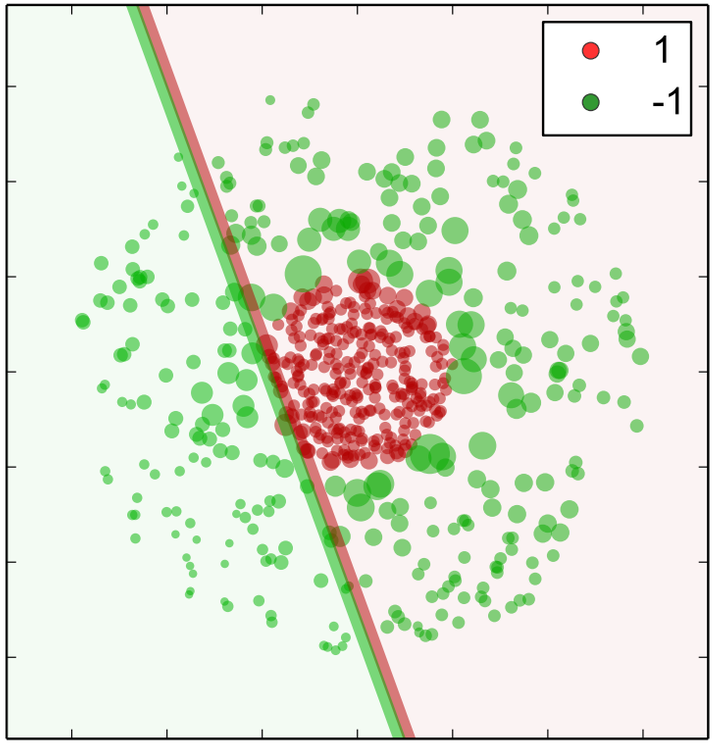




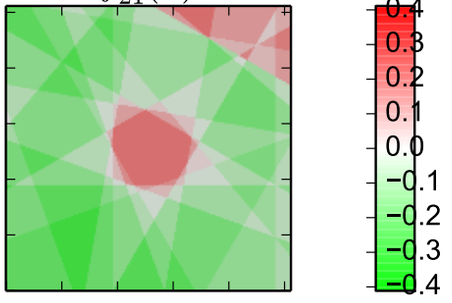
$t = 15$

- h_{15} selected which minimizes $\epsilon_{15}(w)$ (note • in the polar plot).
- $\alpha_{15} = \frac{1}{2}\log(\frac{1-\epsilon_{15}}{\epsilon_{15}})$
- distribution re-weighted
- $f_{15}(x) = \sum_{q=1}^{15} \alpha_q h_q(x)$
- $f'_{15}(x) = \frac{f(x)}{\sum_{q=1}^{15} \alpha_q}$
- $H_{15}(x) = \text{sign}(f_{15}(x))$

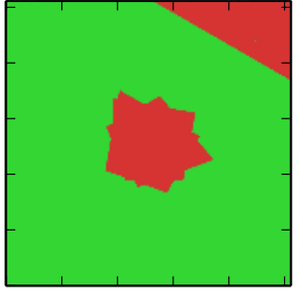
Example 1 – iteration 21



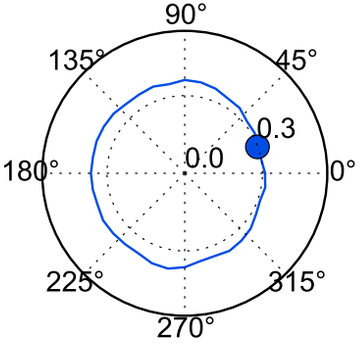
$f'_{21}(x)$

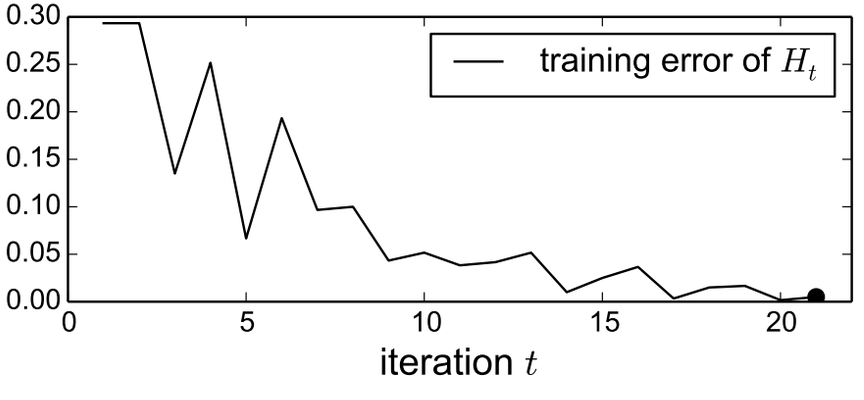


$H_{21}(x) = \text{sign}(f_{21}(x))$



$\epsilon_{21}(w)$
at optimal b

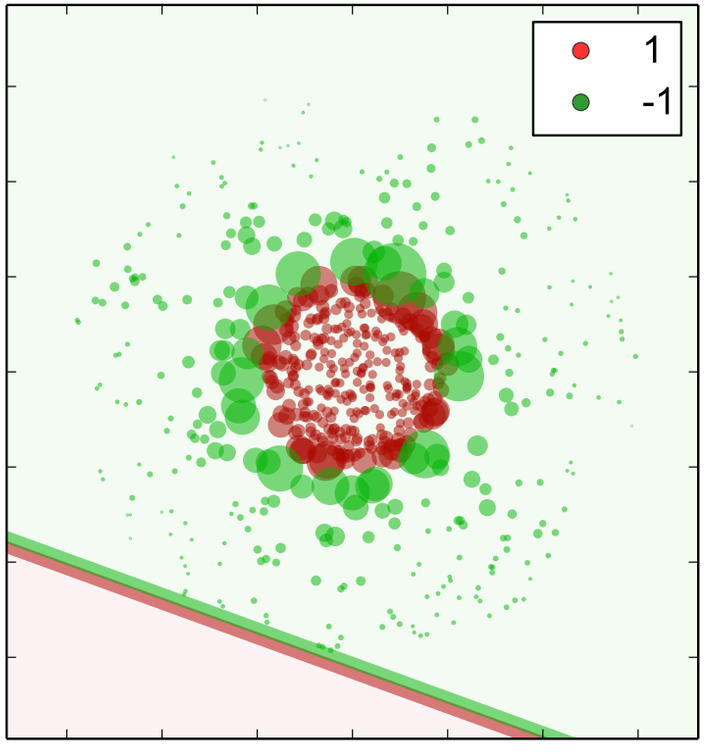




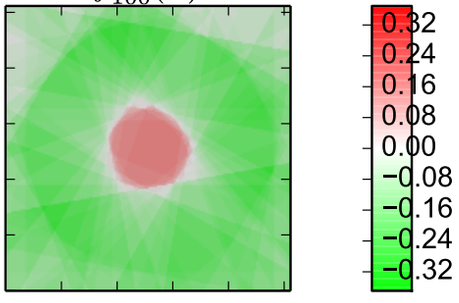
$t = 21$

- h_{21} selected which minimizes $\epsilon_{21}(w)$ (note • in the polar plot).
- $\alpha_{21} = \frac{1}{2}\log(\frac{1-\epsilon_{21}}{\epsilon_{21}})$
- distribution re-weighted
- $f_{21}(x) = \sum_{q=1}^{21} \alpha_q h_q(x)$
- $f'_{21}(x) = \frac{f(x)}{\sum_{q=1}^{21} \alpha_q}$
- $H_{21}(x) = \text{sign}(f_{21}(x))$

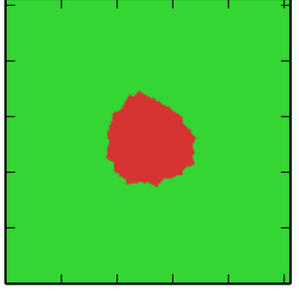
Example 1 – iteration 100



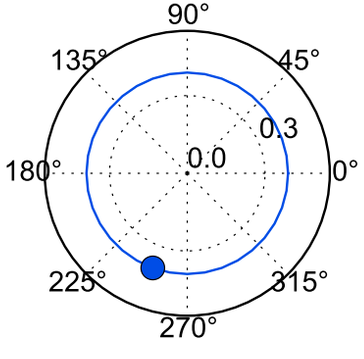
$f'_{100}(x)$

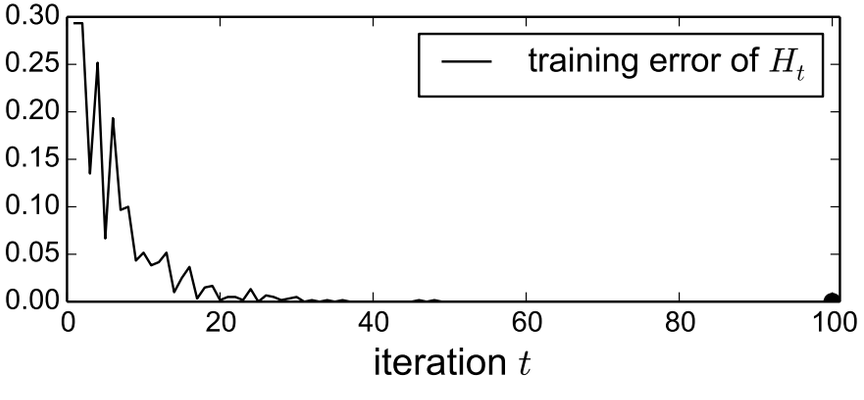


$H_{100}(x) = \text{sign}(f_{100}(x))$



$\epsilon_{100}(w)$
at optimal b

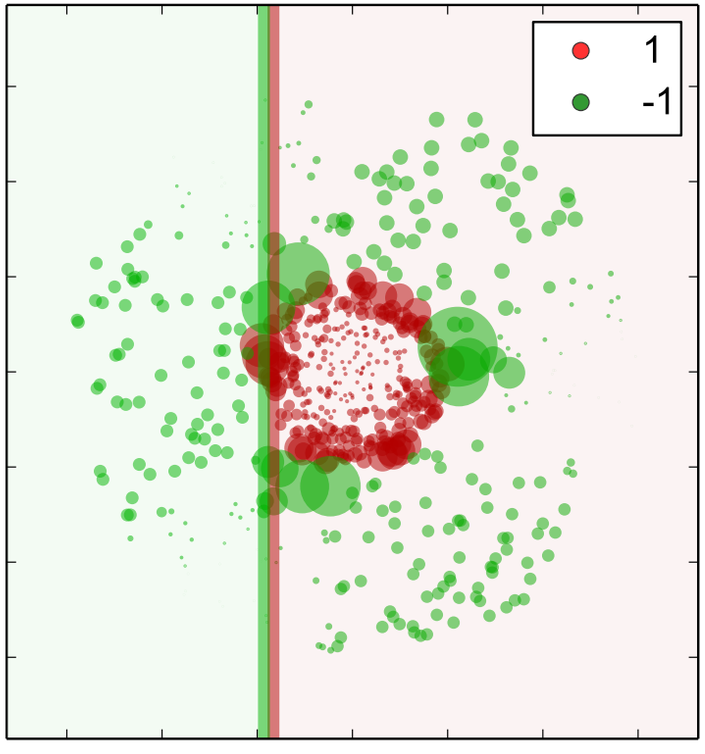




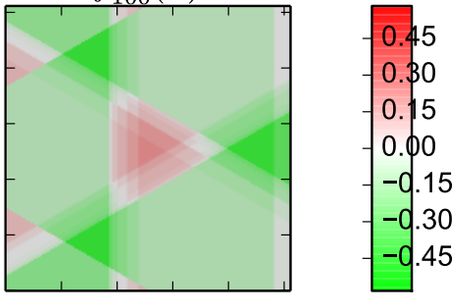
$t = 100$

- h_{100} selected which minimizes $\epsilon_{100}(w)$ (note • in the polar plot).
- $\alpha_{100} = \frac{1}{2}\log(\frac{1-\epsilon_{100}}{\epsilon_{100}})$
- distribution re-weighted
- $f_{100}(x) = \sum_{q=1}^{100} \alpha_q h_q(x)$
- $f'_{100}(x) = \frac{f(x)}{\sum_{q=1}^{100} \alpha_q}$
- $H_{100}(x) = \text{sign}(f_{100}(x))$

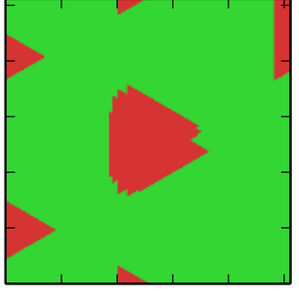
Example 2, $N = 3$ directions only, iteration 100



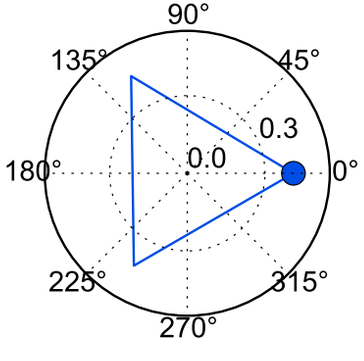
$f'_{100}(x)$

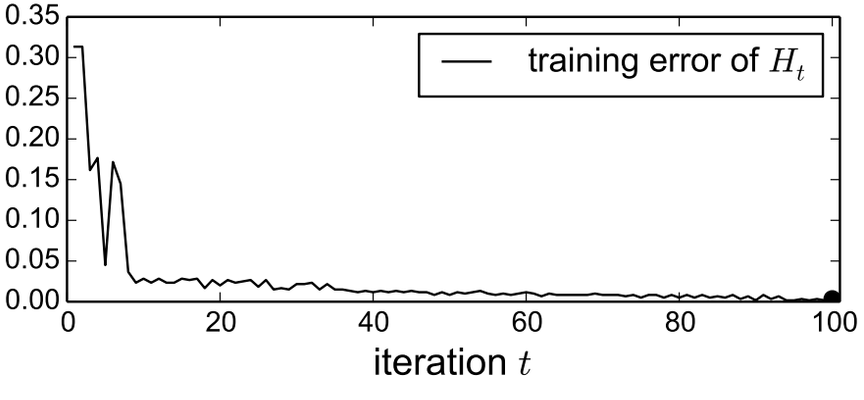


$H_{100}(x) = \text{sign}(f_{100}(x))$



$\epsilon_{100}(w)$
at optimal b





$t = 100$

- h_{100} selected which minimizes $\epsilon_{100}(w)$ (note • in the polar plot).
- $\alpha_{100} = \frac{1}{2}\log(\frac{1-\epsilon_{100}}{\epsilon_{100}})$
- distribution re-weighted
- $f_{100}(x) = \sum_{q=1}^{100} \alpha_q h_q(x)$
- $f'_{100}(x) = \frac{f(x)}{\sum_{q=1}^{100} \alpha_q}$
- $H_{100}(x) = \text{sign}(f_{100}(x))$

Upper bound theorem (1/2)

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

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

Proof: Firstly, let us check the following table:

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

From the first and last columns, it follows that for all (x_i, y_i), there holds:

$$[\![H_T(x_i) \neq y_i]\!] \leq \exp(-y_i f_T(x_i)) \,.$$

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

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

Upper bound theorem (2/2)

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

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

Proof (contd.):

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

But from the distribution update rule:

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

we have that

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

which completes the proof.

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

- The main objective is to minimize $\epsilon = \frac{1}{L}\sum_{i=1}^{L}[\![y_i \neq H_T(x_i)]\!]$ (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

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

given the dataset $\{(x_i, y_i)\}$ 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 $Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}$ with respect to α_t:

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

Choosing α_t

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

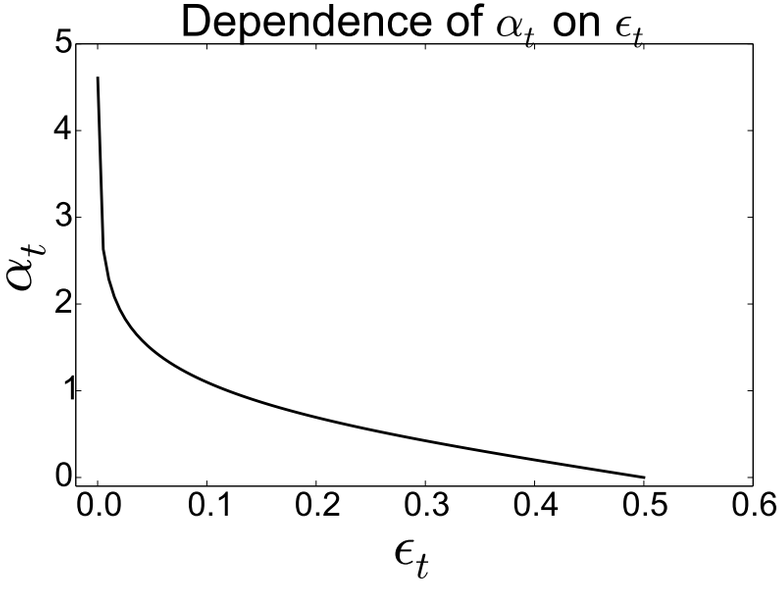
$$\frac{\mathrm{d}Z}{\mathrm{d}\alpha_t} = -\sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(x_i)} y_i h_t(x_i) = 0$$

$$-\underbrace{\sum_{i:y_i=h_t(x_i)} D_t(i)}_{1-\epsilon_t} e^{-\alpha_t} + \underbrace{\sum_{i:y_i\neq h_t(x_i)} D(i)}_{\epsilon_t} e^{\alpha_t} = 0$$

$$-e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t = 0$$

$$\alpha_t + \log \epsilon_t = -\alpha_t + \log(1-\epsilon_t)$$

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



Choosing h_t

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

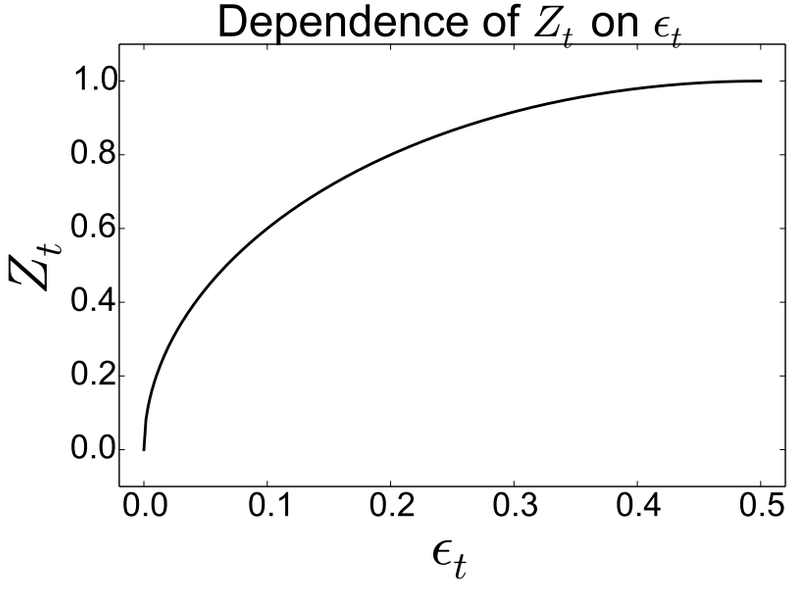
$$\begin{aligned} Z_t &= \sum_{i=1}^{L} D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\ &= \sum_{i:y_i=h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i\neq h_t(x_i)} D_t(i) e^{\alpha_t} \\ &= (1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

$\Rightarrow$ Z_t is minimised by selecting h_t with minimal weighted error ϵ_t.

Choosing h_t

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

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



$\Rightarrow$ 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 $\alpha_t \quad \Rightarrow \quad Z_t$ is minimized by h_t with minimal ϵ_t.

Comments

- The process of selecting α_t and $h_t(x)$ 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 $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots)$ starting from $\vec{\alpha}_0 = (0, 0, \ldots)$.

Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} = \frac{e^{-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)}}{L\Pi_{q=1}^{t} Z_q}$$

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & < 1, \quad y_i = h_t(x_i) \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & > 1, \quad y_i \neq h_t(x_i) \end{cases}$$

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example.

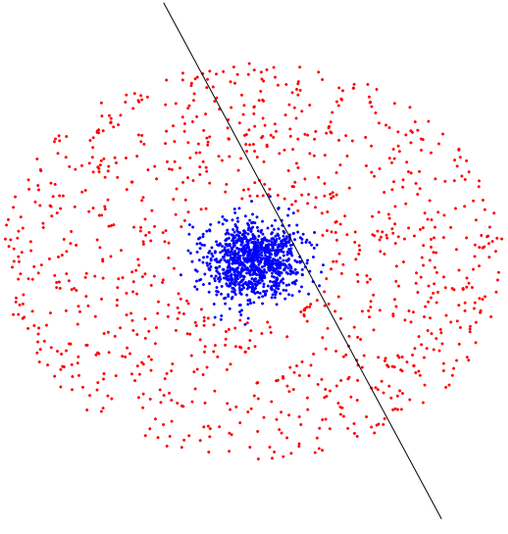
Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} = \frac{e^{-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)}}{L\Pi_{q=1}^{t} Z_q}$$

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & < 1, \quad y_i = h_t(x_i) \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & > 1, \quad y_i \neq h_t(x_i) \end{cases}$$



$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example.

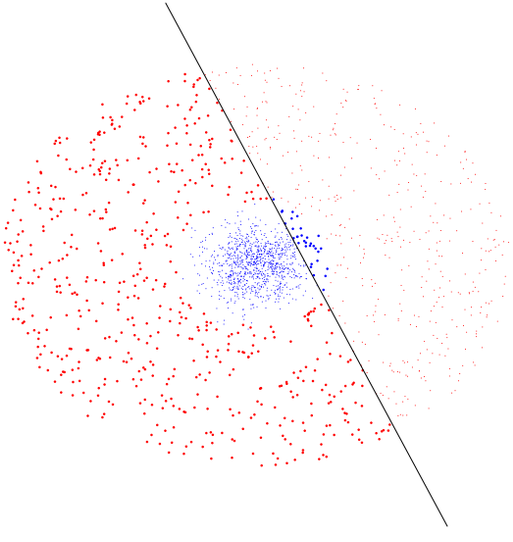
Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} = \frac{e^{-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)}}{L\Pi_{q=1}^{t} Z_q}$$

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & < 1, \quad y_i = h_t(x_i) \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & > 1, \quad y_i \neq h_t(x_i) \end{cases}$$



$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example.

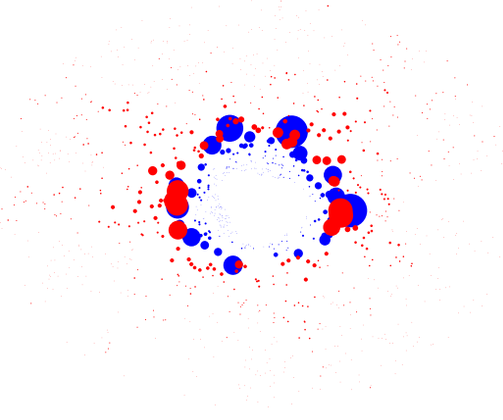
Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} = \frac{e^{-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)}}{L\Pi_{q=1}^{t} Z_q}$$

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & < 1, \quad y_i = h_t(x_i) \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & > 1, \quad y_i \neq h_t(x_i) \end{cases}$$



$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example.

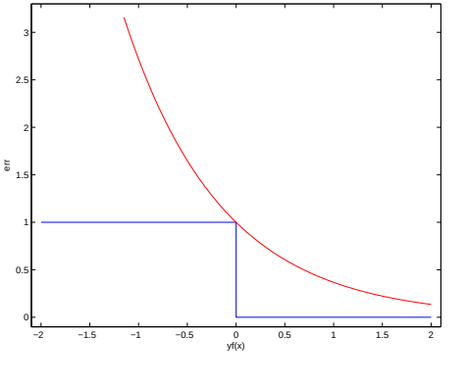
Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} = \frac{e^{-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)}}{L\Pi_{q=1}^{t} Z_q}$$

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & < 1, \quad y_i = h_t(x_i) \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & > 1, \quad y_i \neq h_t(x_i) \end{cases}$$



⇒ Increase (decrease) weight of wrongly (correctly) classified examples. The weight is the upper bound on the error of a given example.

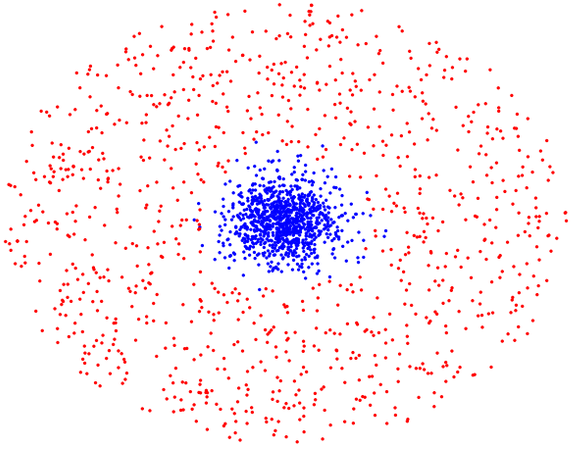
Summary of the Algorithm

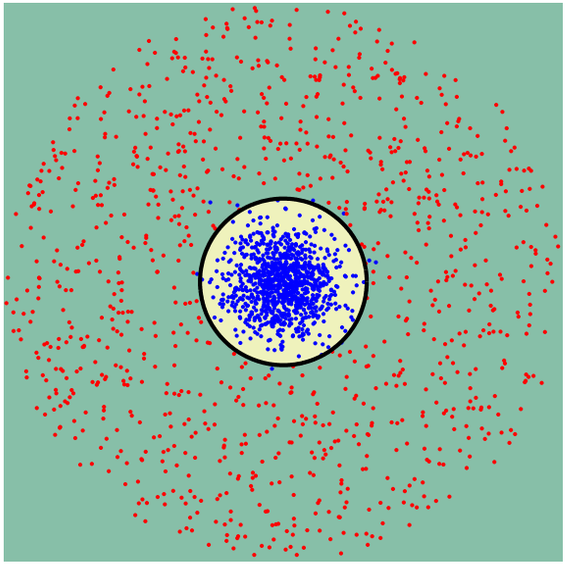
Initialization ...

Summary of the Algorithm

Initialization ...

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



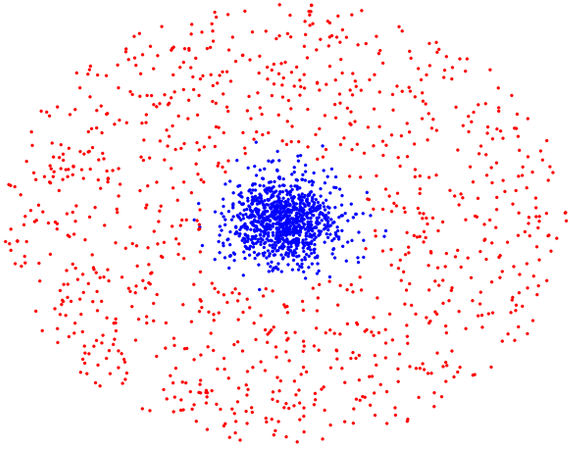


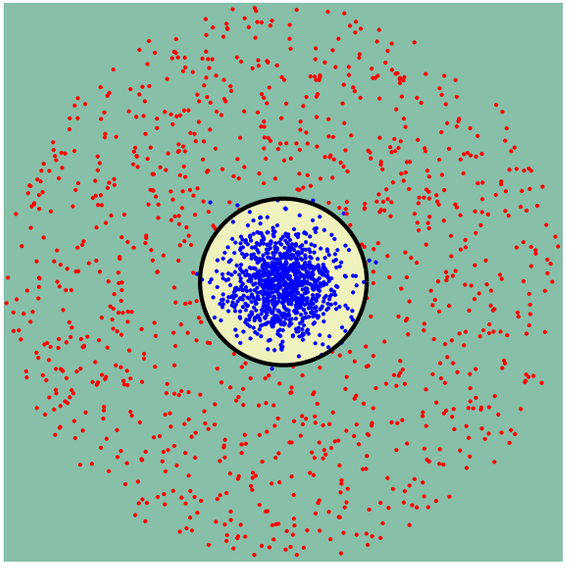
Summary of the Algorithm

Initialization ...

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

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



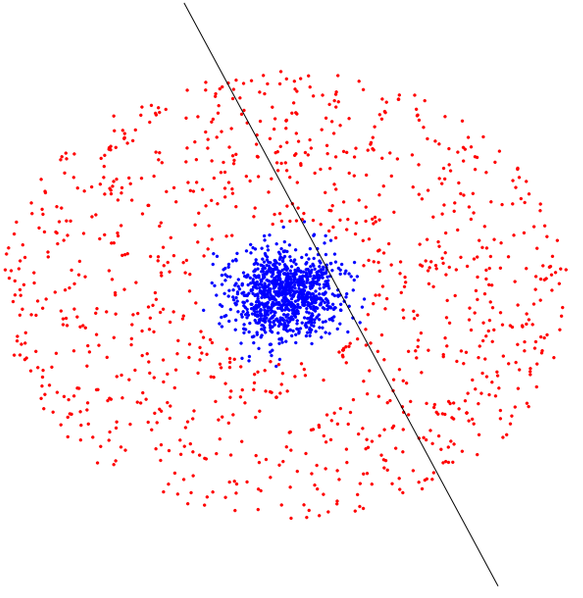


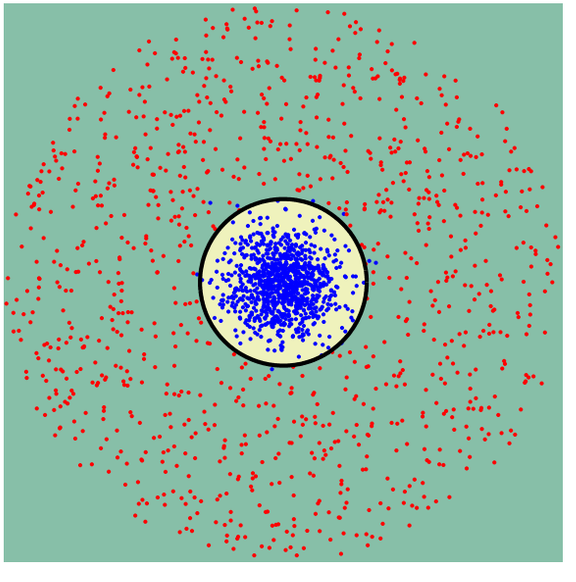
Summary of the Algorithm

Initialization ... $t = 1$

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

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



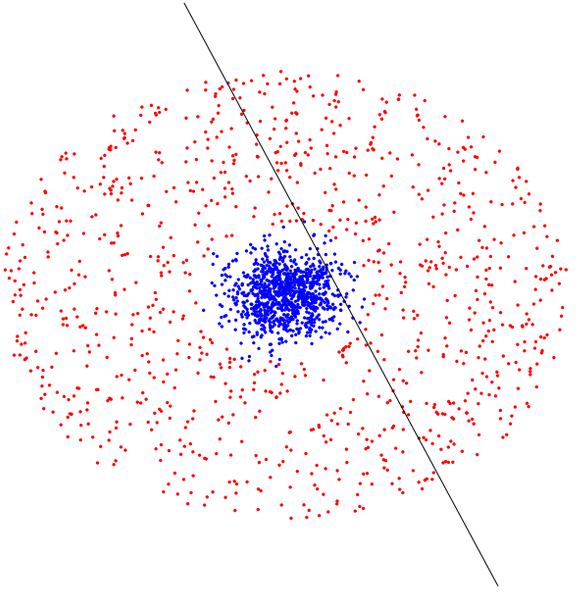


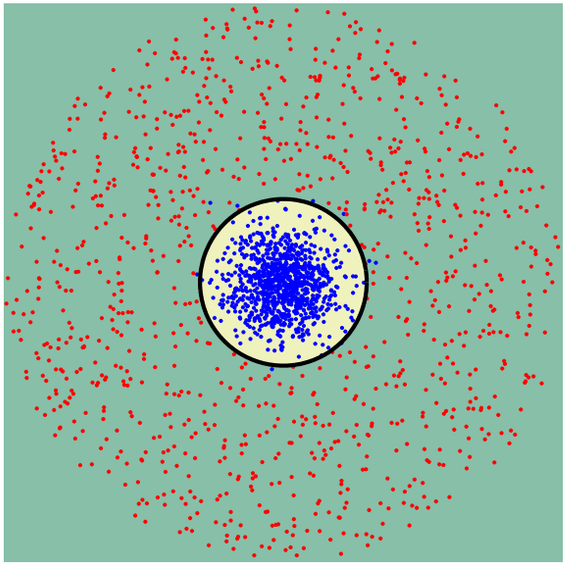
Summary of the Algorithm

Initialization ... $t = 1$

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

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

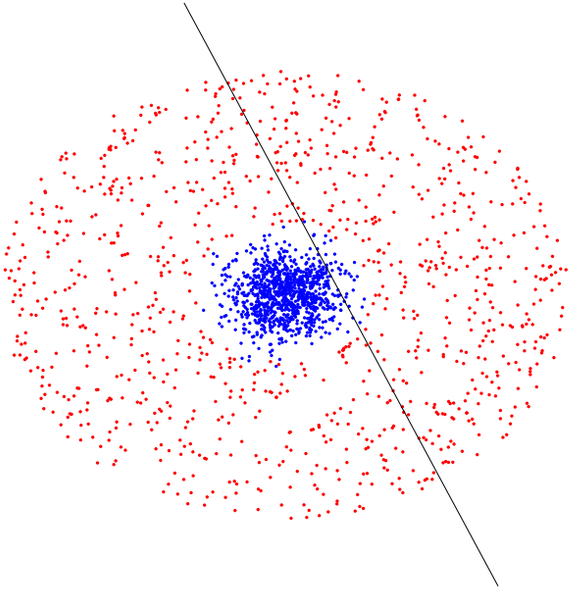




Summary of the Algorithm

Initialization ...

$t = 1$

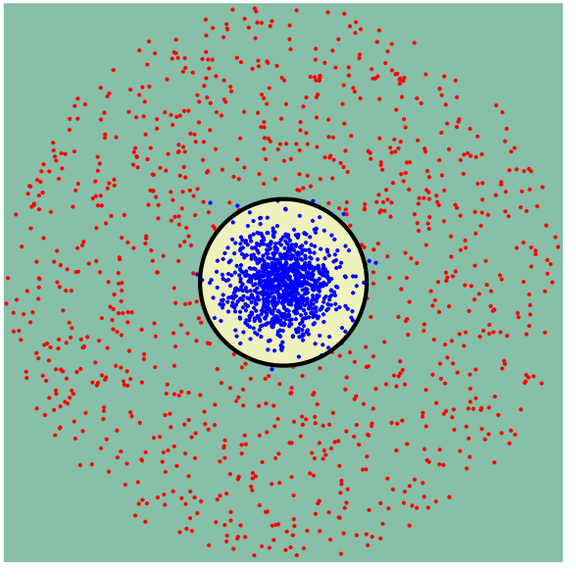


For $t = 1, ..., T$:

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

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

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



Summary of the Algorithm

Initialization ... $t = 1$

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

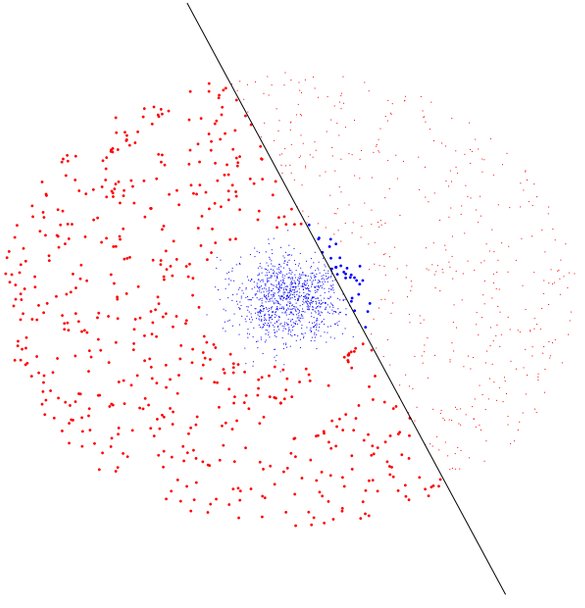
- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t;\ \epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

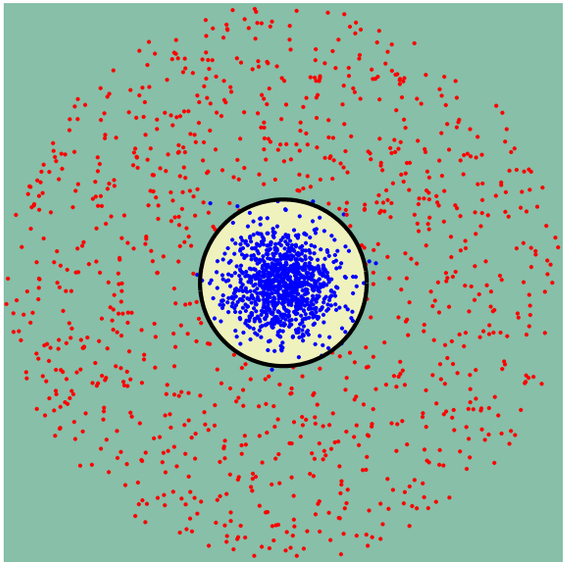
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

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

Output the final classifier:

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





Comments

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

Summary of the Algorithm

Initialization ... $t = 1$

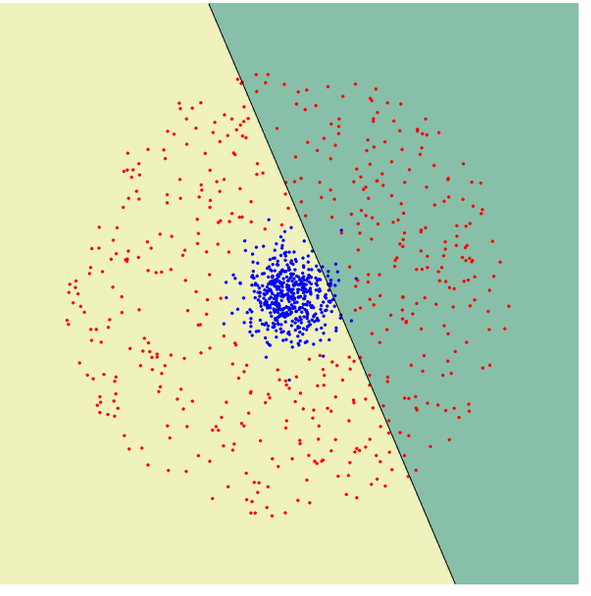
For $t = 1, ..., T$:

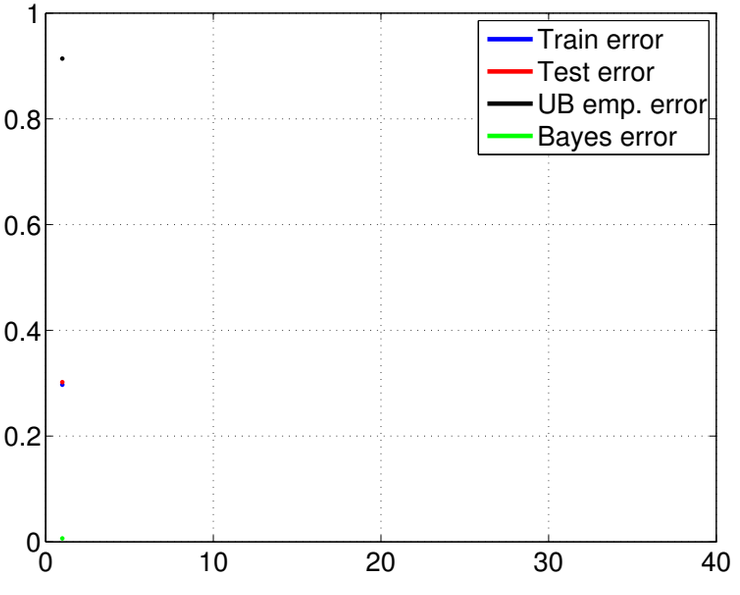
- Find $h_t = \arg\min_{h \in \mathcal{B}} \epsilon_t;\ \epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update $$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

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

Output the final classifier:

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





Comments

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

Summary of the Algorithm

Initialization ...

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

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

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

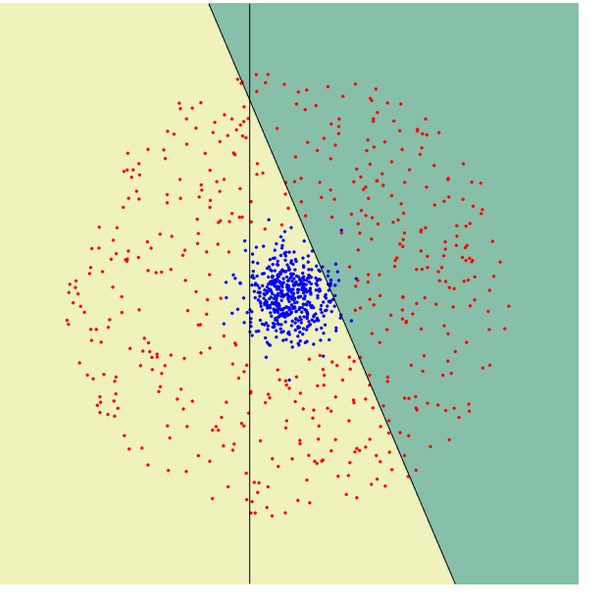
Output the final classifier:

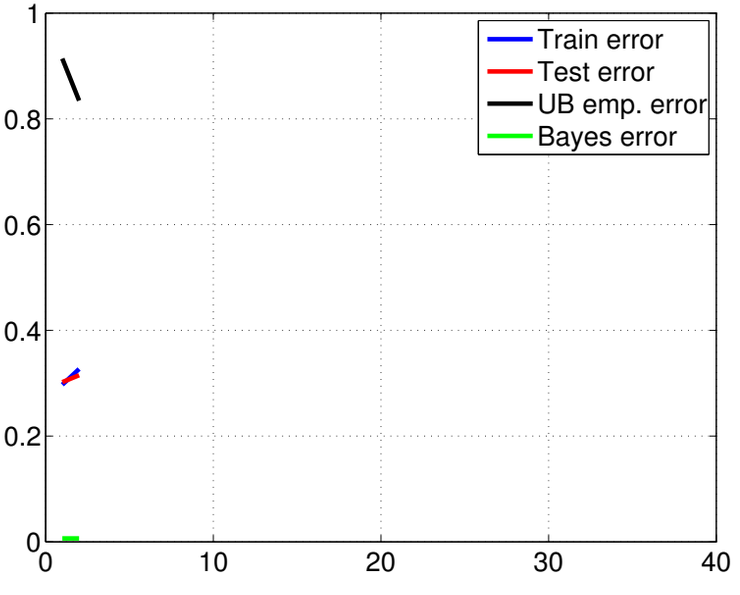
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Comments

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

$t = 2$





Summary of the Algorithm

Initialization ...

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

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

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

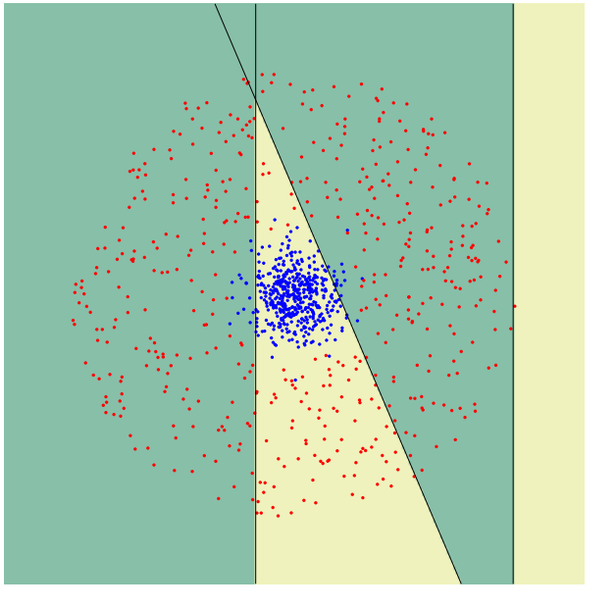
Output the final classifier:

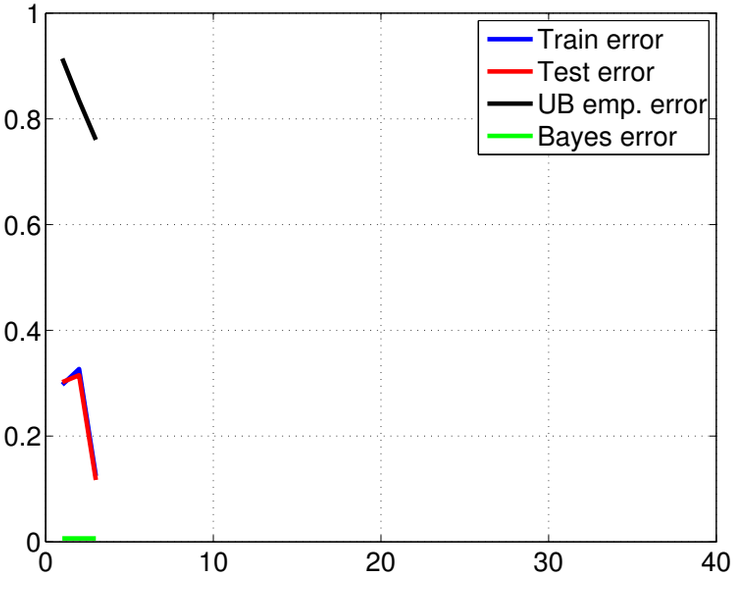
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Comments

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

$t = 3$





Summary of the Algorithm

Initialization ...

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

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

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

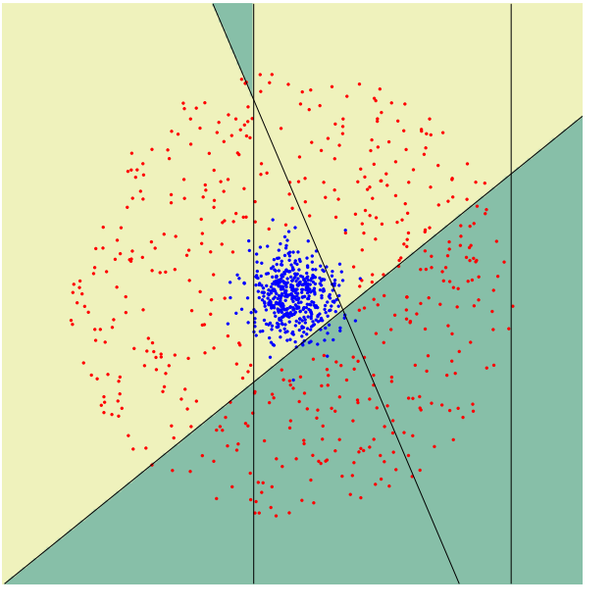
Output the final classifier:

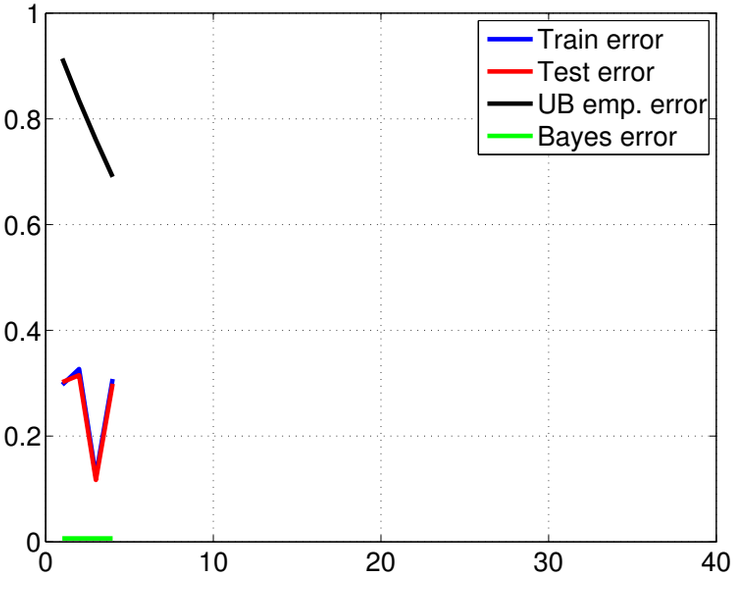
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

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

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

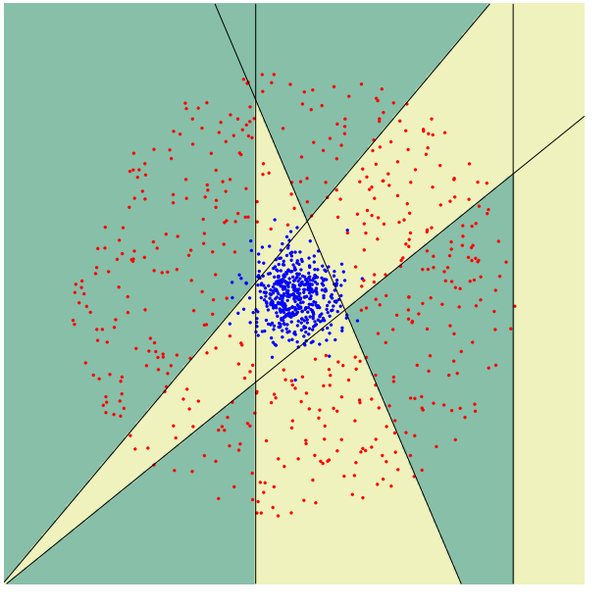
Output the final classifier:

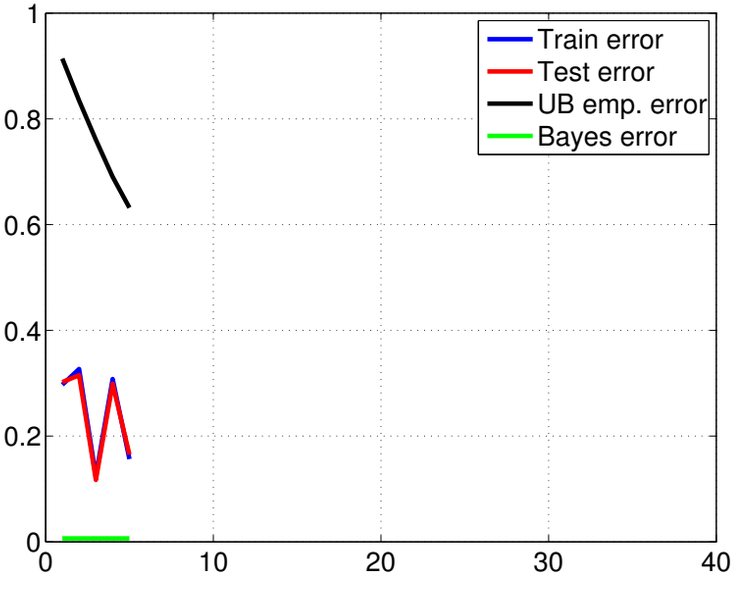
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

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

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

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

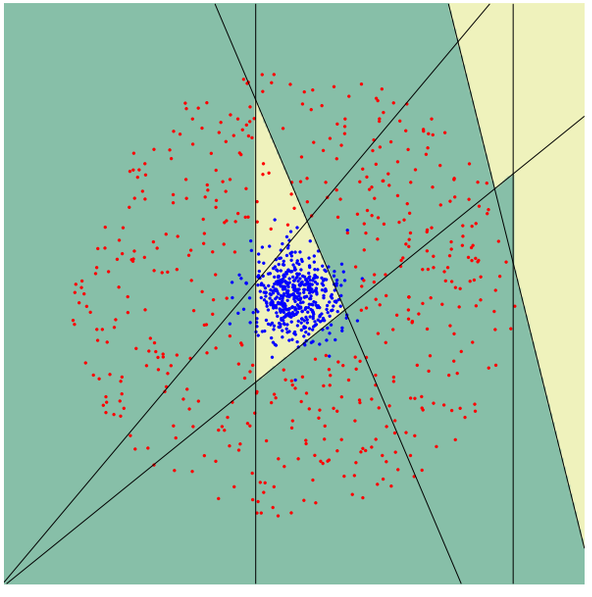
Output the final classifier:

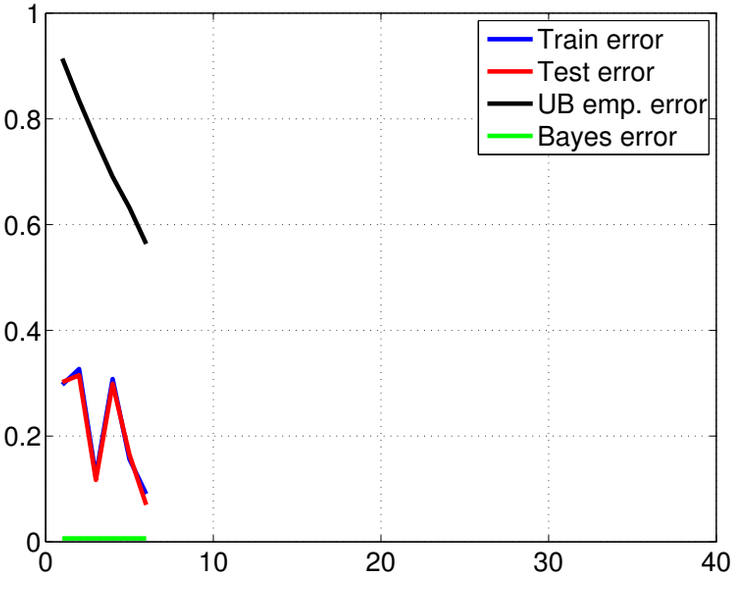
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Comments

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

$t = 6$





Summary of the Algorithm

Initialization ...

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

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

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

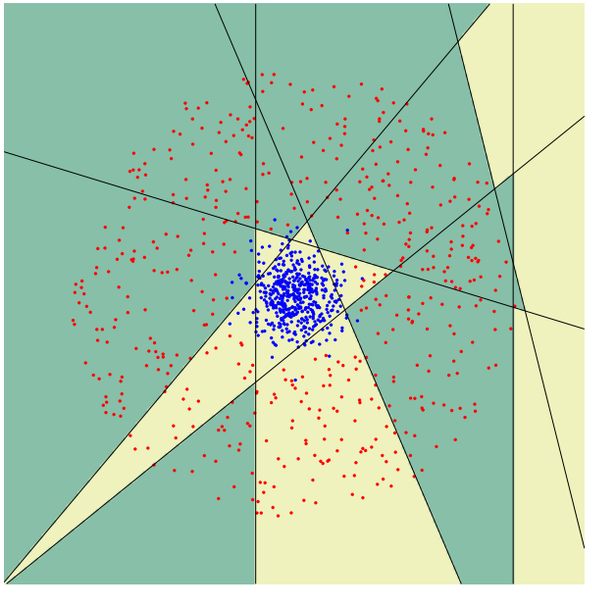
Output the final classifier:

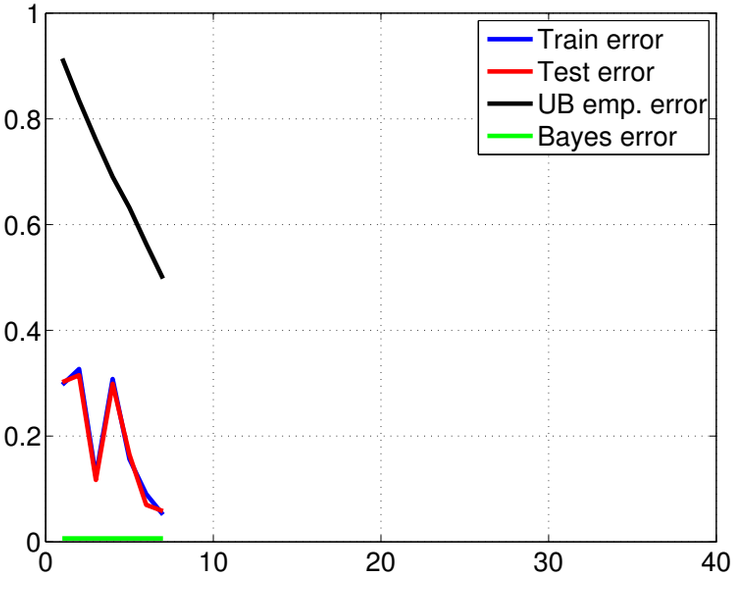
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Comments

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

$t = 7$





Summary of the Algorithm

Initialization ...

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

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

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

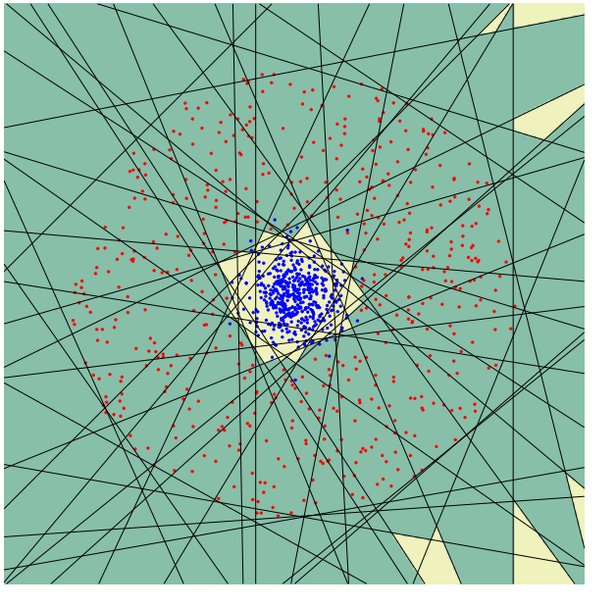
Output the final classifier:

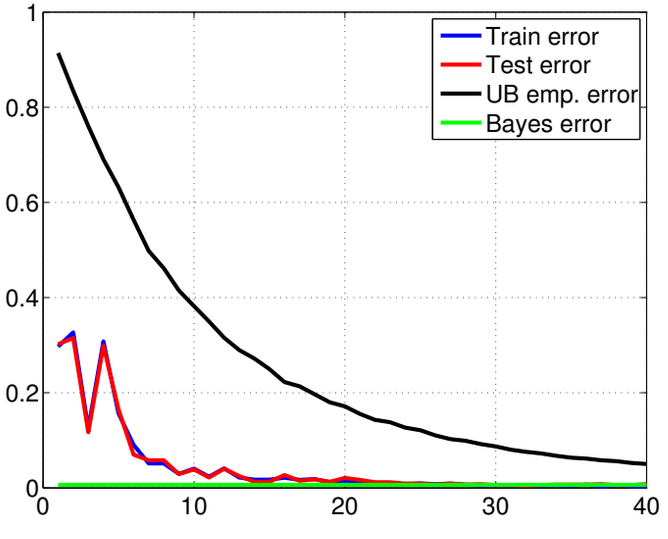
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Comments

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

$t = 40$





100 steps  detail 

Does AdaBoost generalise?

Margins in SVM

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

Margins in AdaBoost

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

Maximising margins in AdaBoost

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

Upper bounds based on margin

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

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.
- a 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

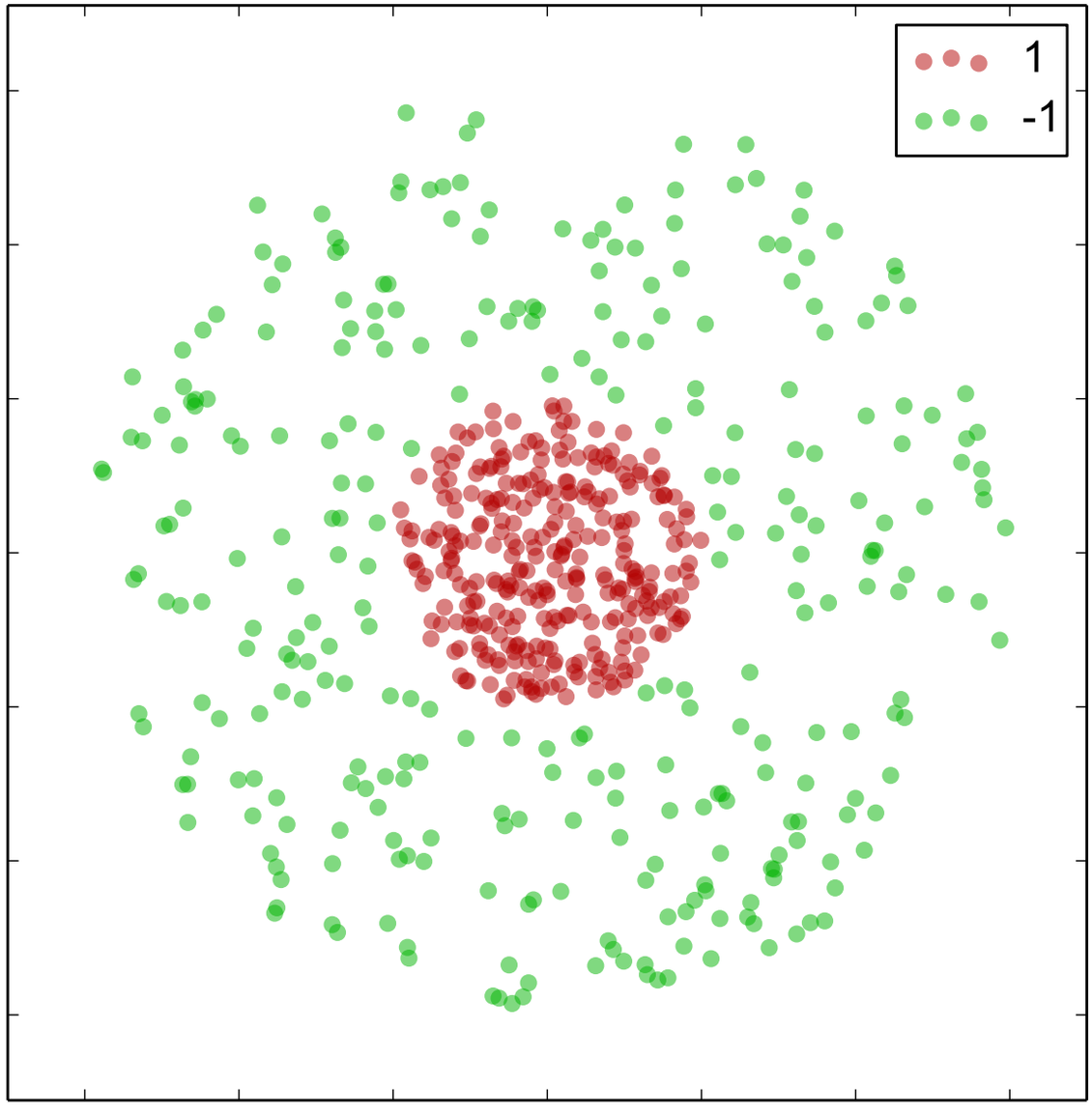
- Discrete ($h : \mathcal{X} \to \{0, 1\}$)
- Multiclass AdaBoost.M1 ($h : \mathcal{X} \to \{0, 1, ..., k\}$)
- Multiclass AdaBoost.M2 ($h : \mathcal{X} \to [0, 1]^k$)
- Real valued AdaBoost.R ($Y = [0, 1]$, $h : \mathcal{X} \to [0, 1]$)

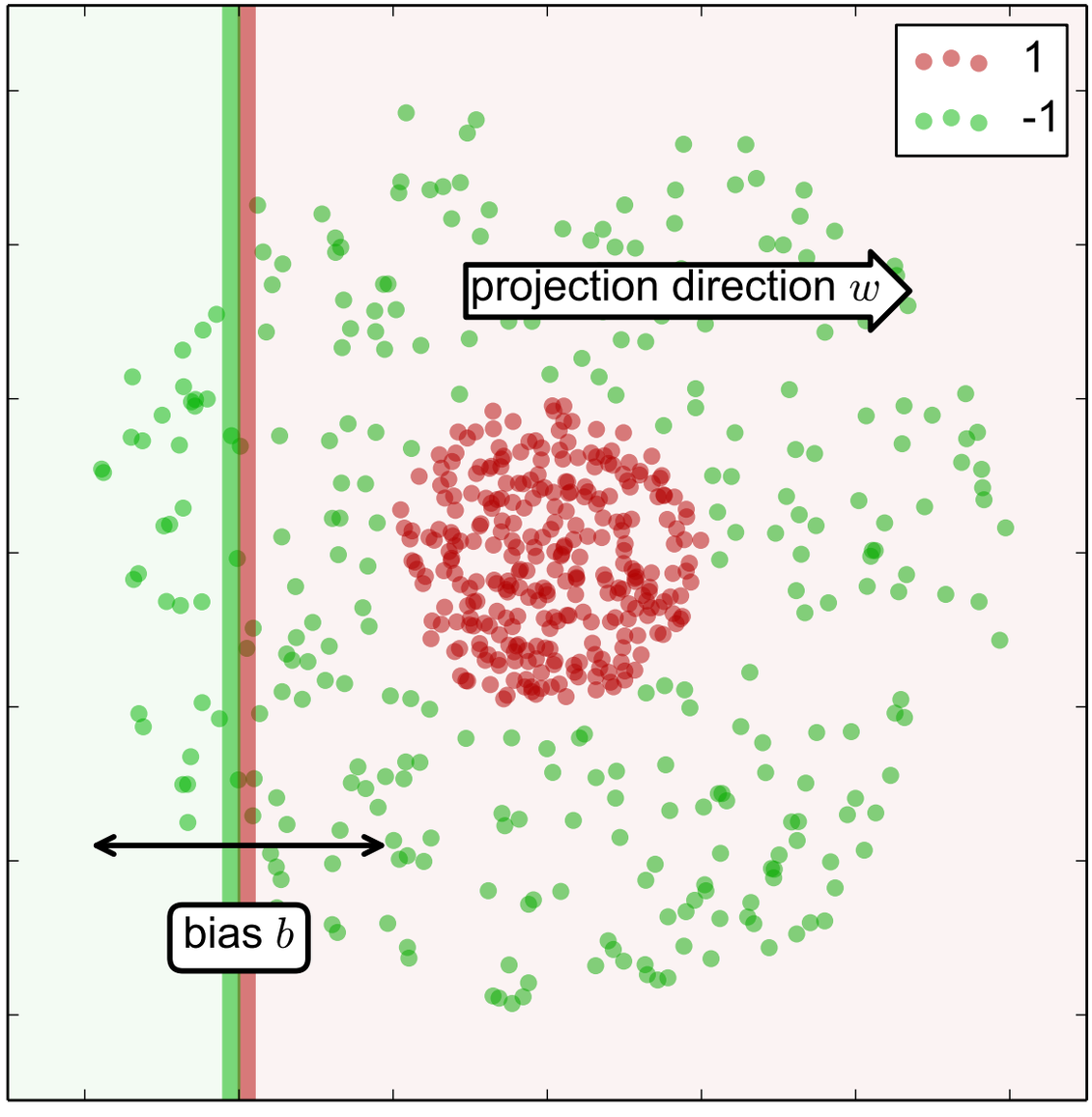
Schapire & Singer 1997

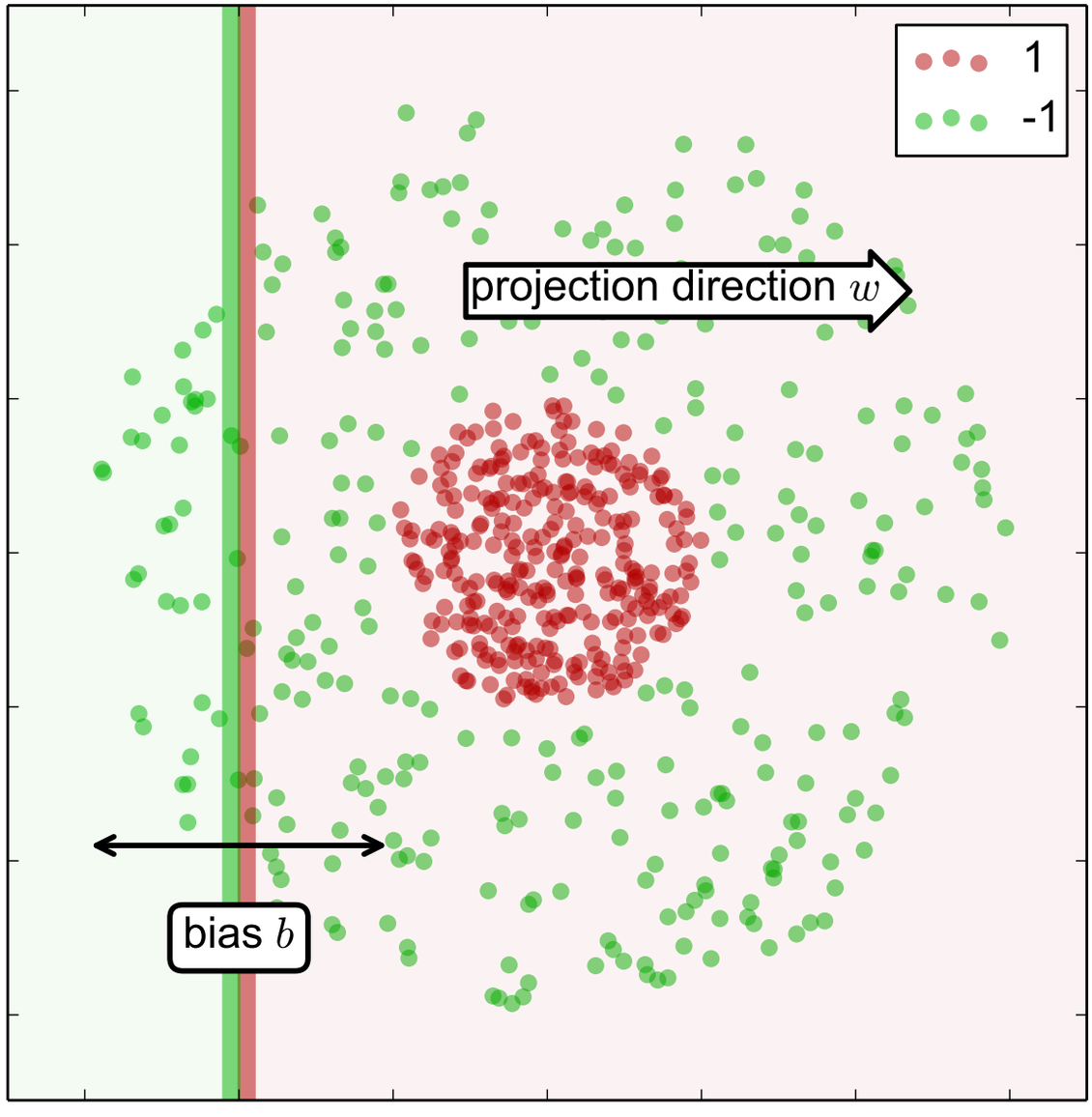
- Confidence rated prediction ($h : \mathcal{X} \to R$, two-class)
- Multilabel AdaBoost.MR, AdaBoost.MH (different formulation of minimised loss)

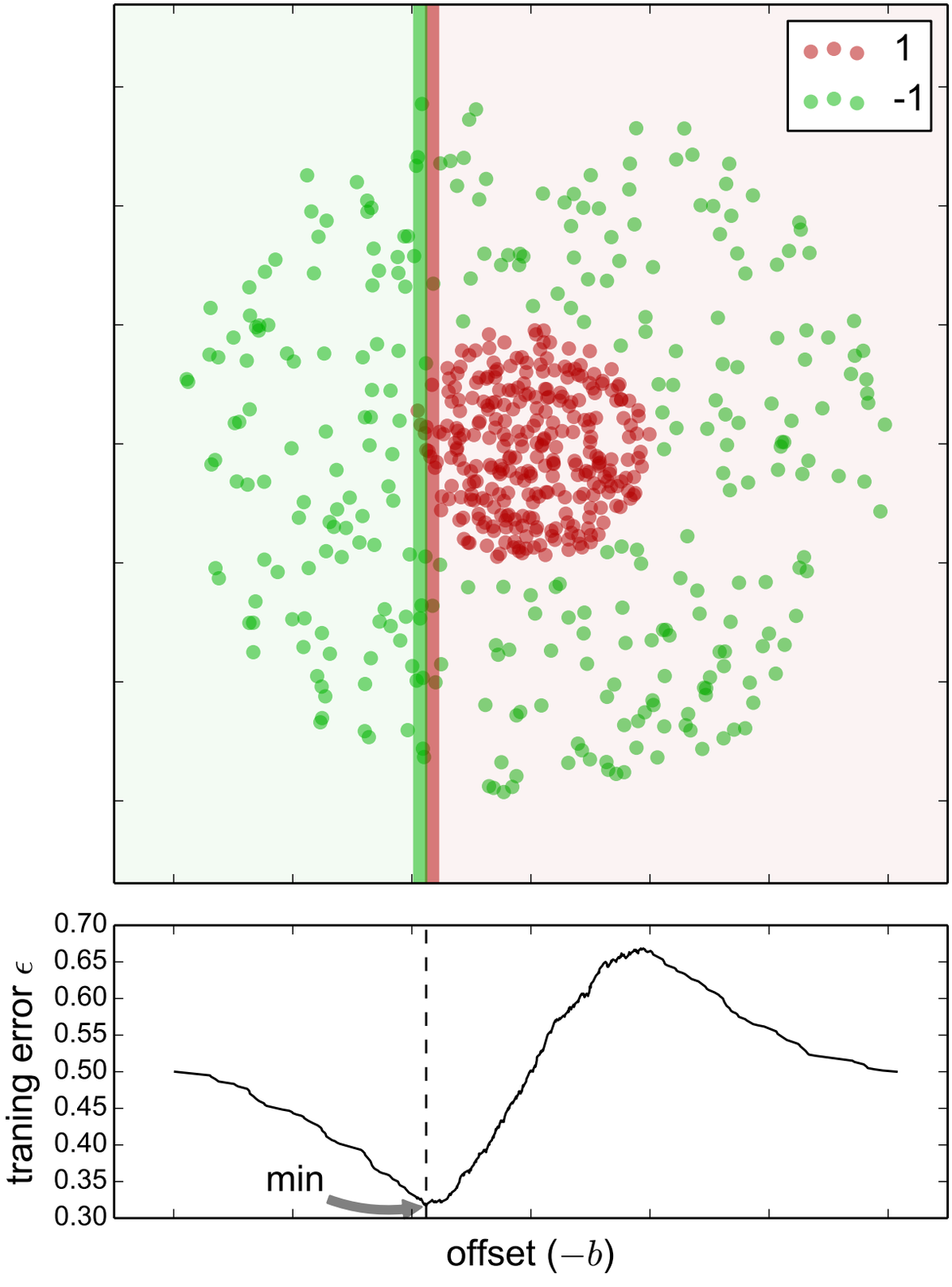
... Many other modifications since then (WaldBoost, cascaded AB, online AB, ...)

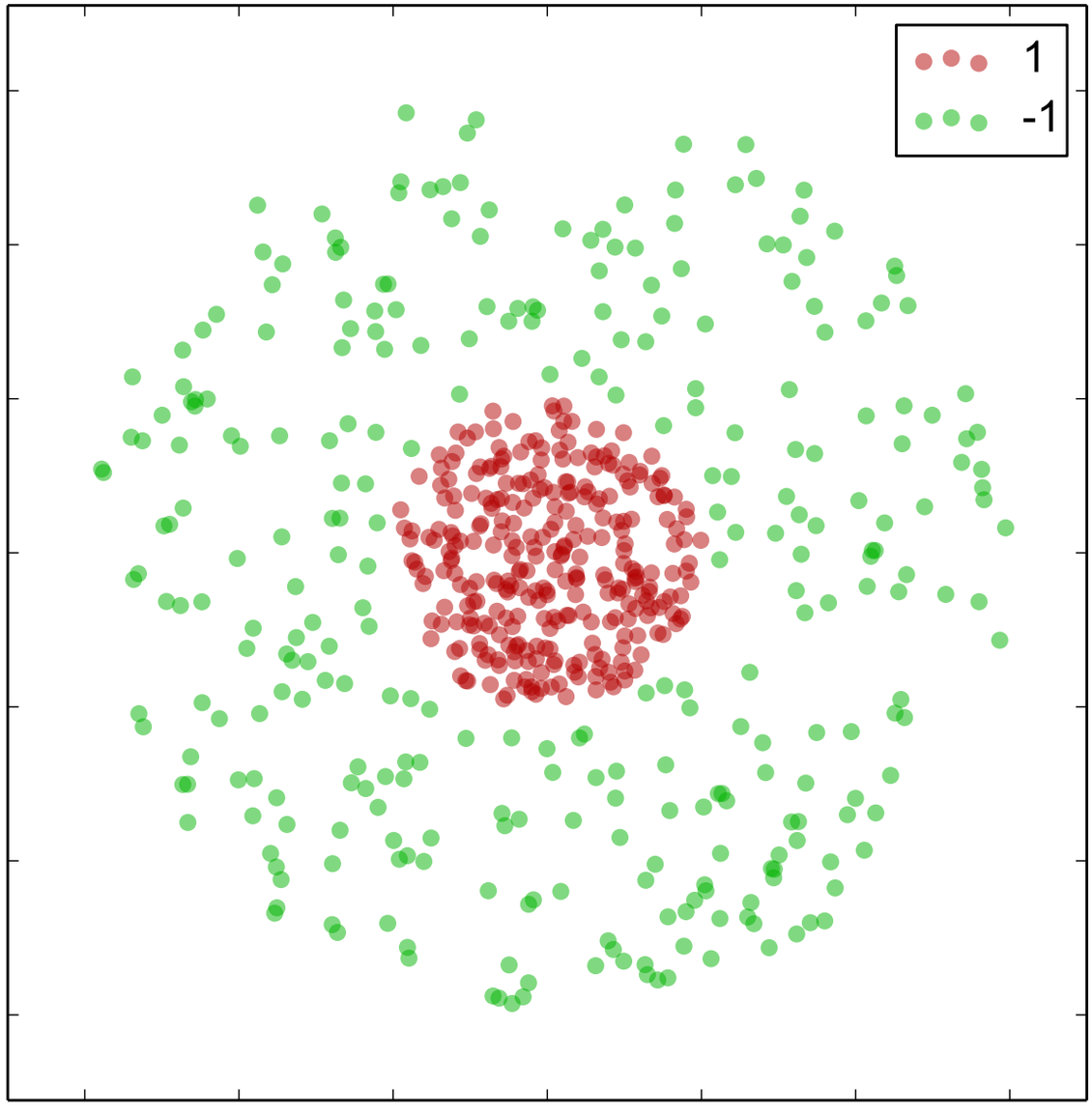

m p

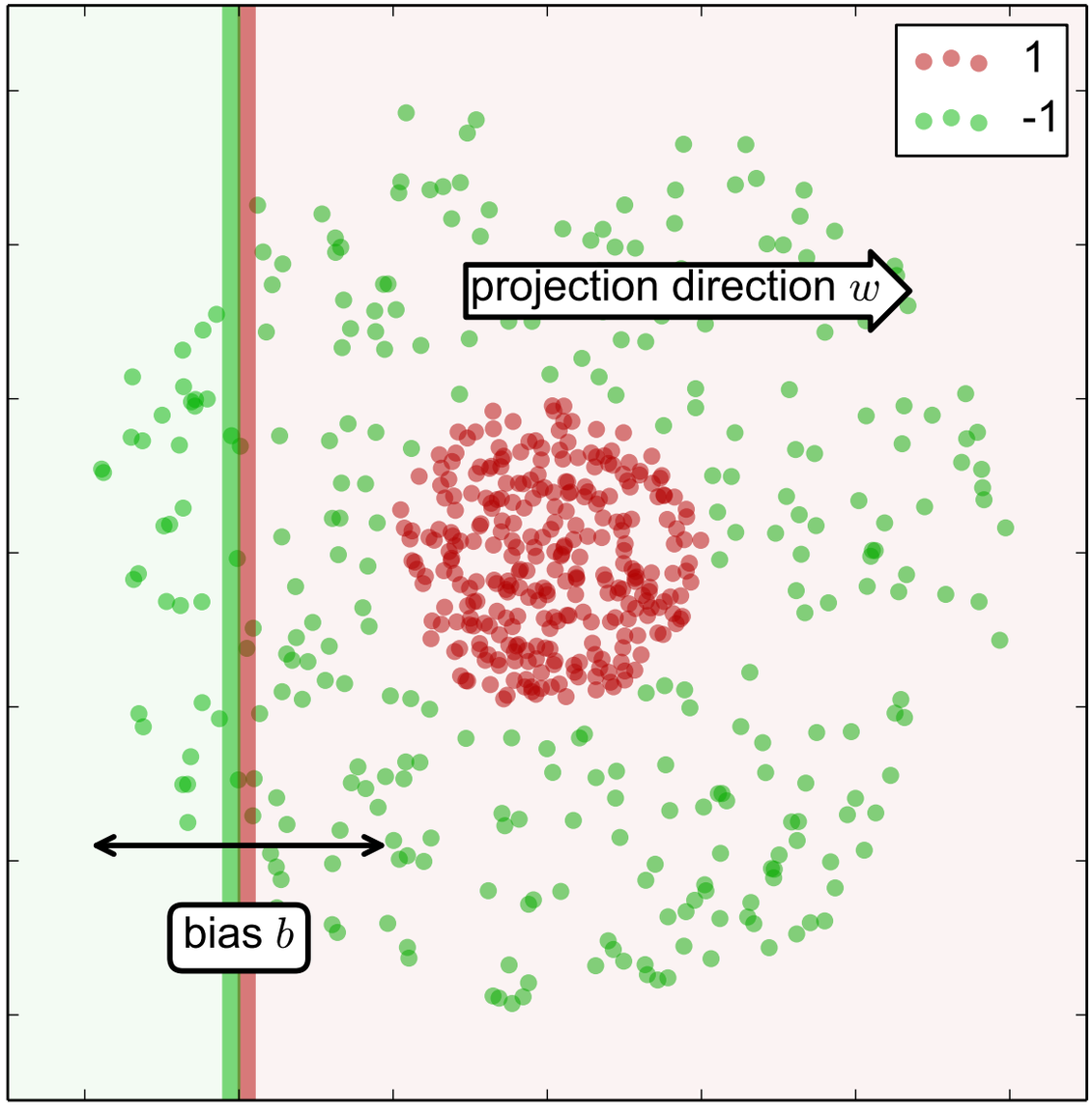

1
-1

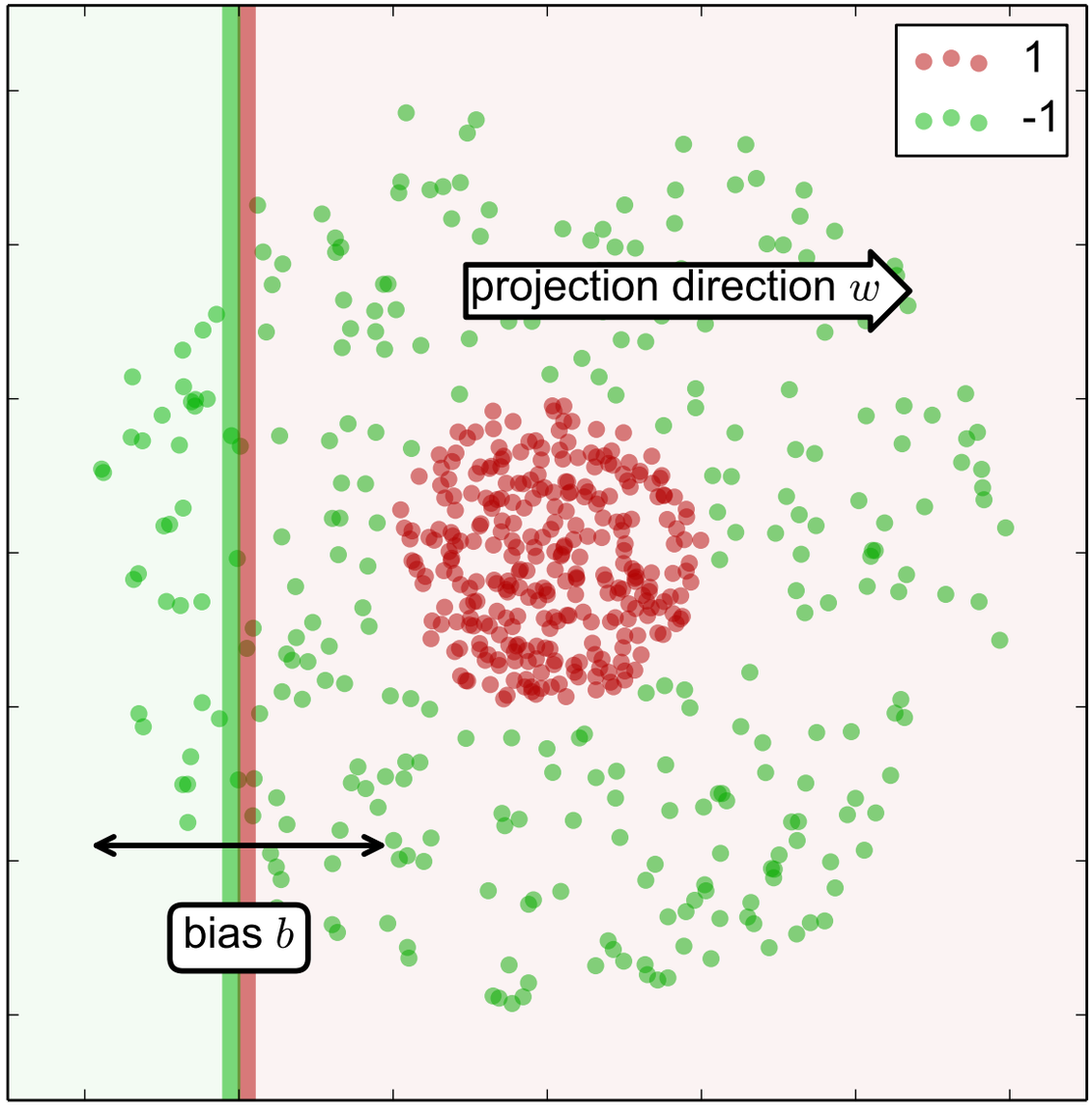

1
-1
projection direction w
bias b

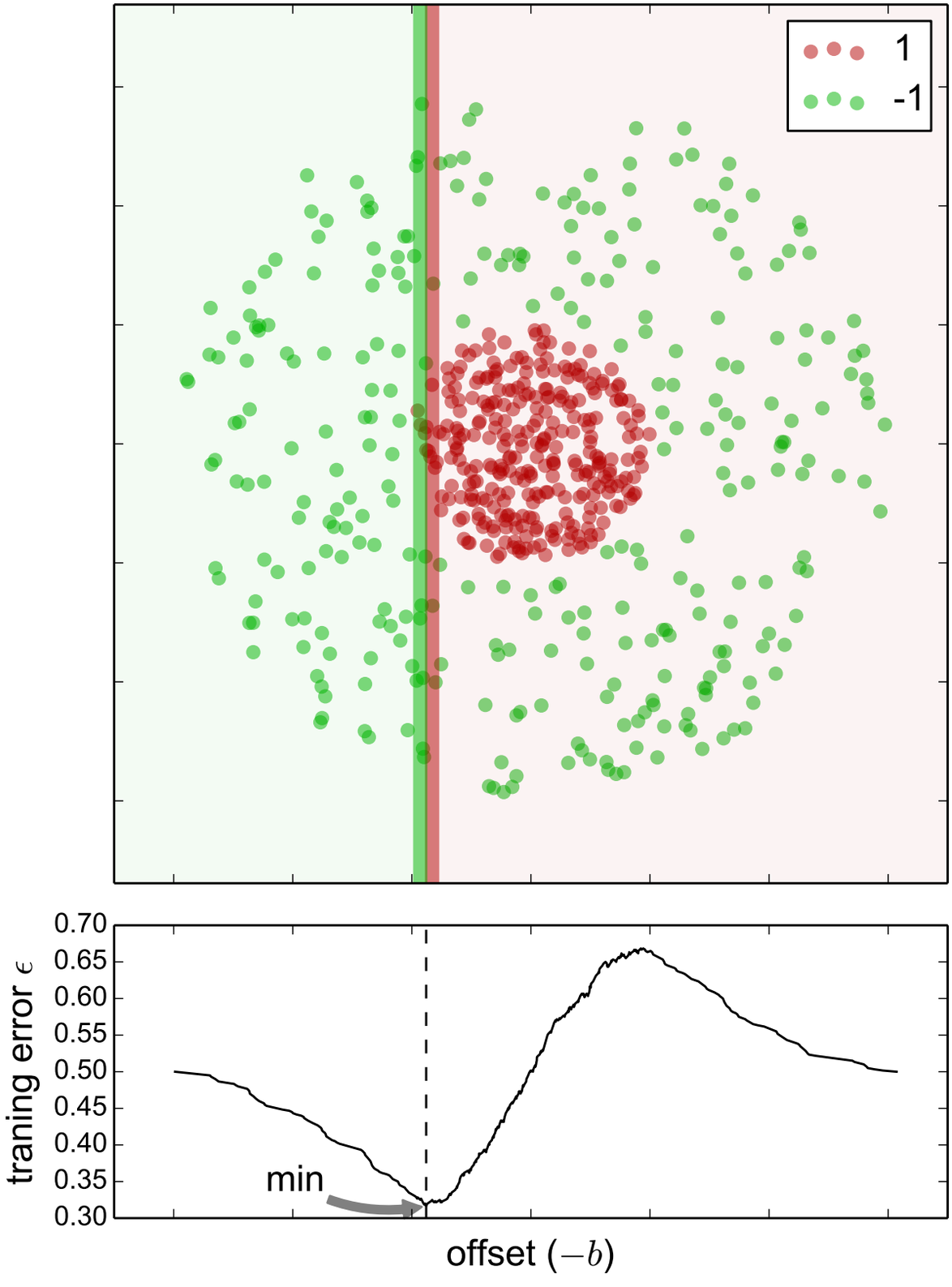

1
-1
projection direction w
bias b

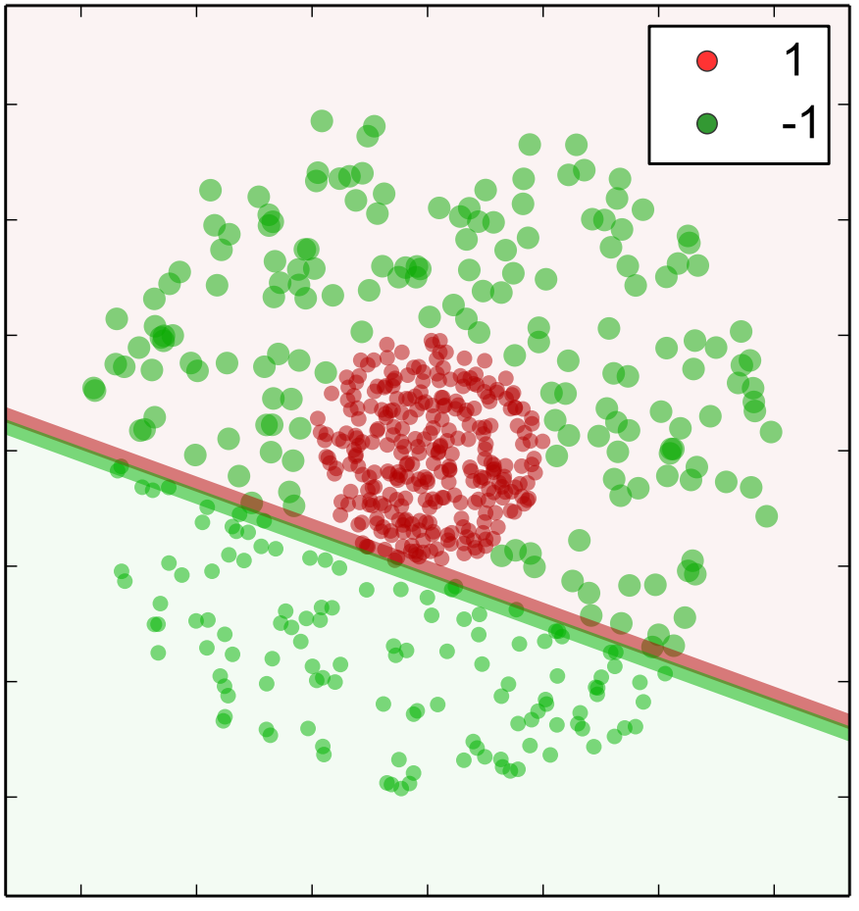

1
-1
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
traning error ϵ
min
offset $(-b)$


1
-1

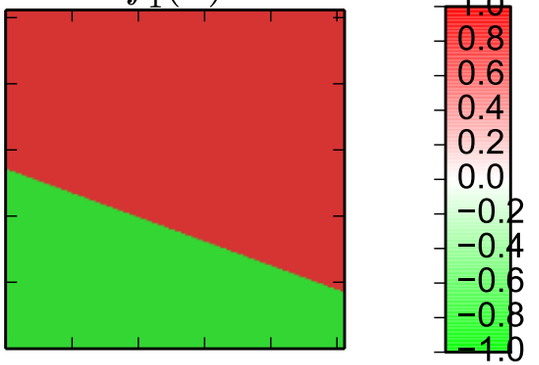

1
-1
projection direction w
bias b


1
-1
projection direction w
bias b

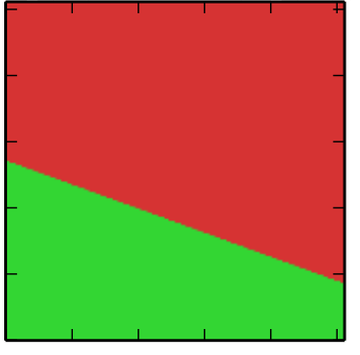

1
-1
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
traning error ϵ
min
offset $(-b)$



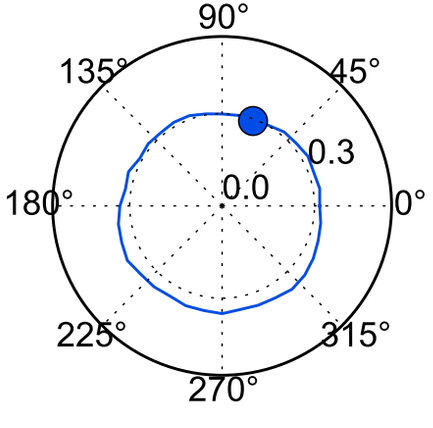
$f_1'(x)$

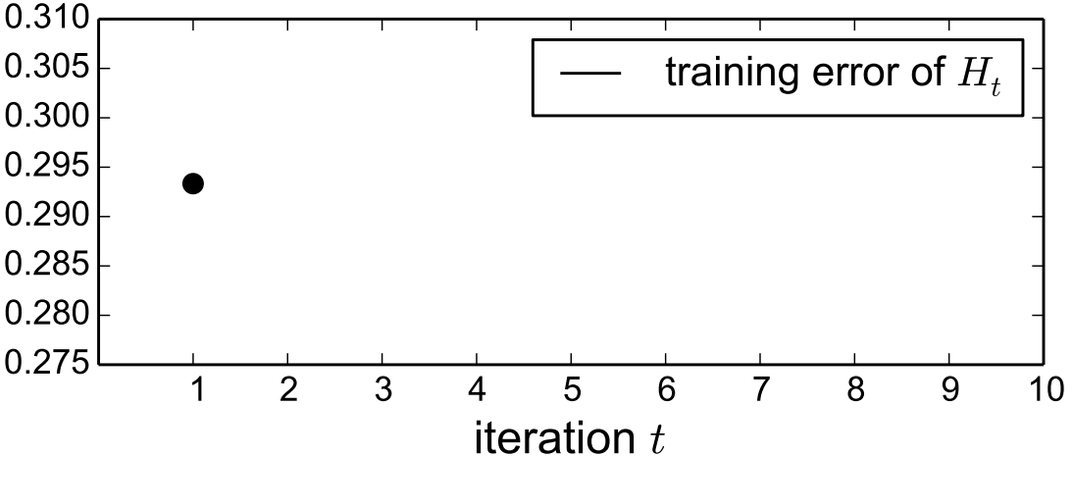


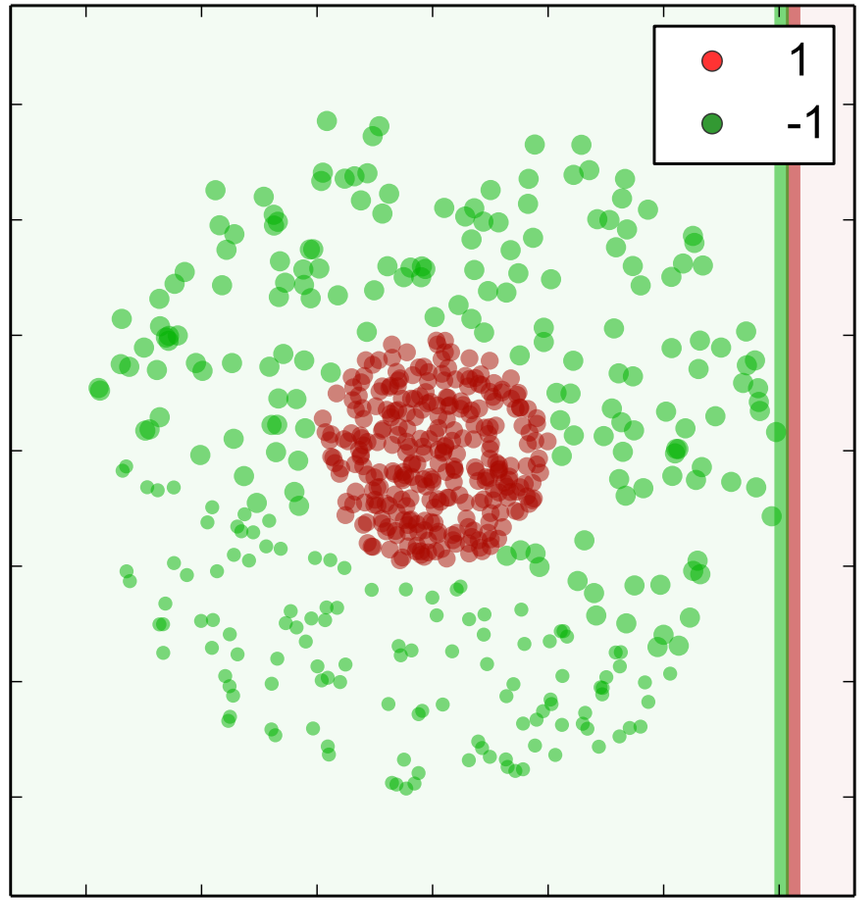
$H_1(x) = \text{sign}(f_1(x))$



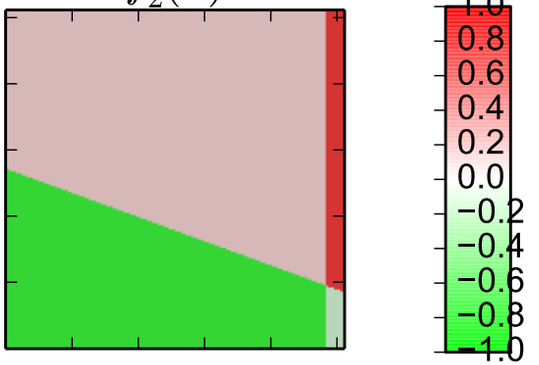
$\epsilon_1(w)$
at optimal b



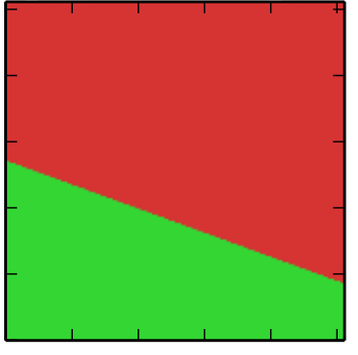




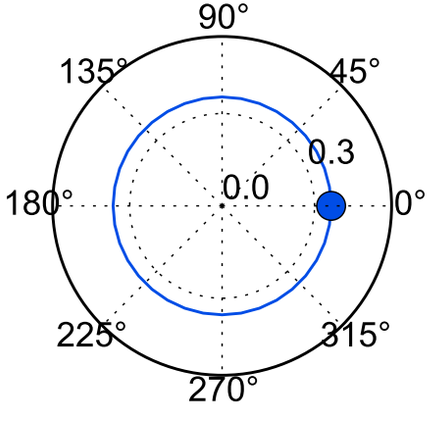
$f_2'(x)$

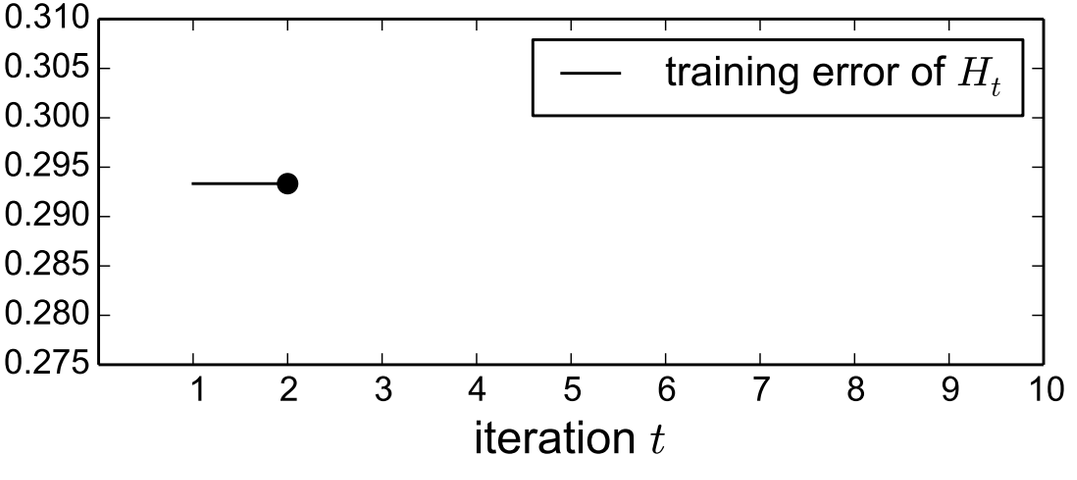


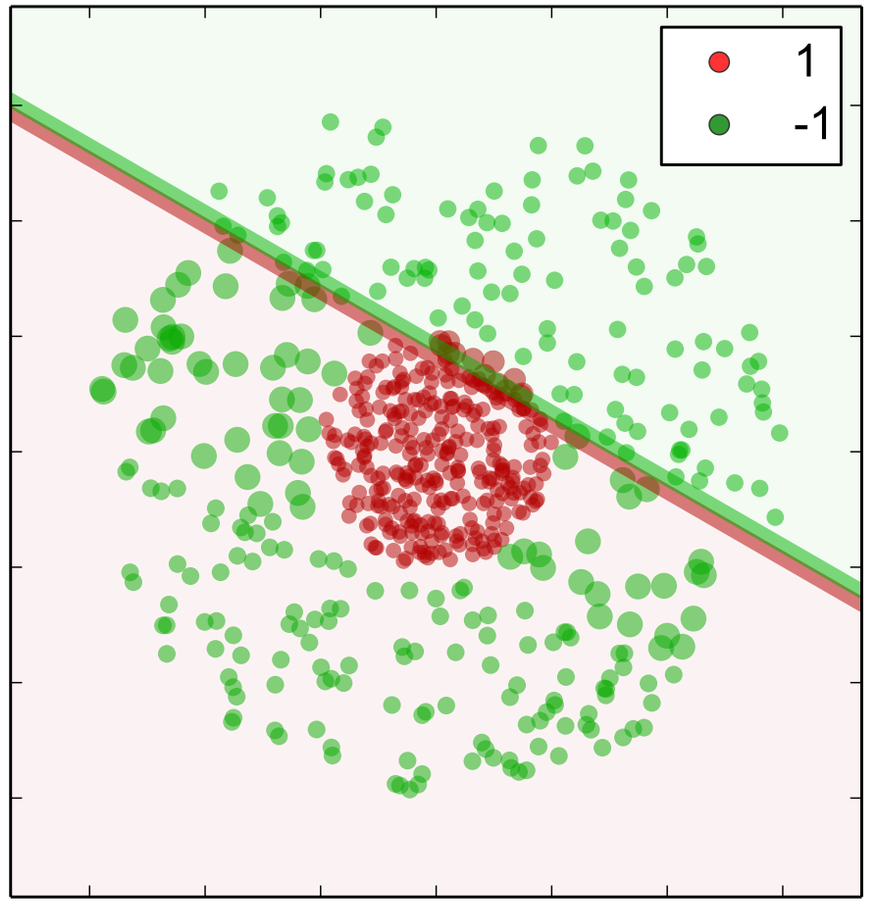
$H_2(x) = \text{sign}(f_2(x))$

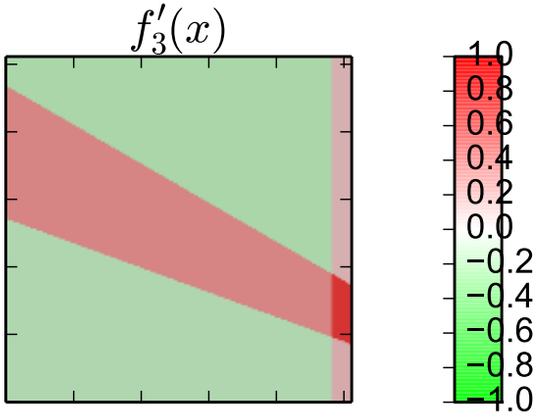


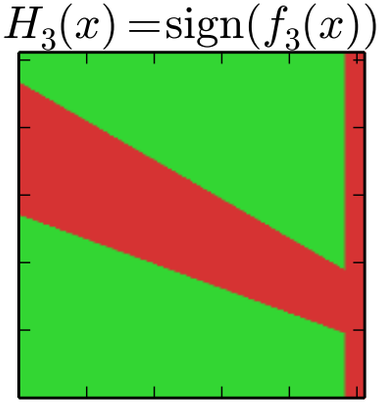
$\epsilon_2(w)$
at optimal b

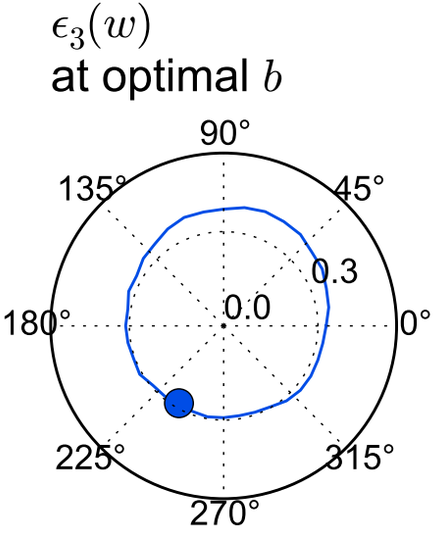


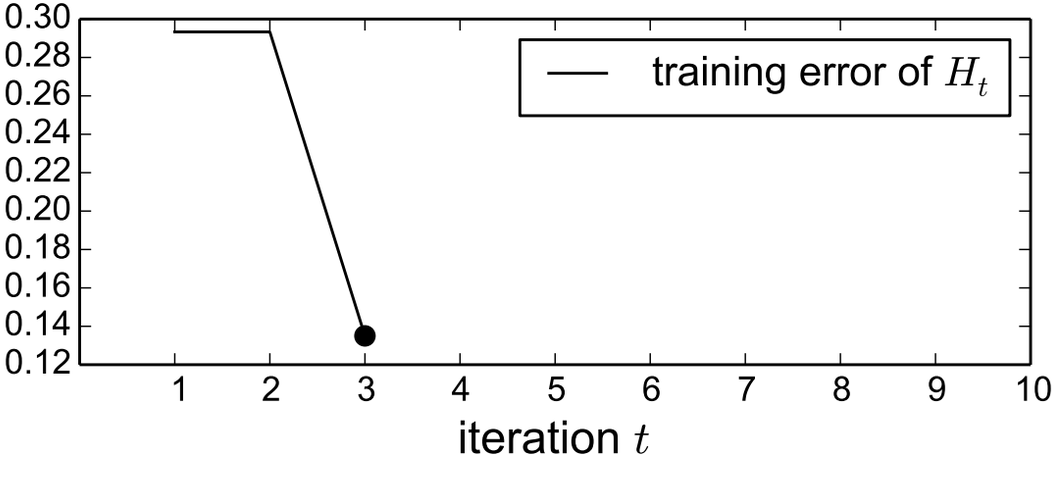


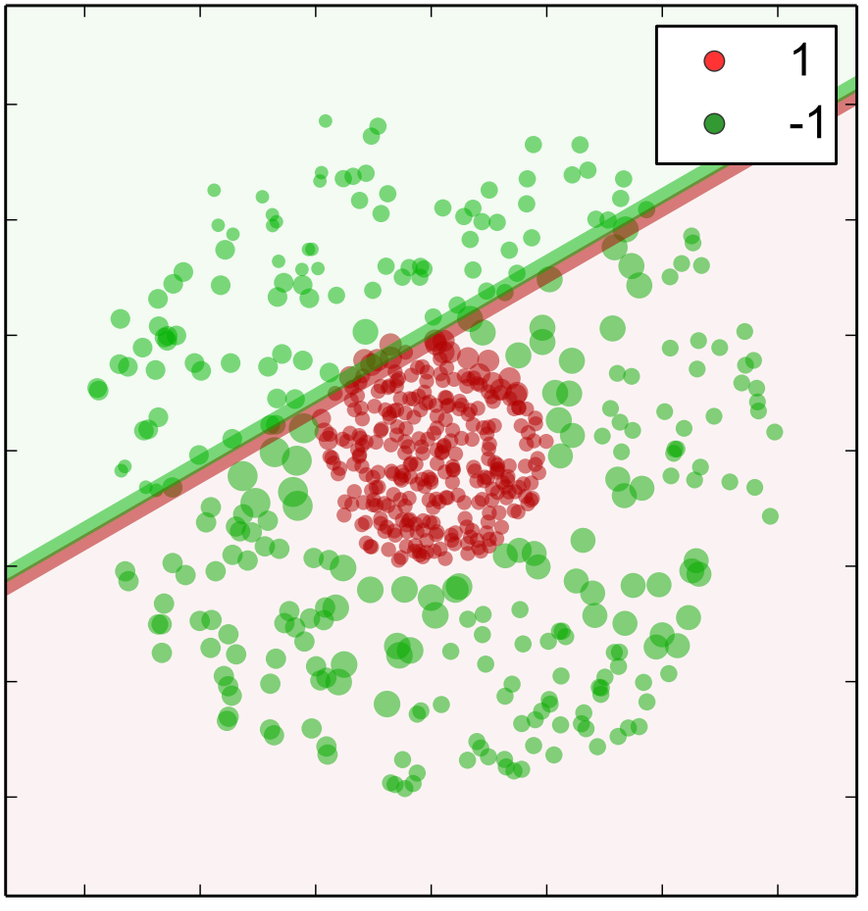

1
-1


$f_3'(x)$
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0

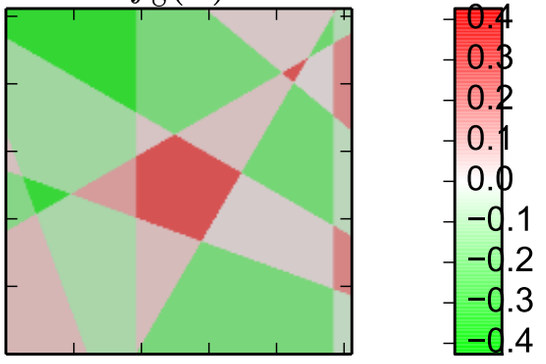

$H_3(x) = \text{sign}(f_3(x))$


$\epsilon_3(w)$
at optimal b
90°
135°
45°
180°
0°
225°
315°
270°
0.0
0.3

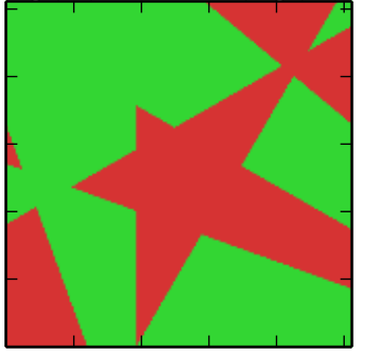

0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.14
0.12
1
2
3
4
5
6
7
8
9
10
training error of H_t
iteration t



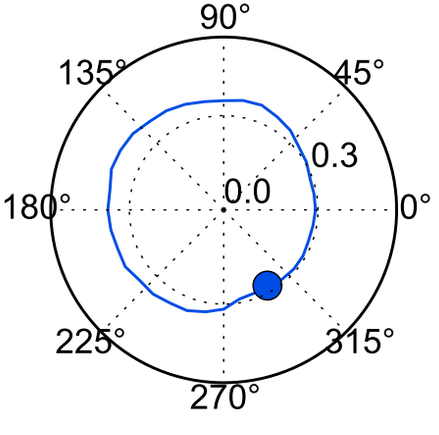
$f_8'(x)$

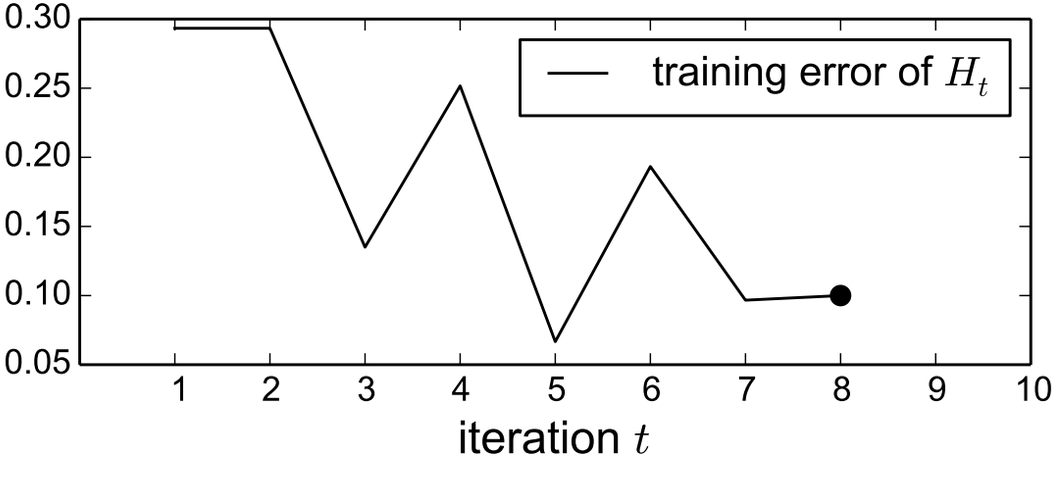


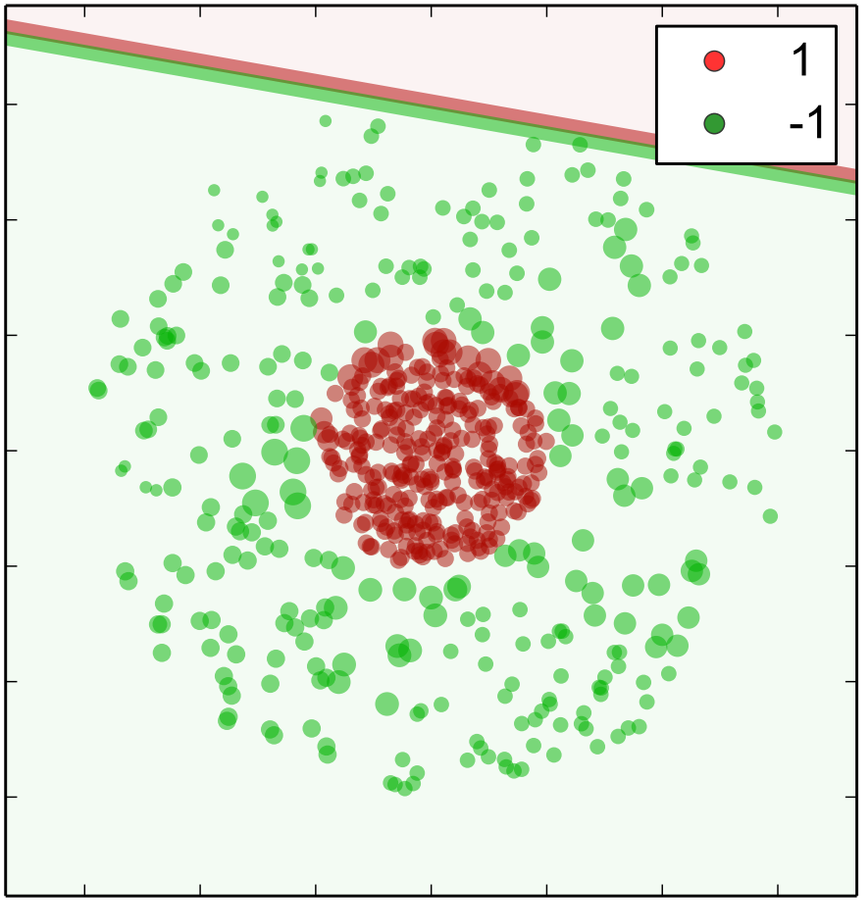
$H_8(x) = \text{sign}(f_8(x))$



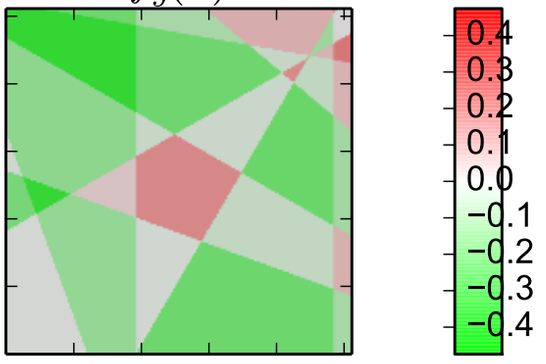
$\epsilon_8(w)$
at optimal b



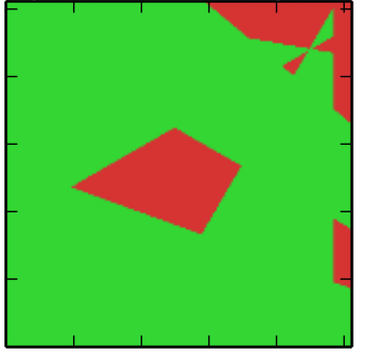




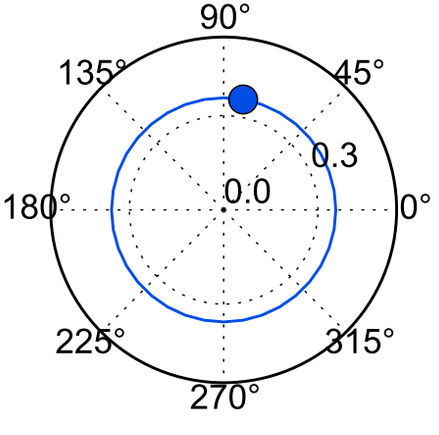
$f'_9(x)$

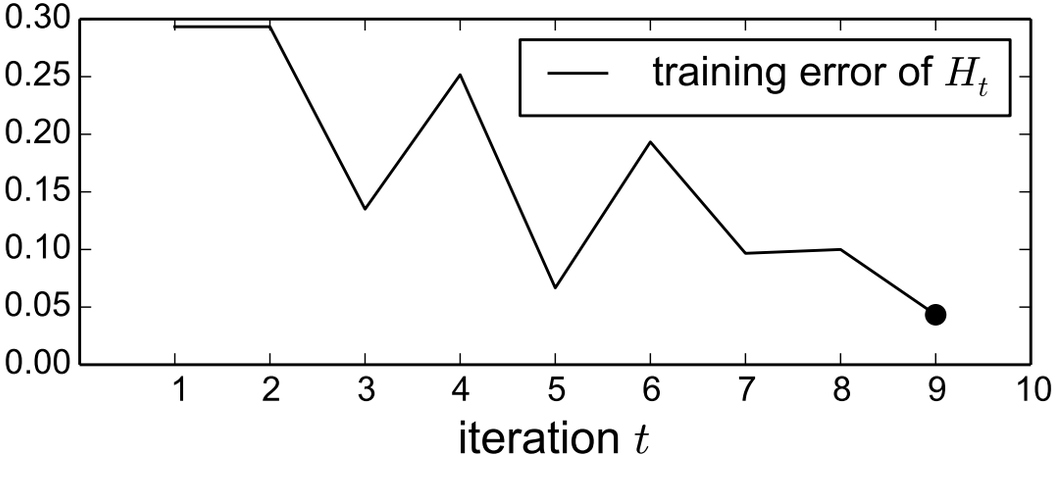


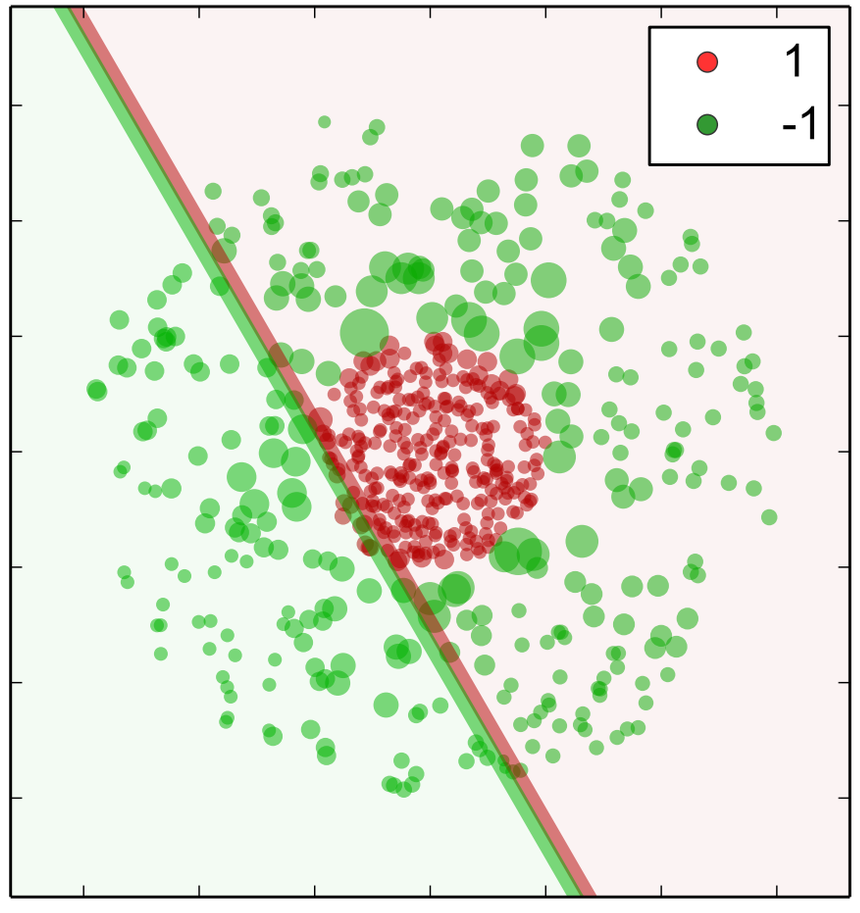
$H_9(x) = \text{sign}(f_9(x))$



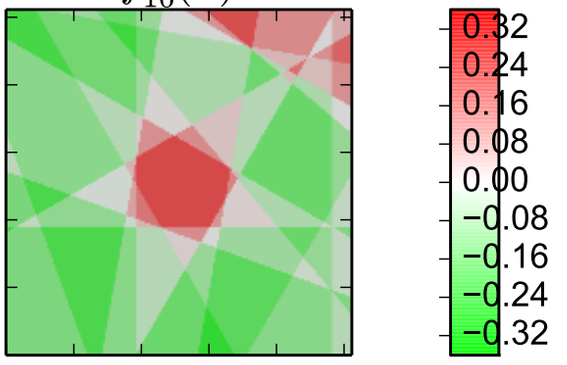
$\epsilon_9(w)$
at optimal b



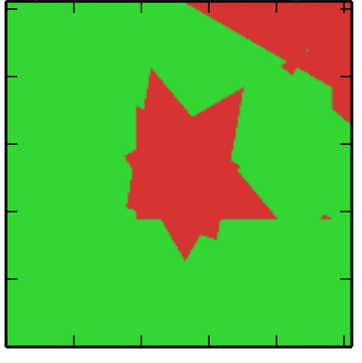




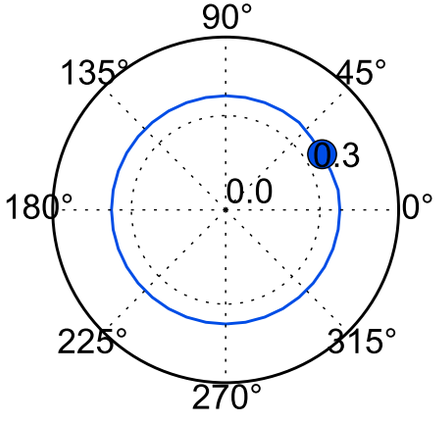
$f'_{16}(x)$

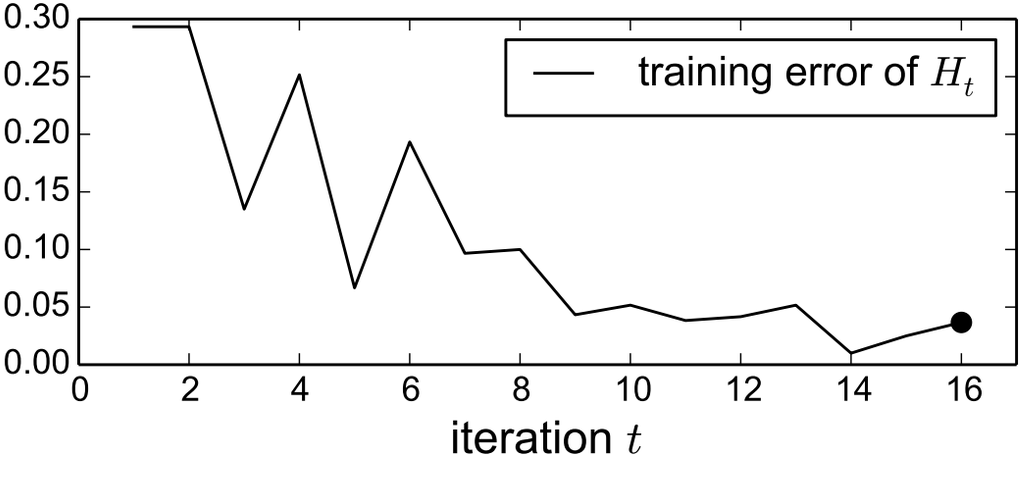


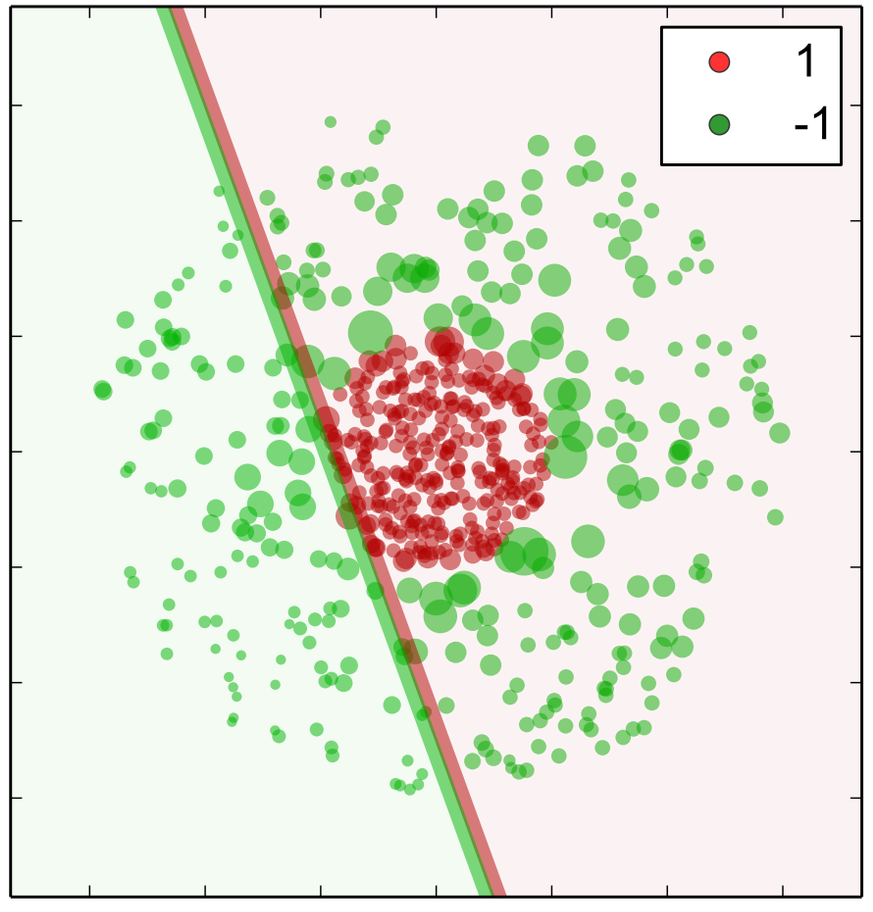
$H_{16}(x) = \text{sign}(f_{16}(x))$



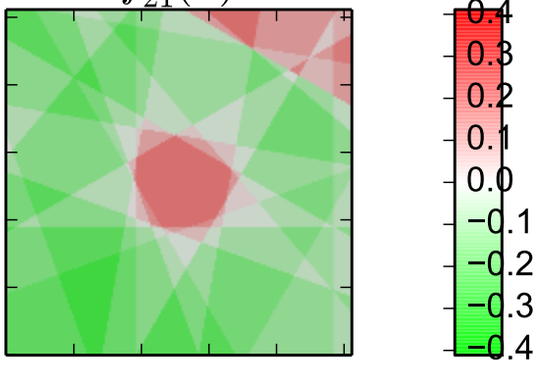
$\epsilon_{16}(w)$
at optimal b



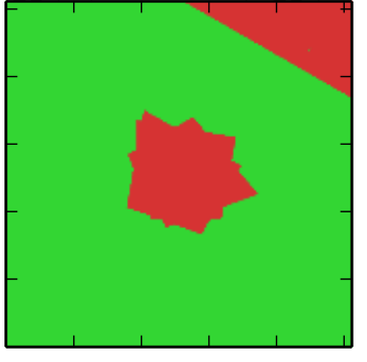




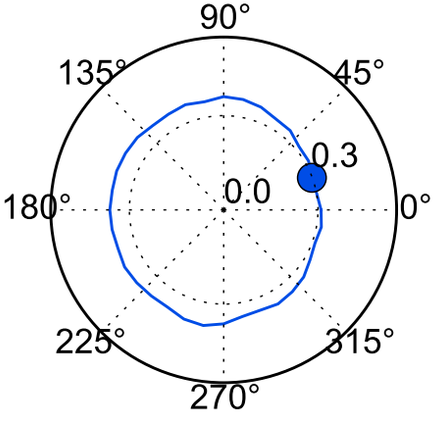
$f'_{21}(x)$

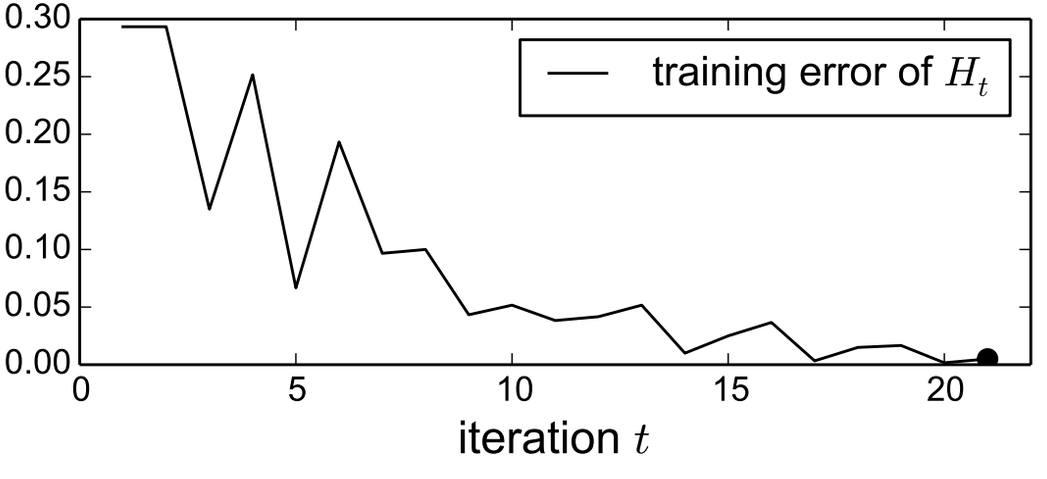


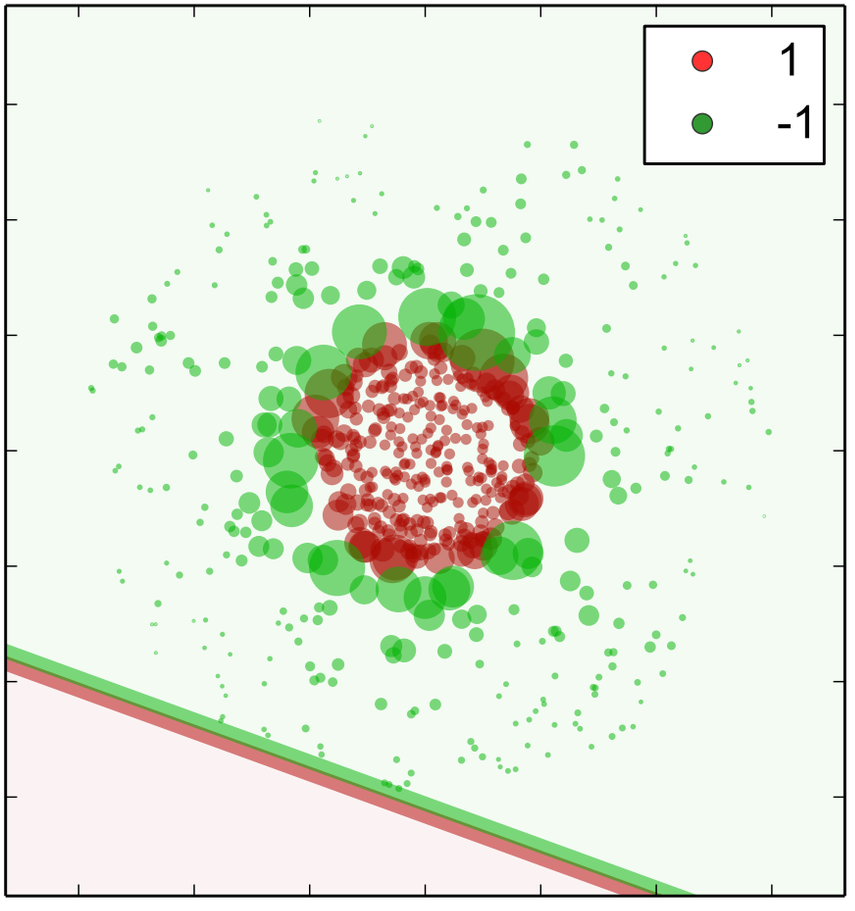
$H_{21}(x) = \text{sign}(f_{21}(x))$



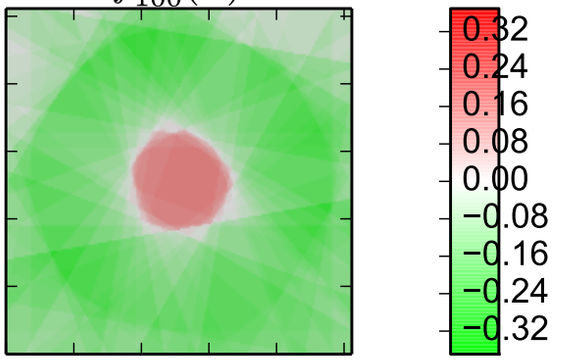
$\epsilon_{21}(w)$
at optimal b



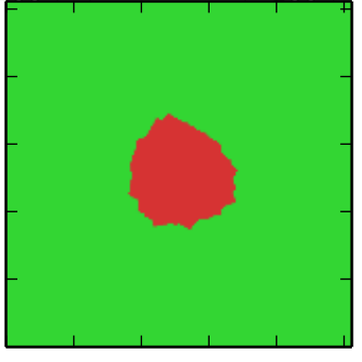




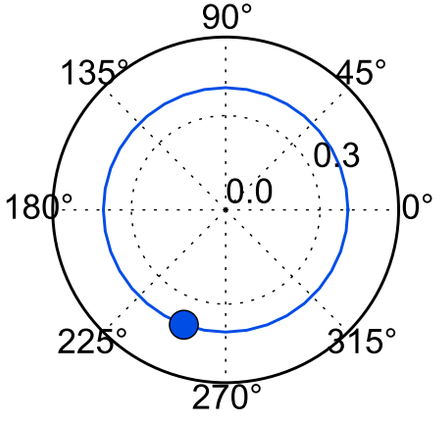
$f'_{100}(x)$

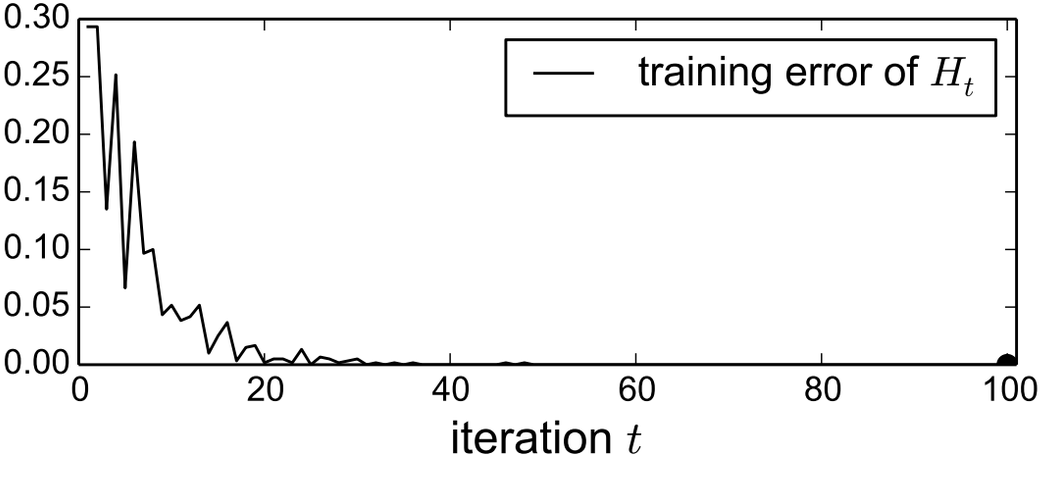


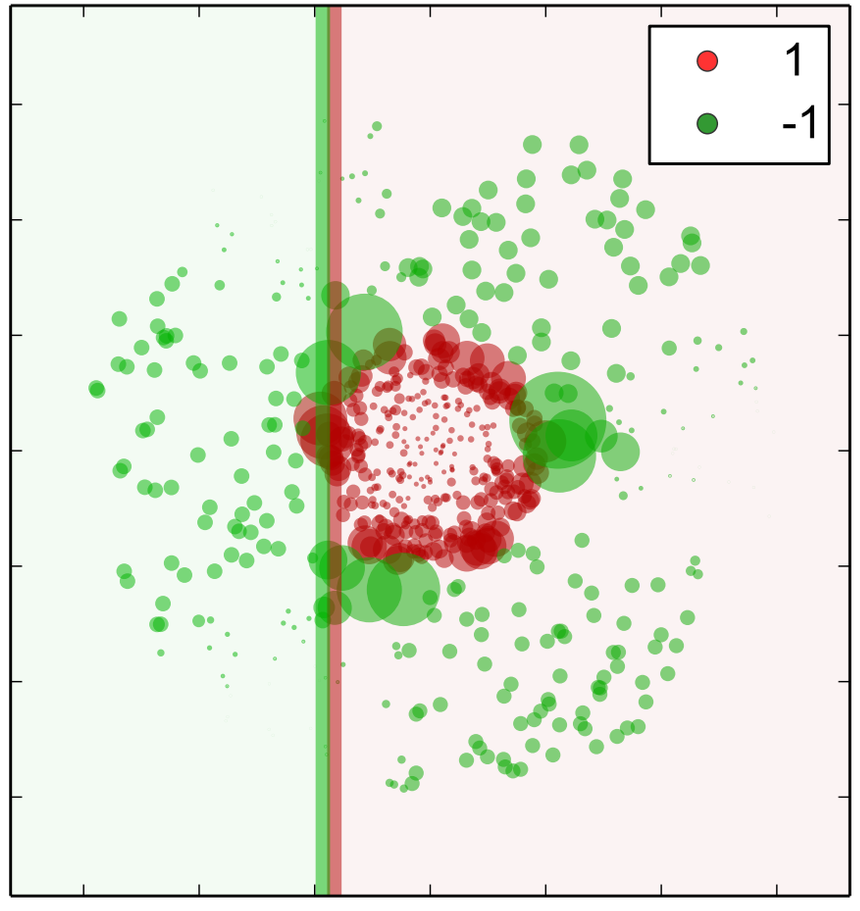
$H_{100}(x) = \text{sign}(f_{100}(x))$



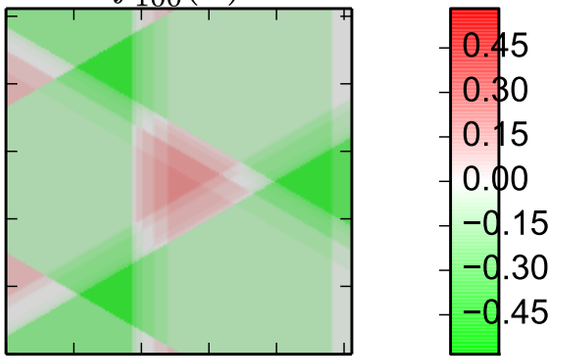
$\epsilon_{100}(w)$
at optimal b



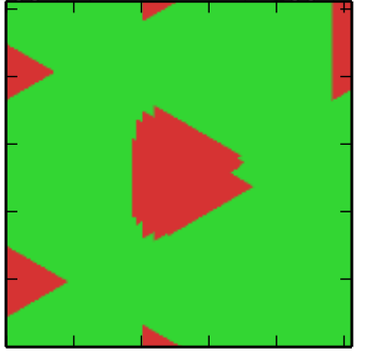




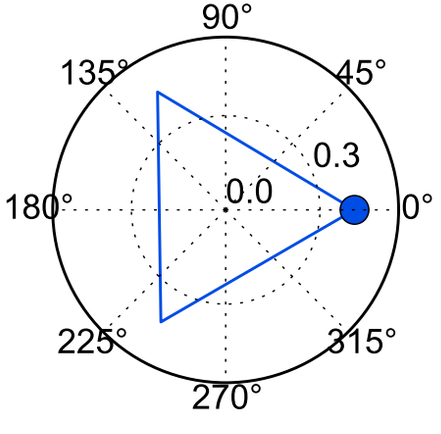
$f'_{100}(x)$

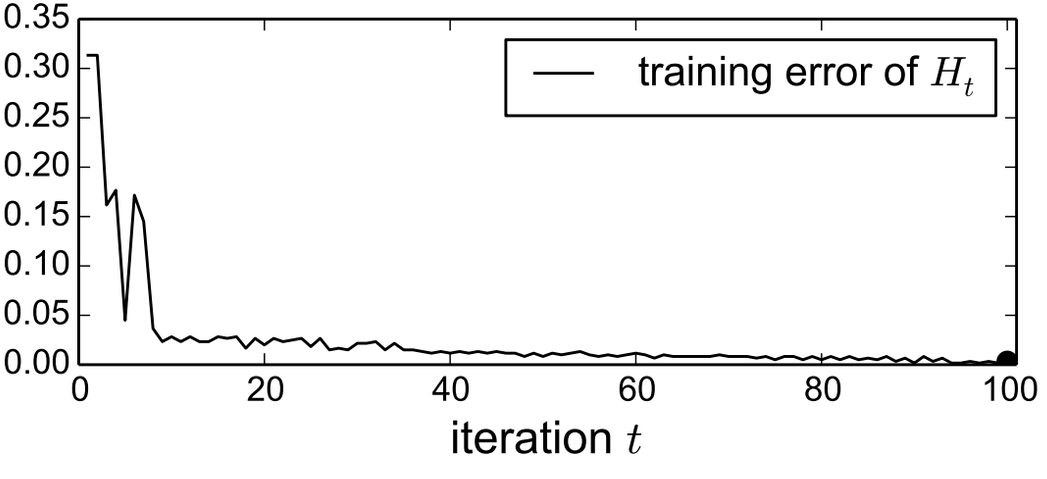


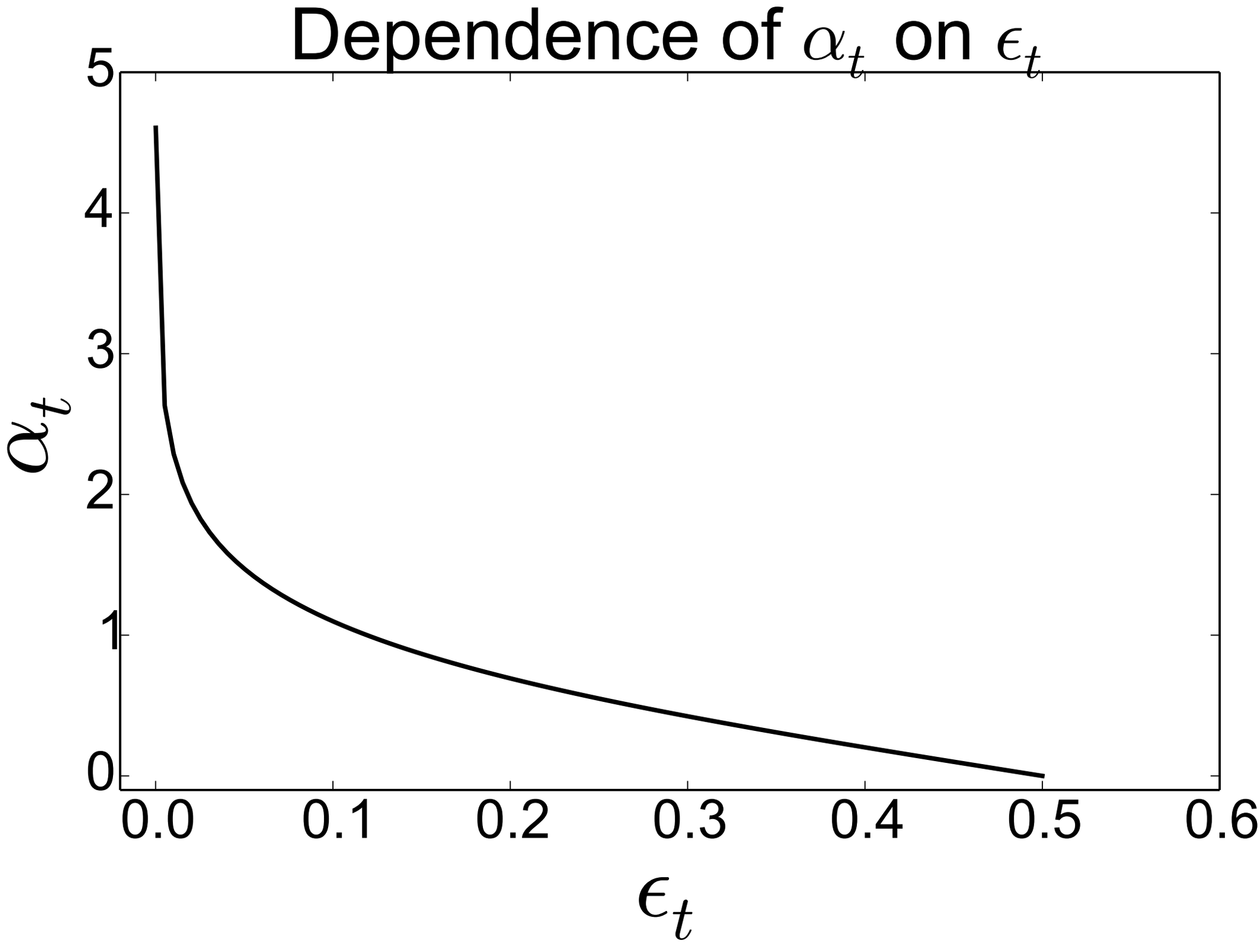
$H_{100}(x) = \text{sign}(f_{100}(x))$

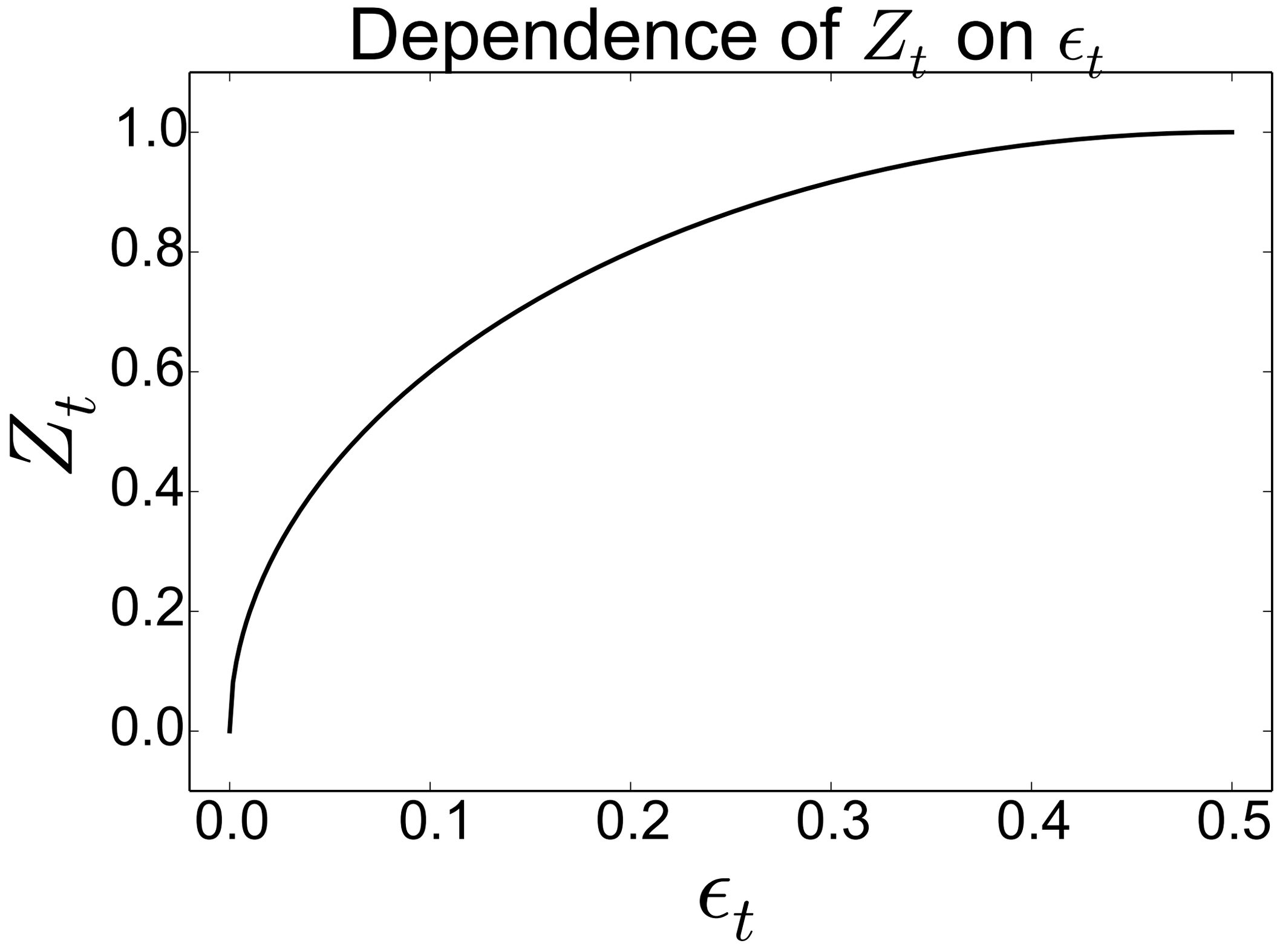


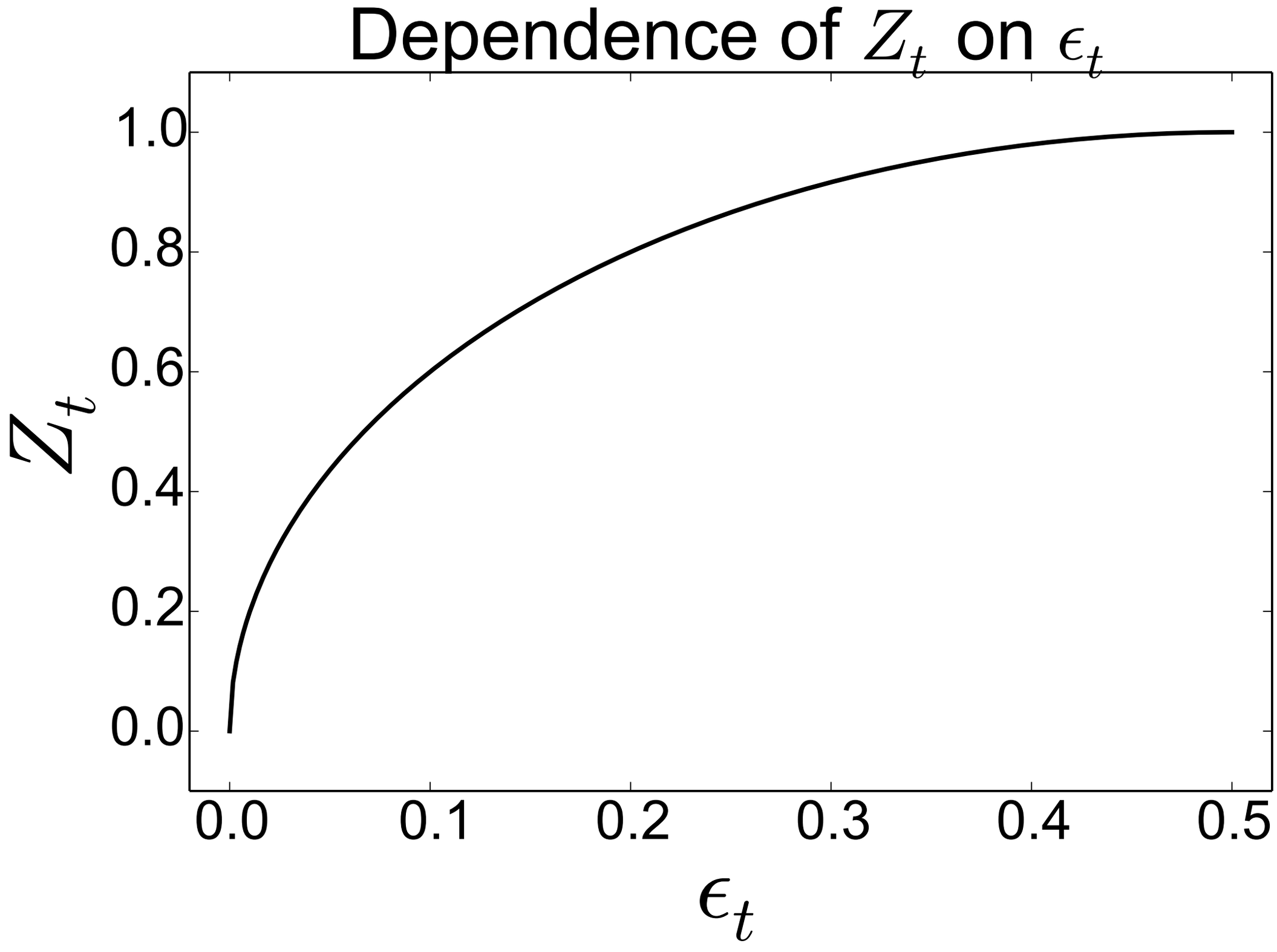
$\epsilon_{100}(w)$
at optimal b

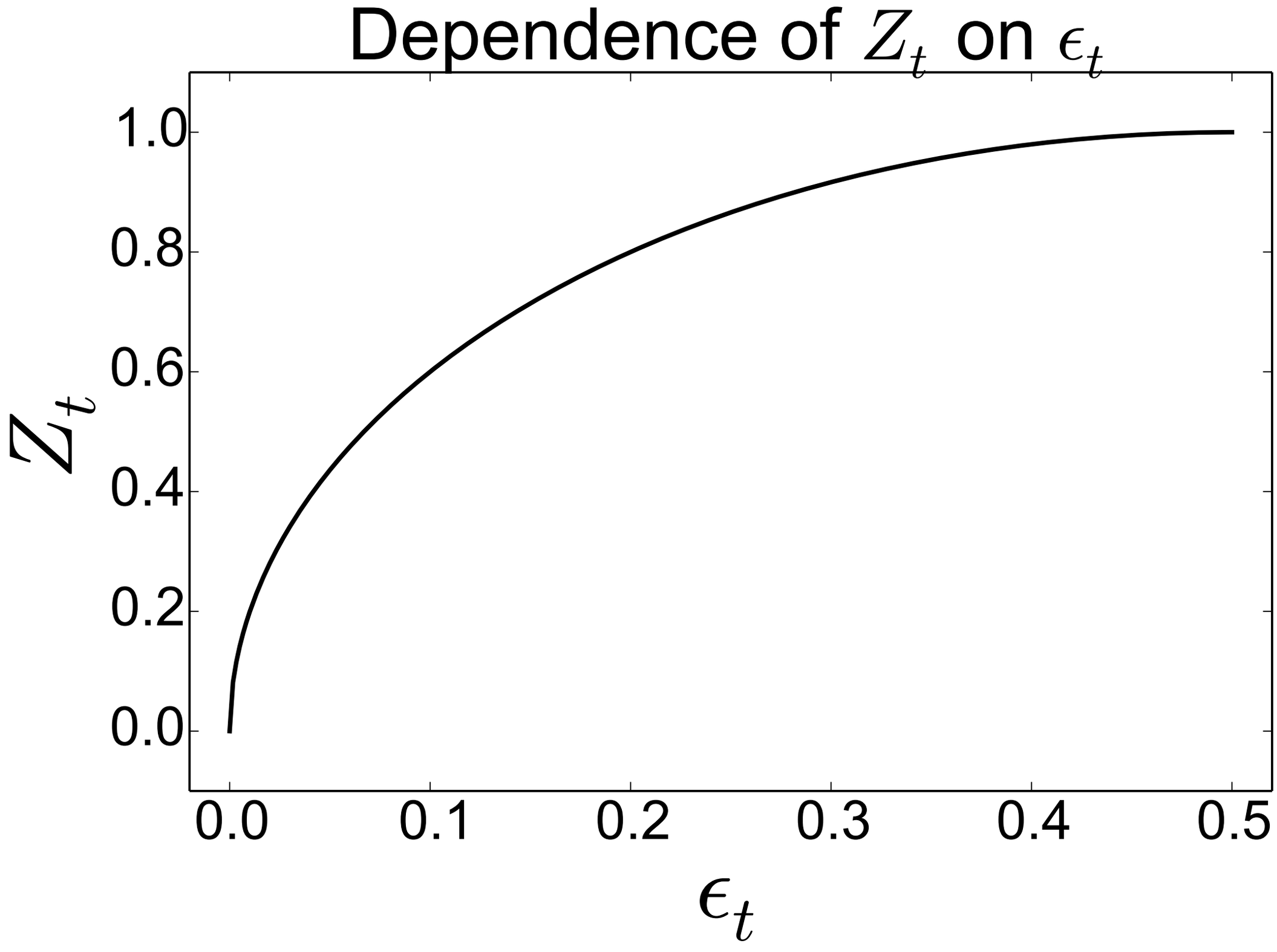


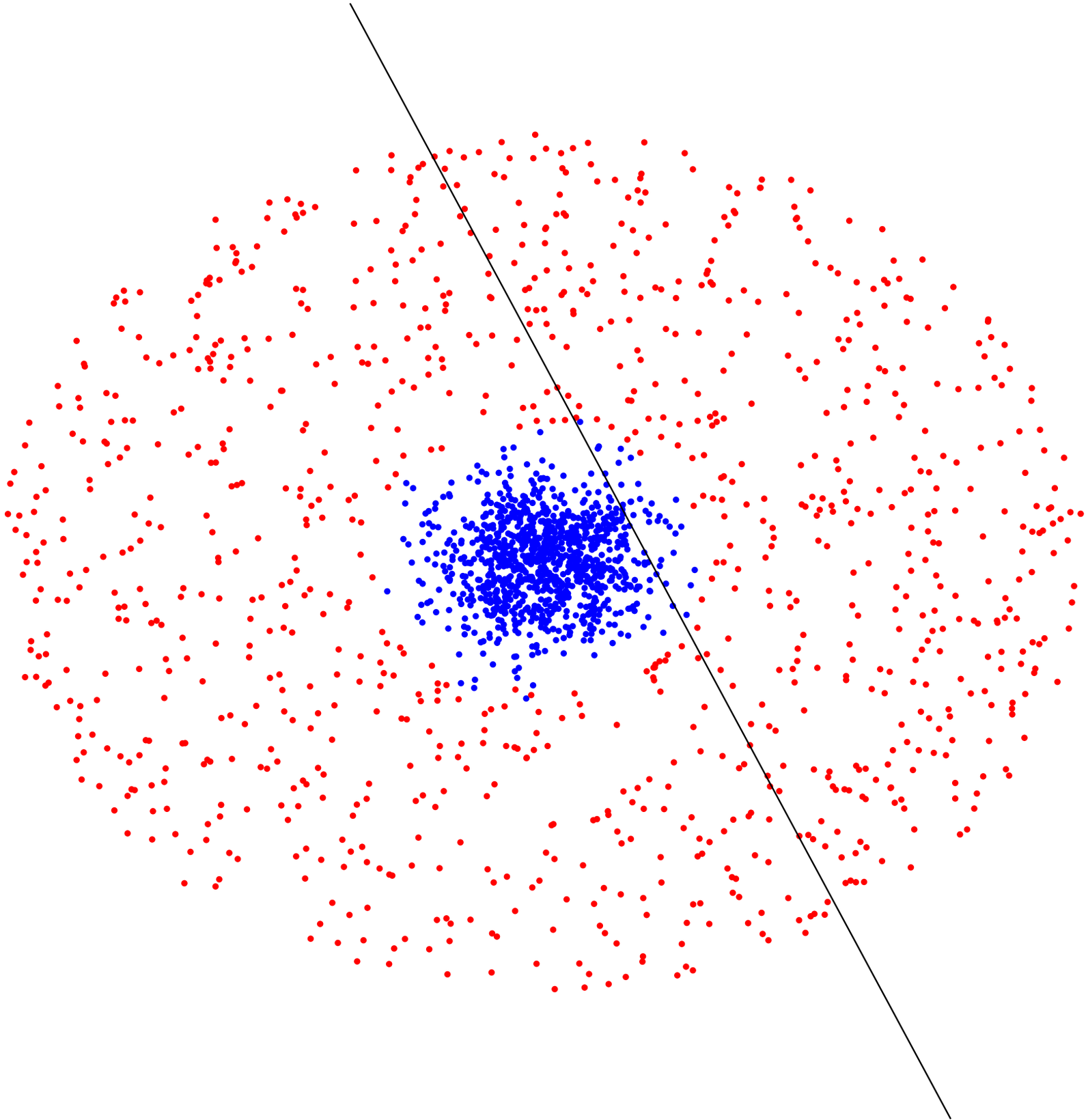


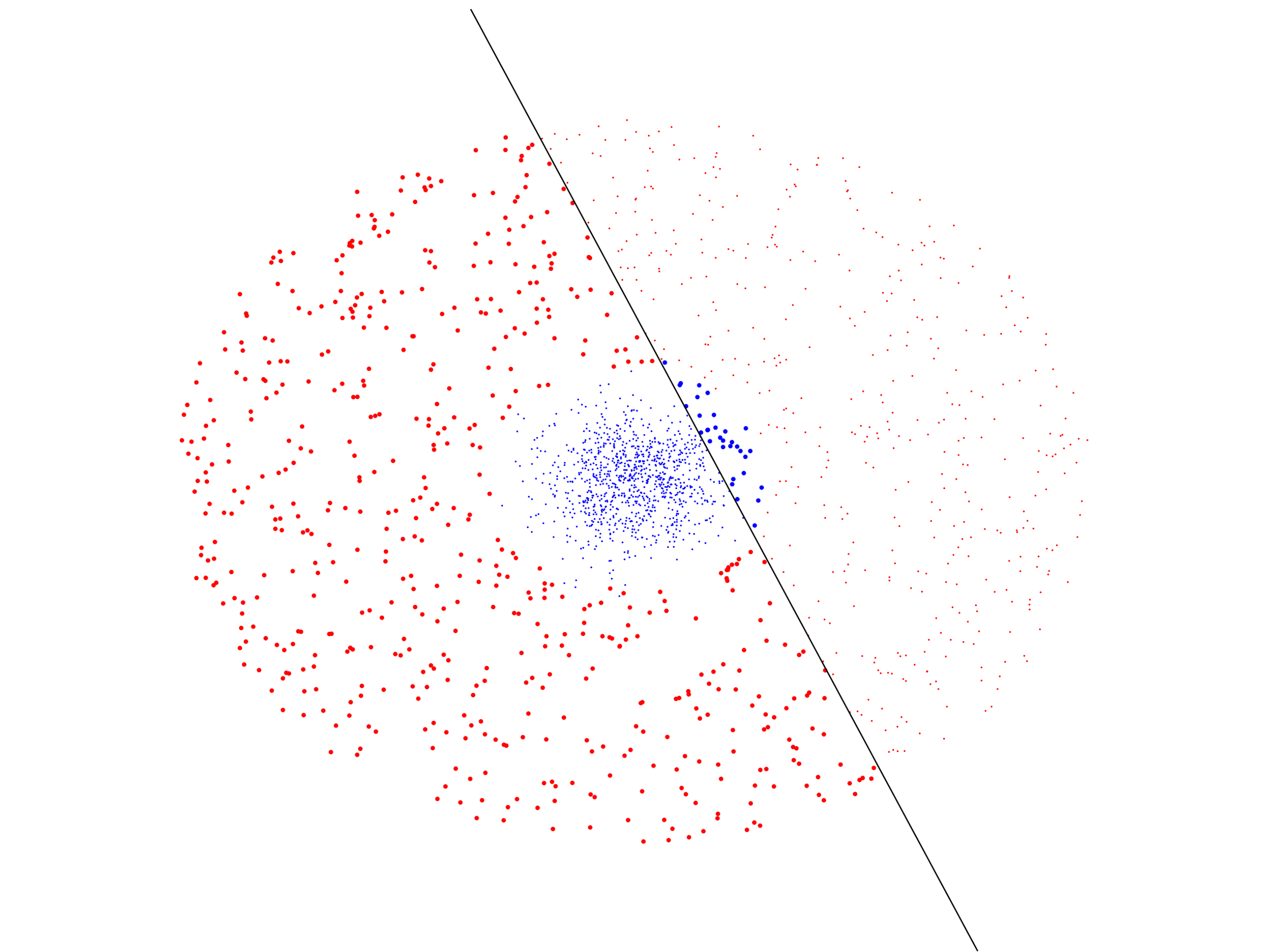

Dependence of α_t on ϵ_t
α_t
ϵ_t
0
1
2
3
4
5
0.0
0.1
0.2
0.3
0.4
0.5
0.6

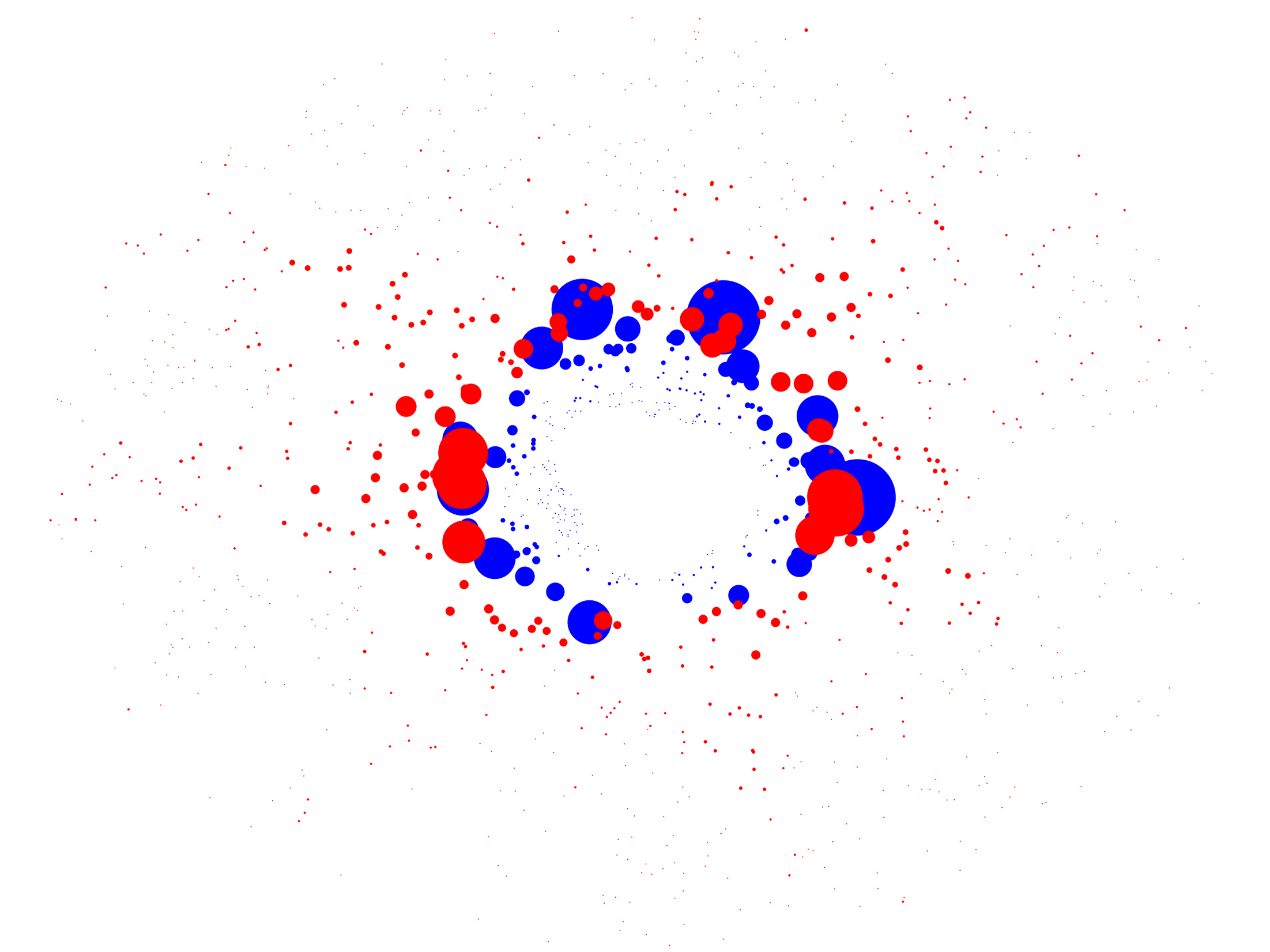

Dependence of Z_t on ϵ_t
1.0
0.8
0.6
0.4
0.2
0.0
Z_t
0.0
0.1
0.2
0.3
0.4
0.5
ϵ_t

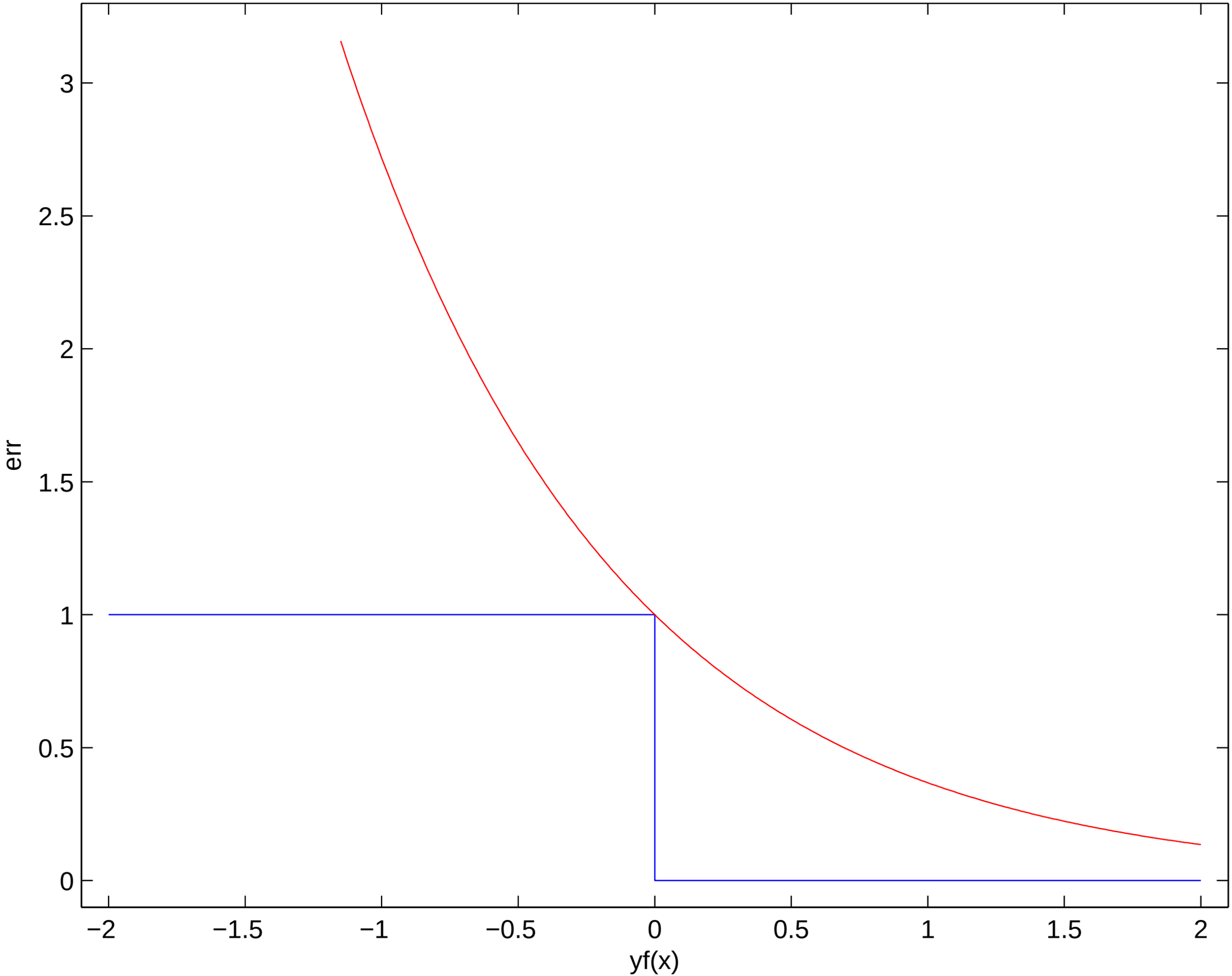

Dependence of Z_t on ϵ_t
1.0
0.8
0.6
0.4
0.2
0.0
Z_t
0.0
0.1
0.2
0.3
0.4
0.5
ϵ_t

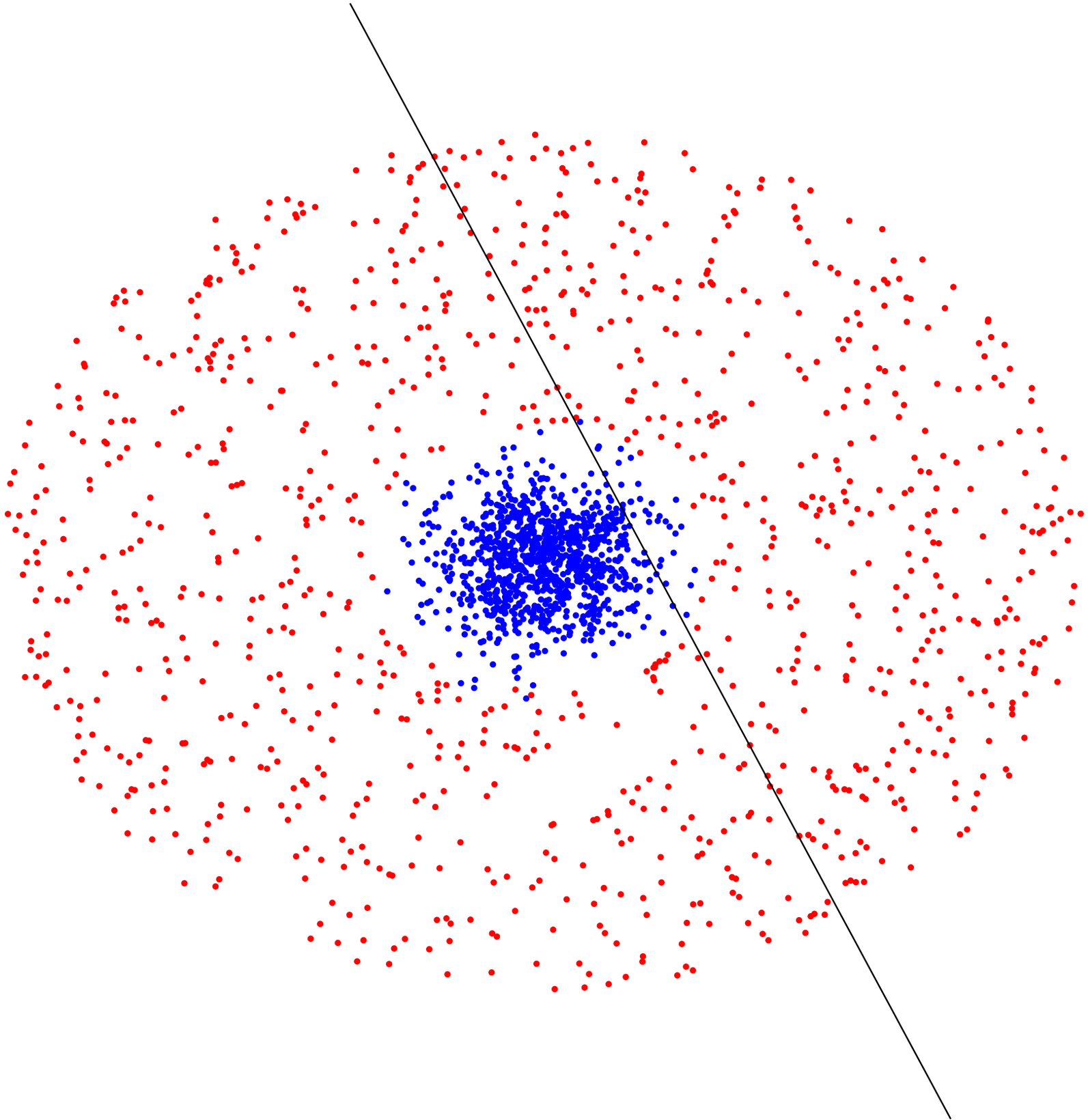

Dependence of Z_t on ϵ_t
1.0
0.8
0.6
0.4
0.2
0.0
Z_t
0.0
0.1
0.2
0.3
0.4
0.5
ϵ_t

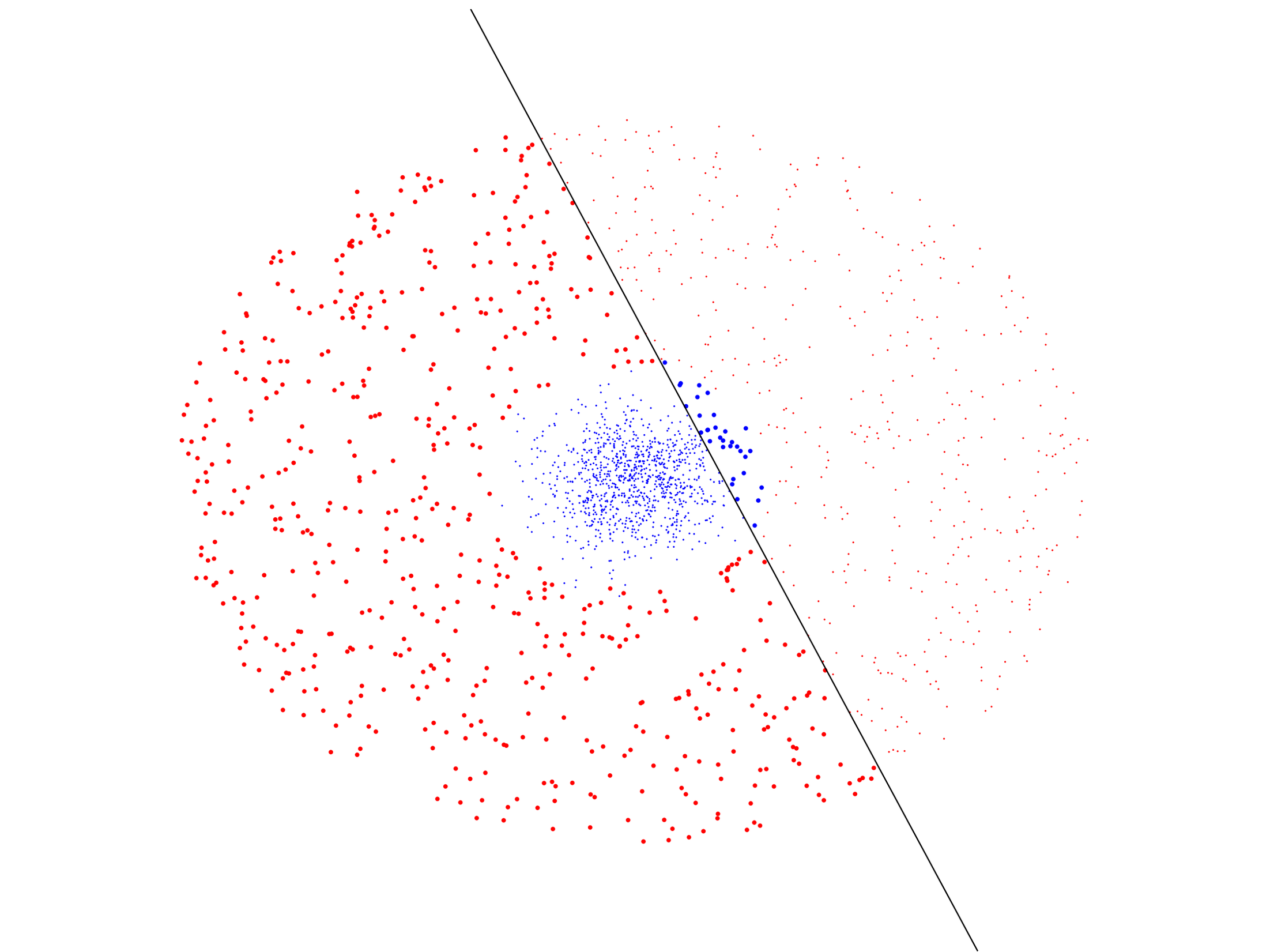


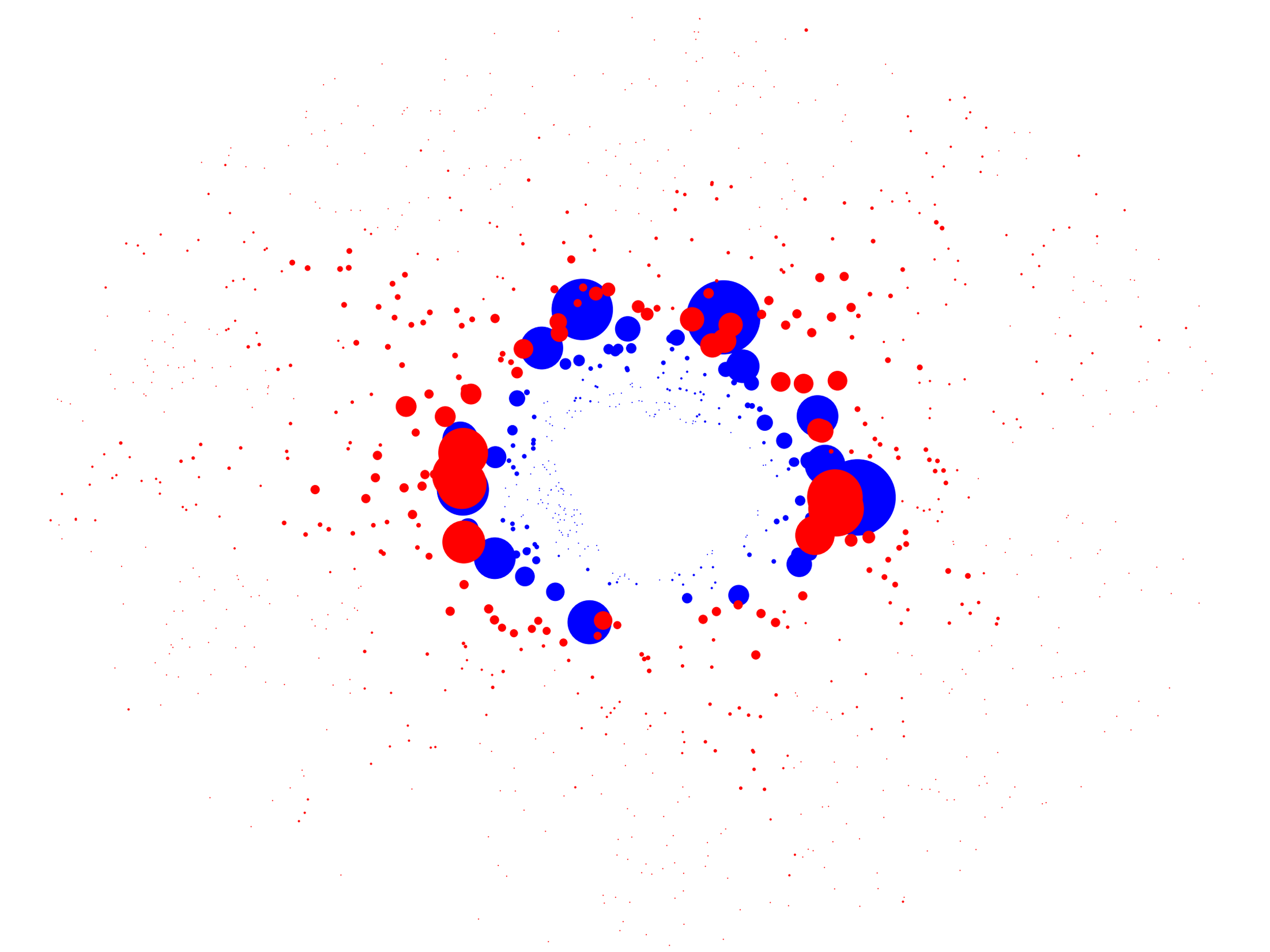


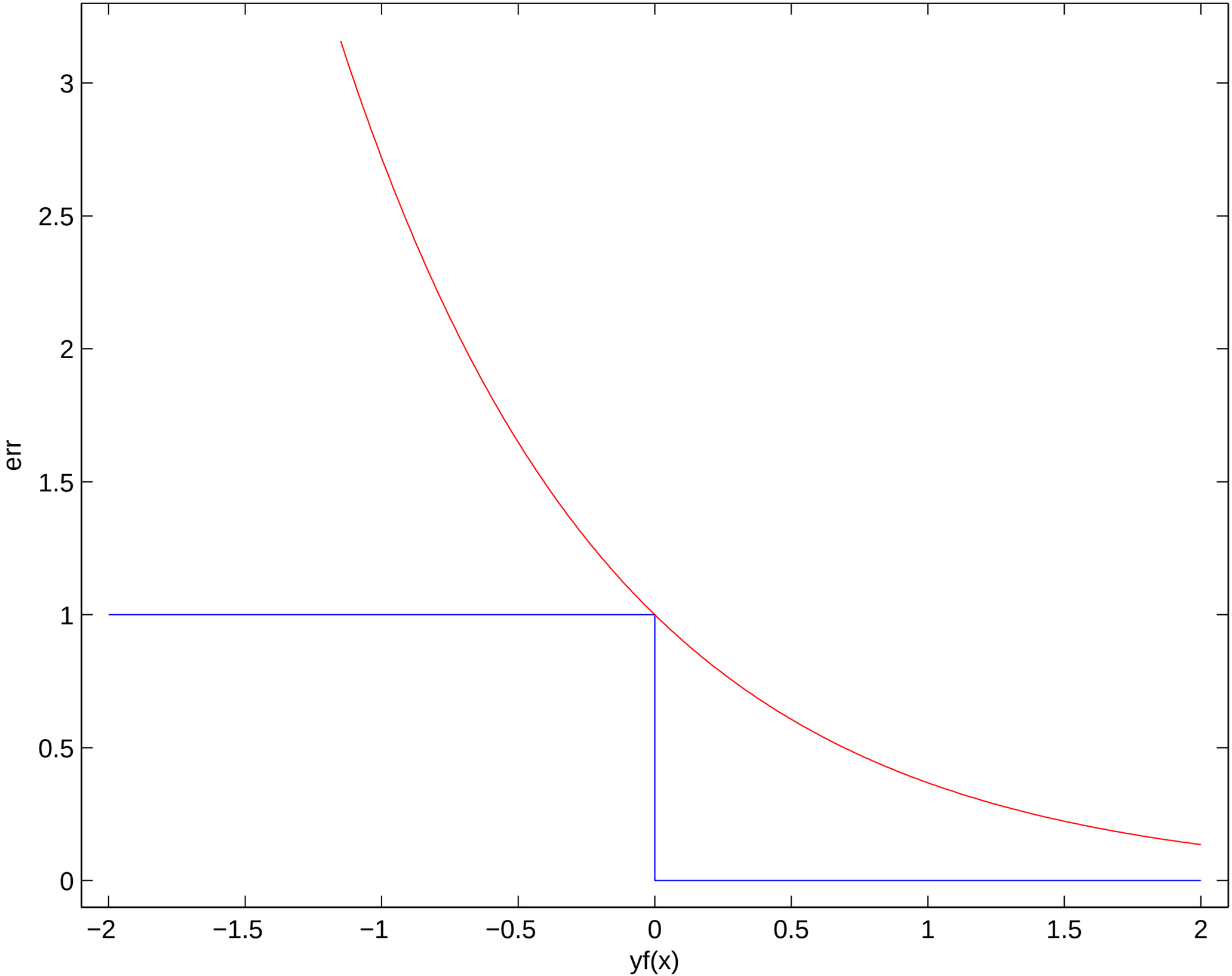


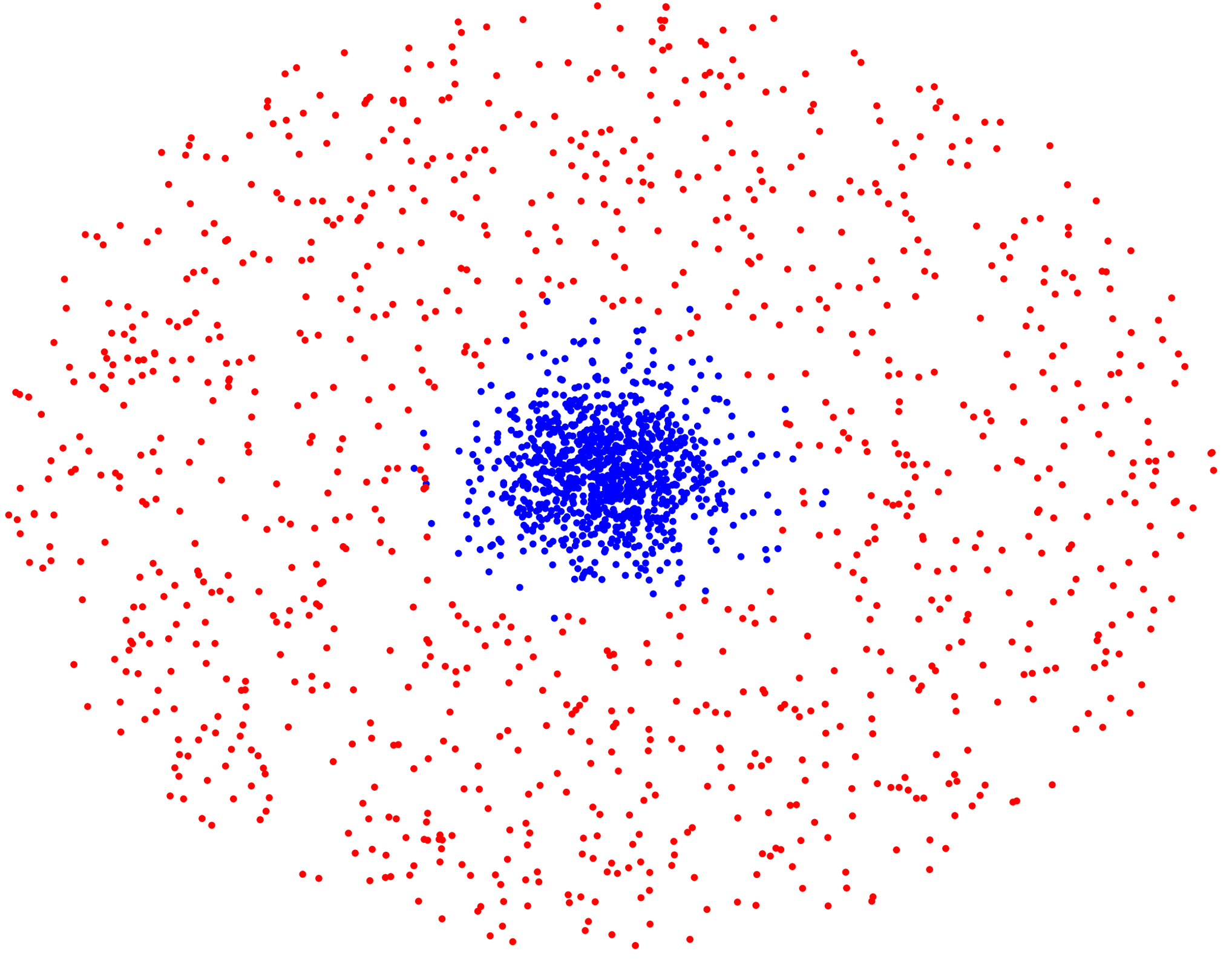

err
yf(x)
3
2.5
2
1.5
1
0.5
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2

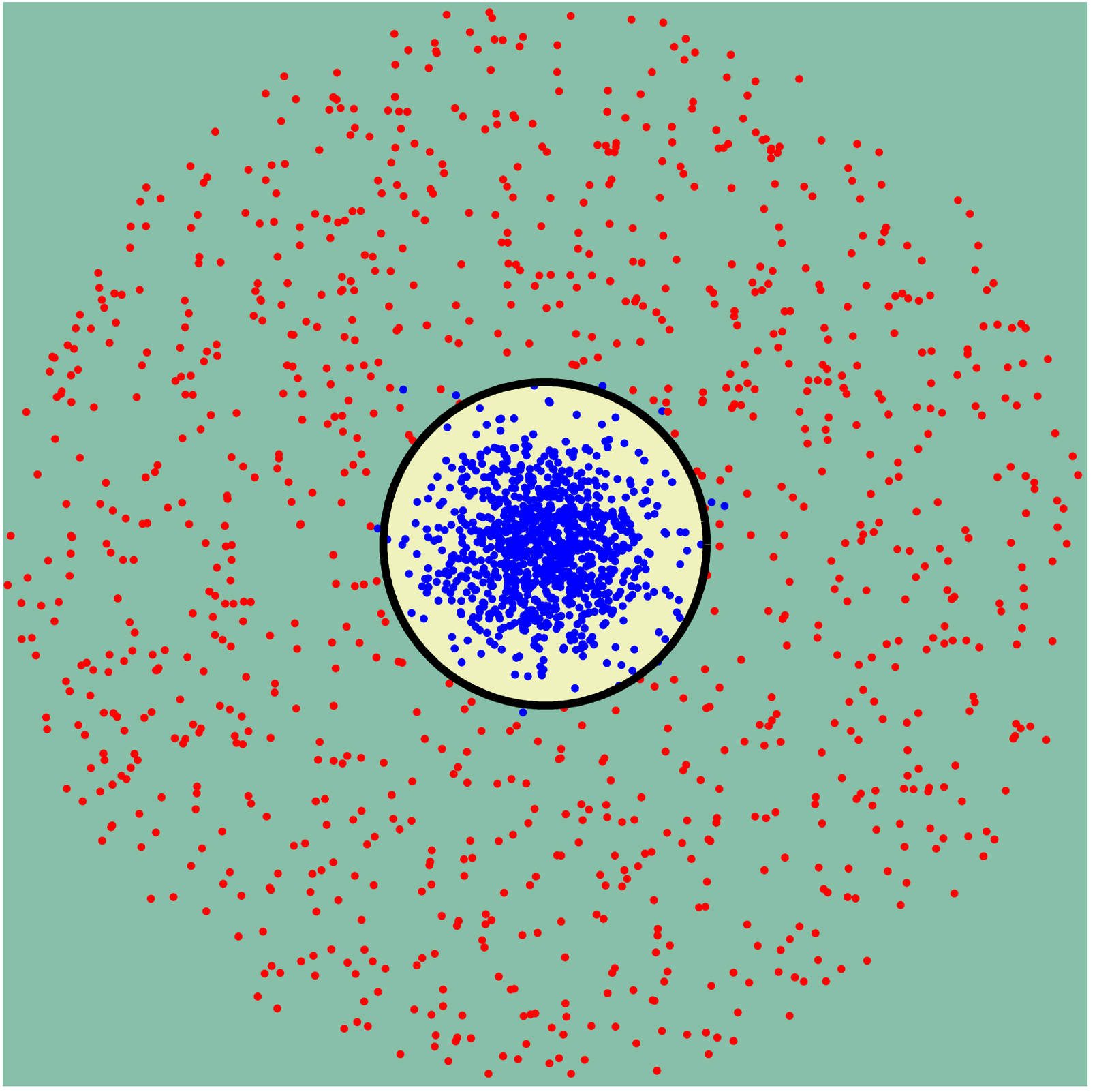


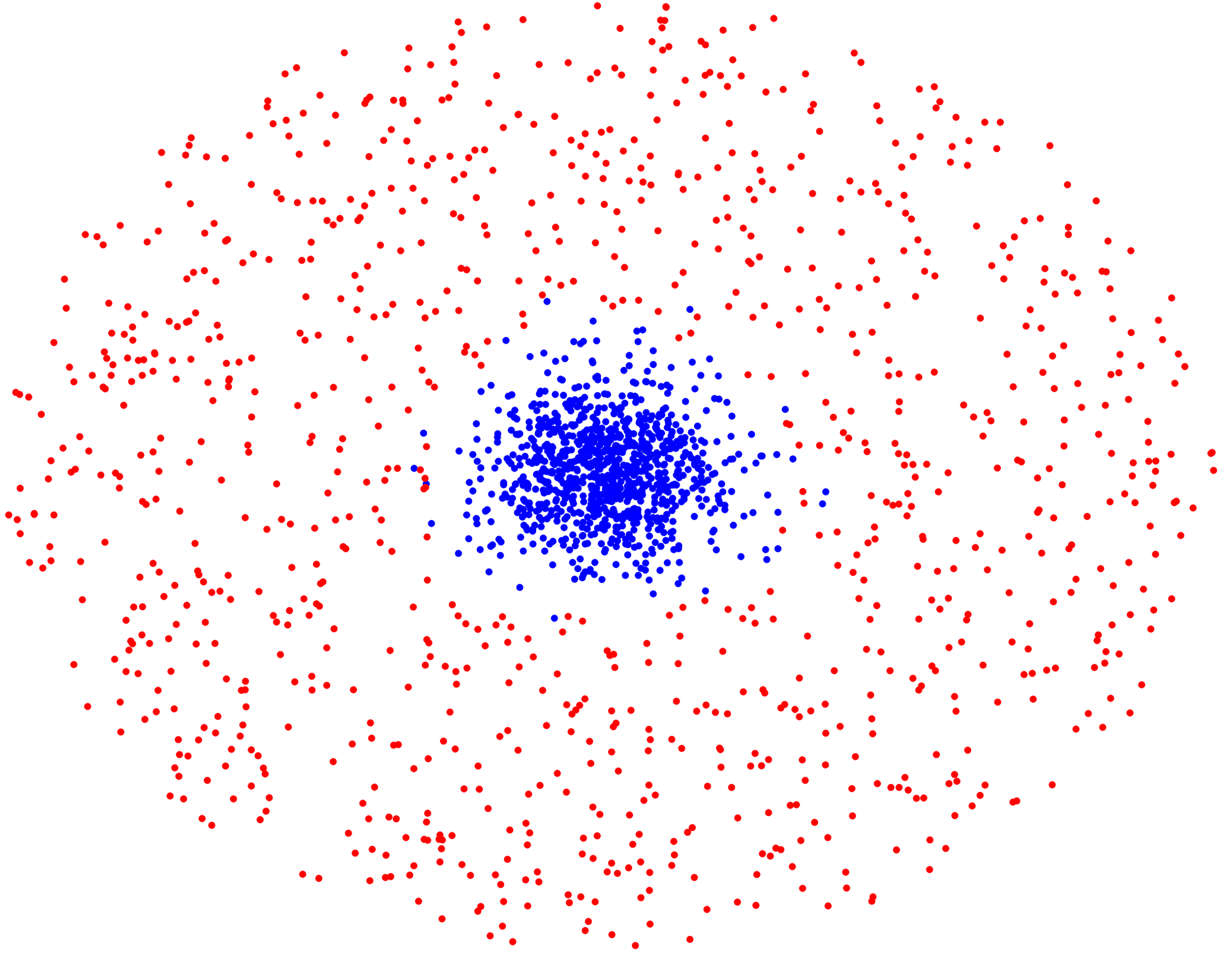


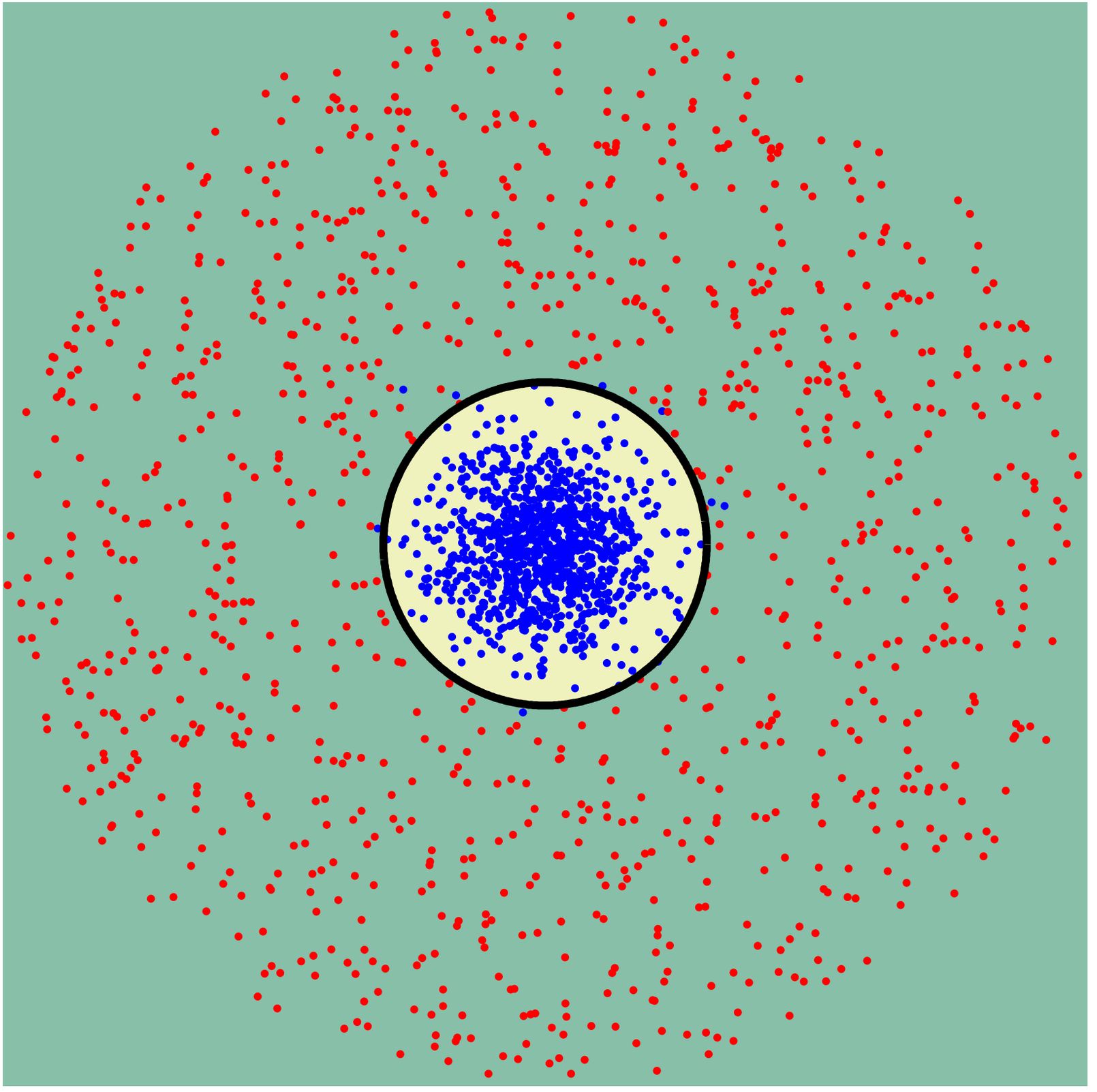


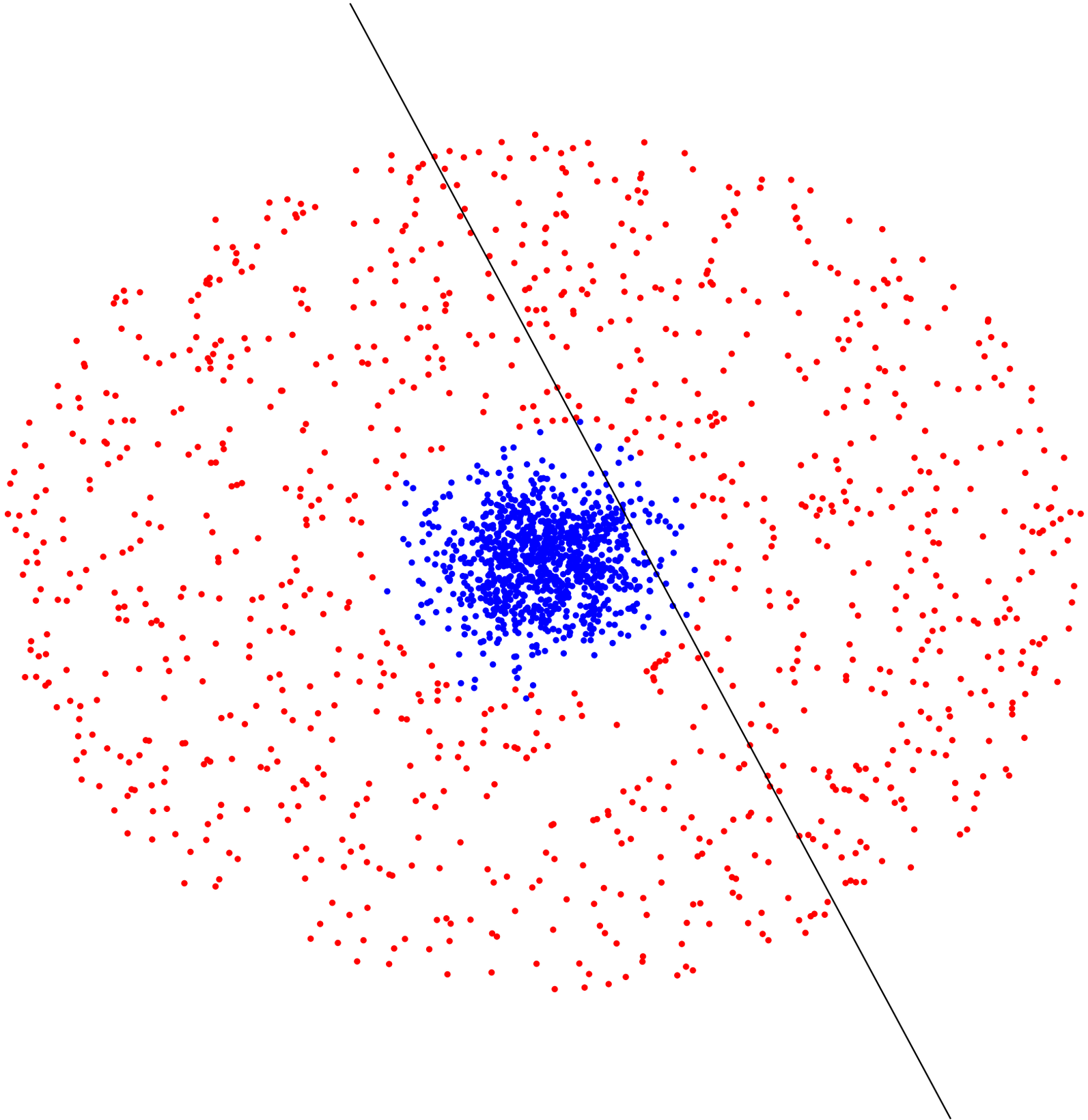

err
yf(x)
3
2.5
2
1.5
1
0.5
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2

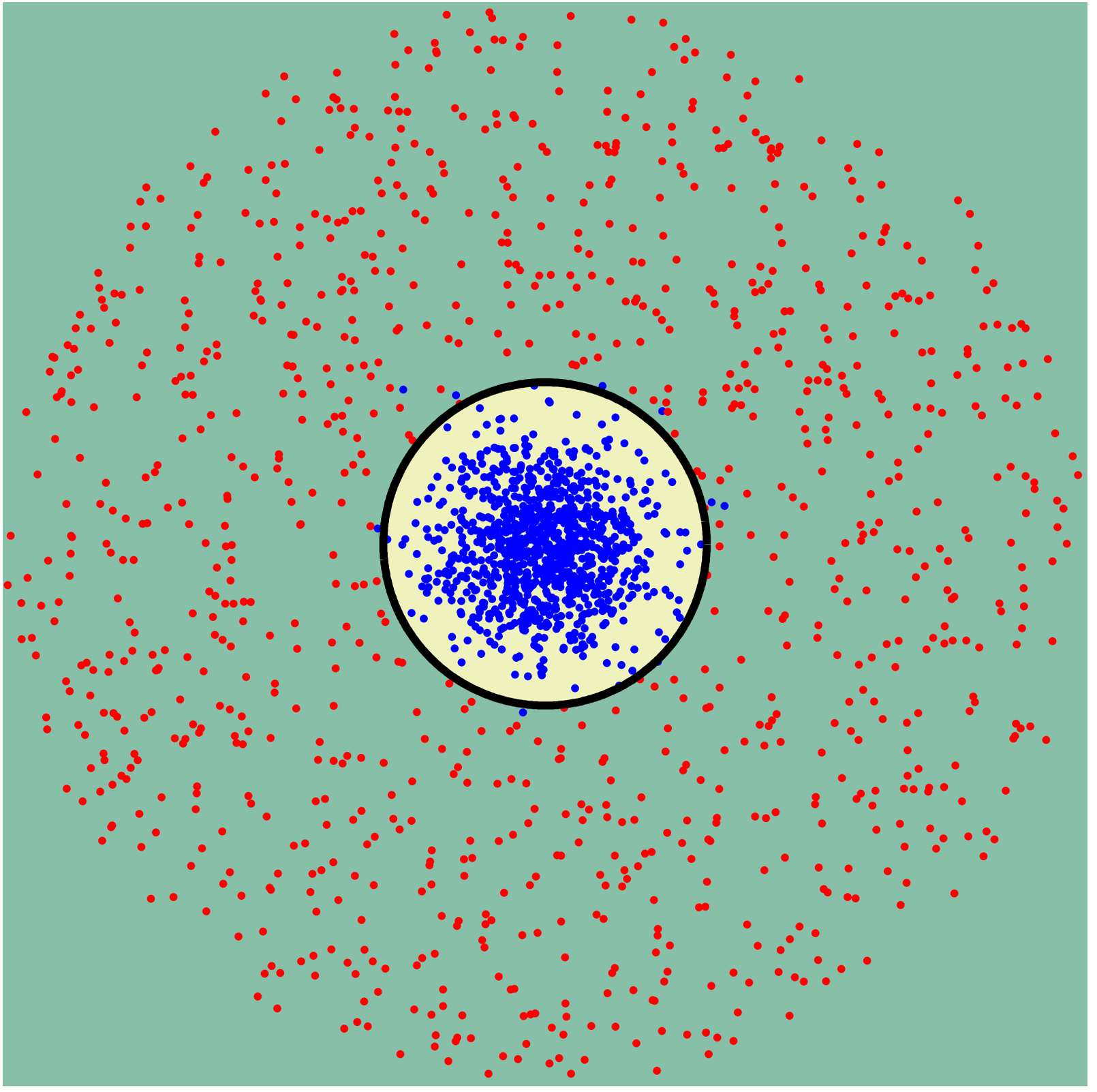


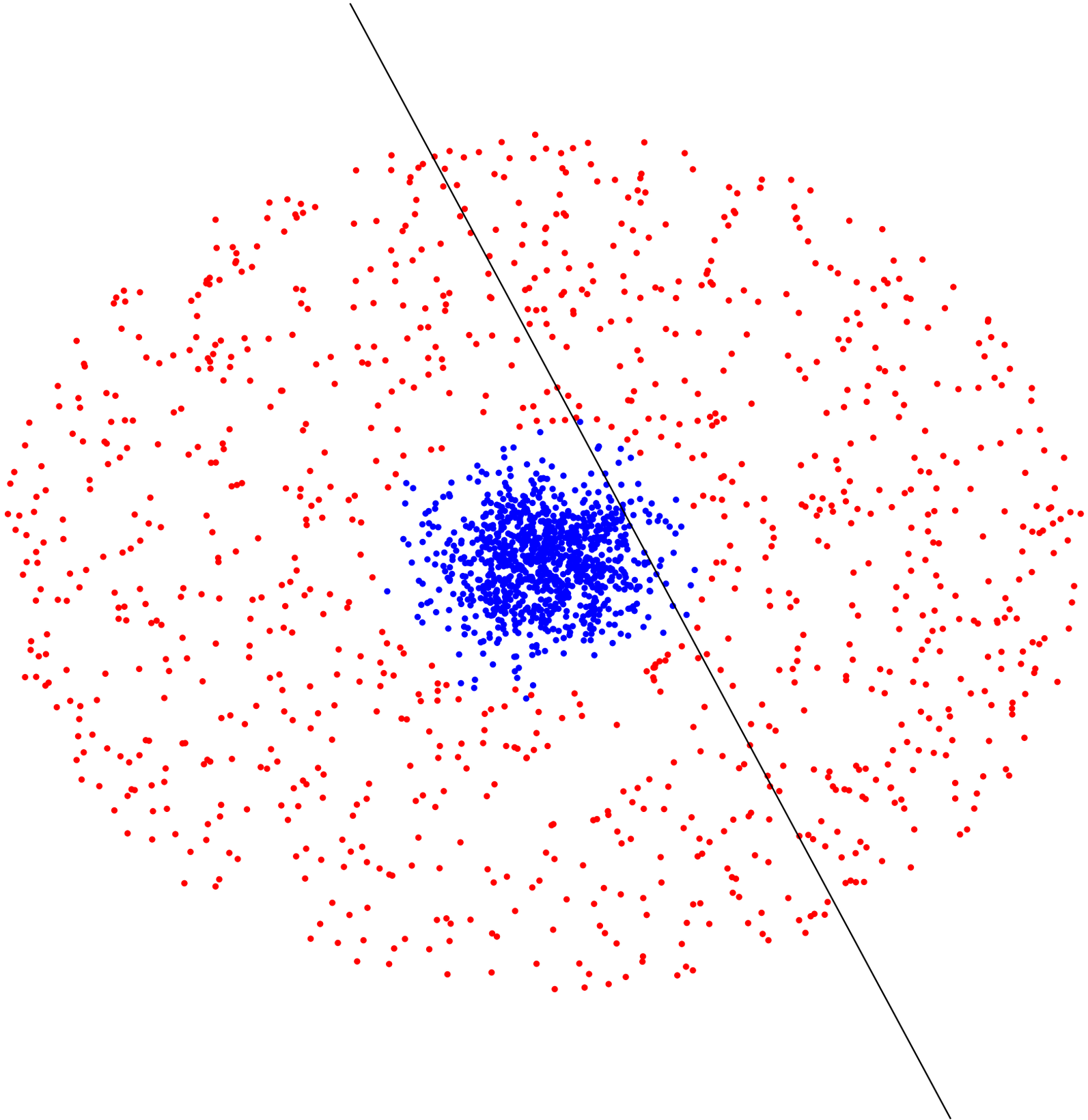


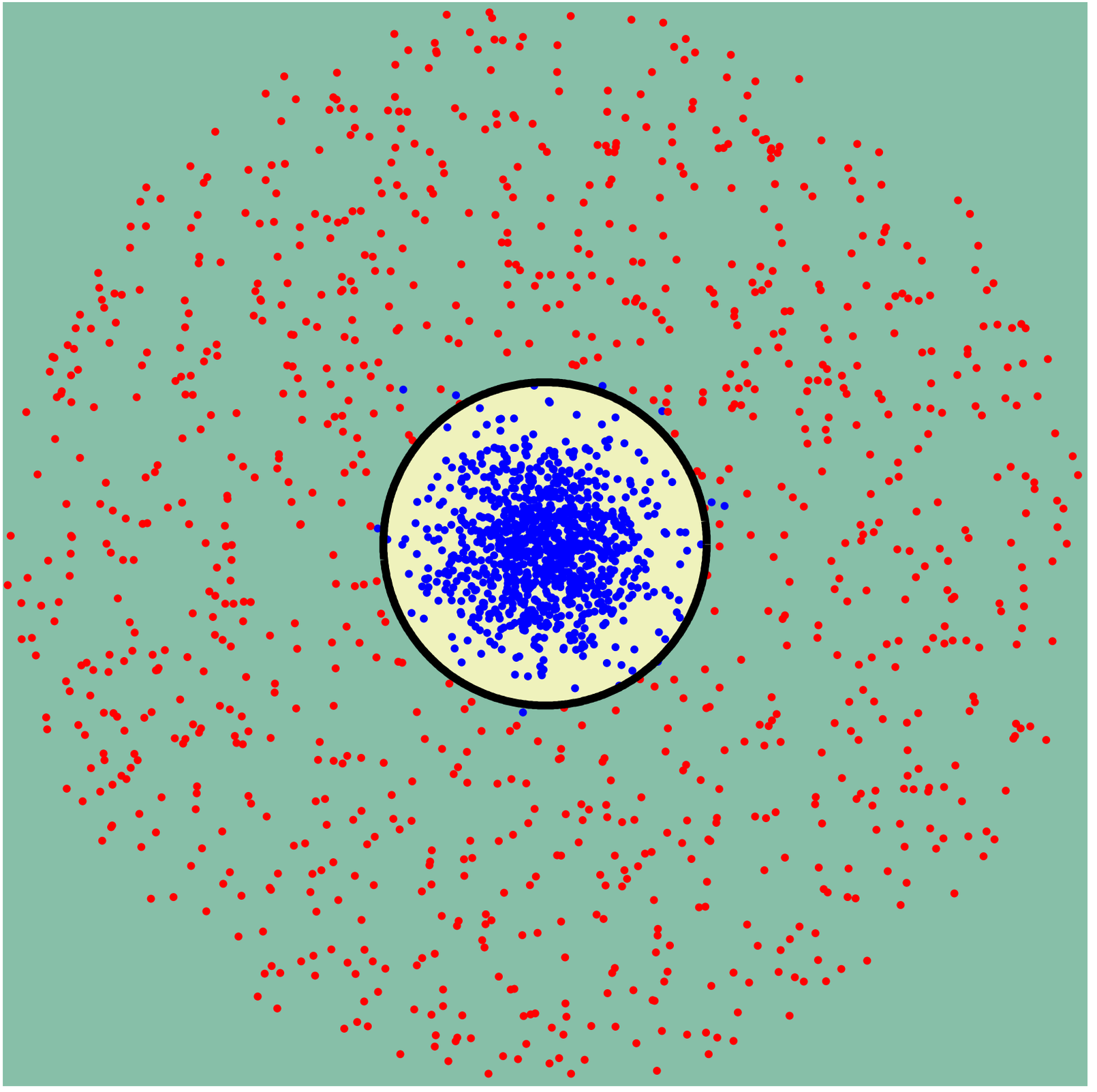


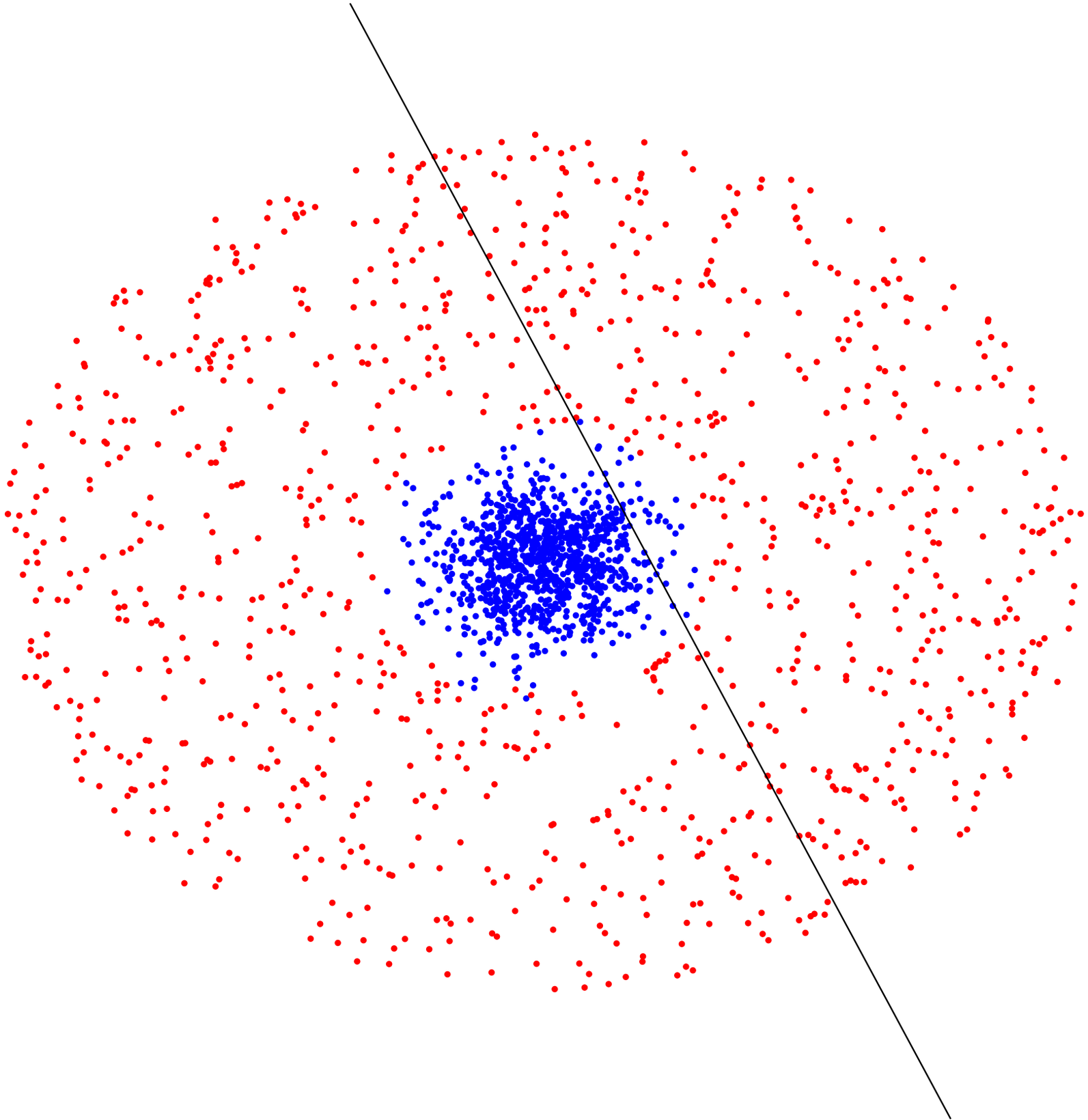


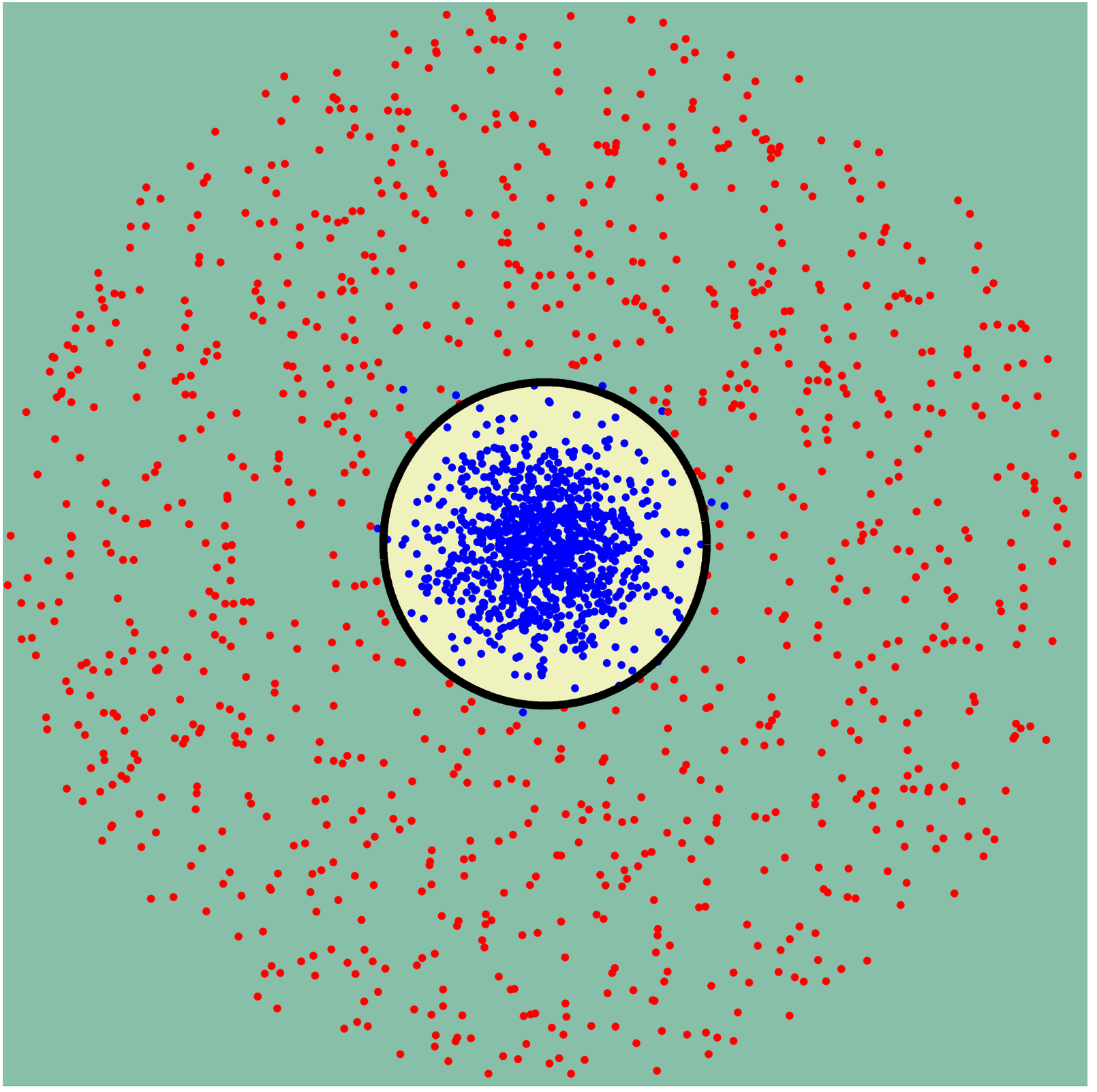


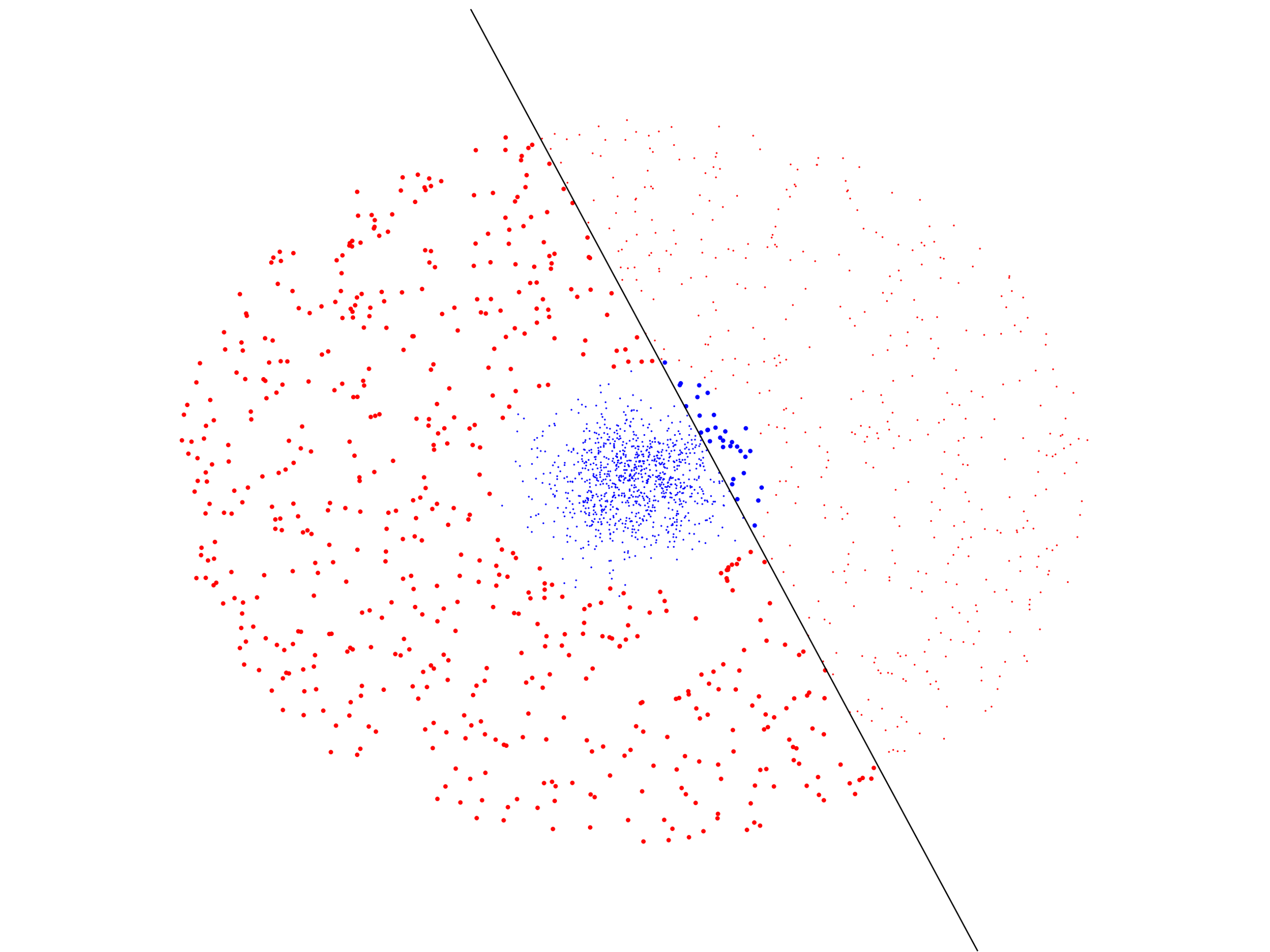


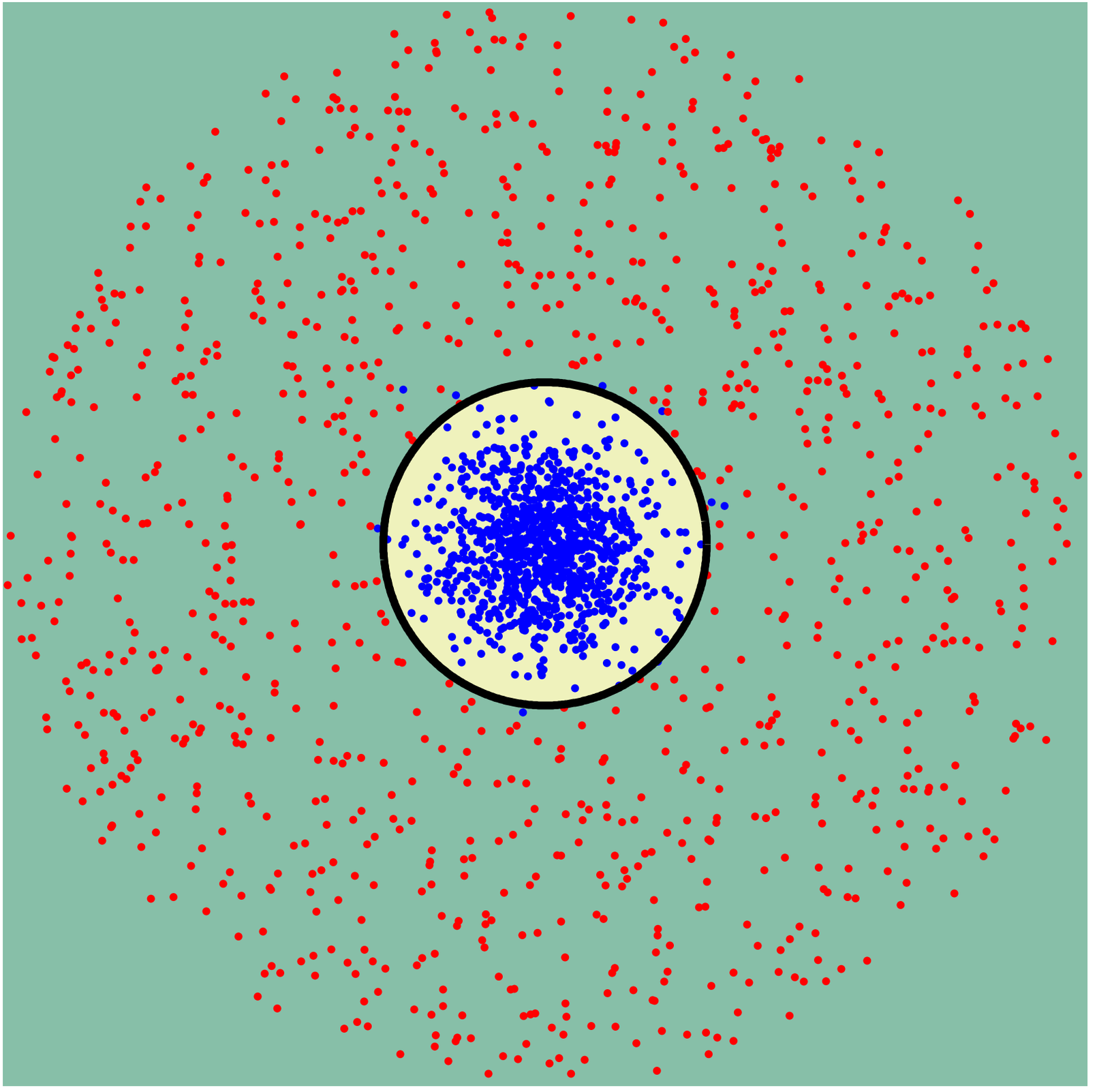


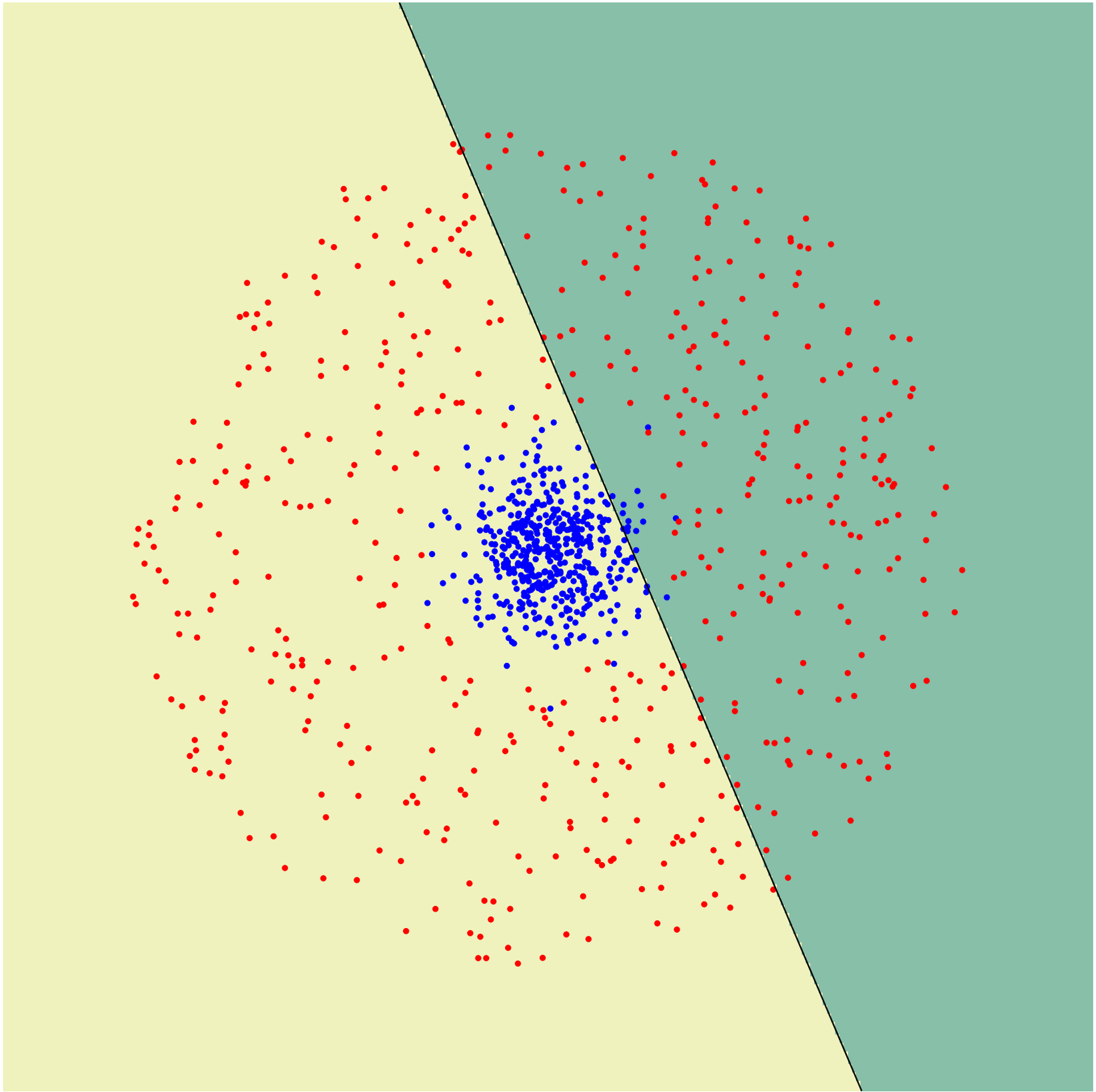


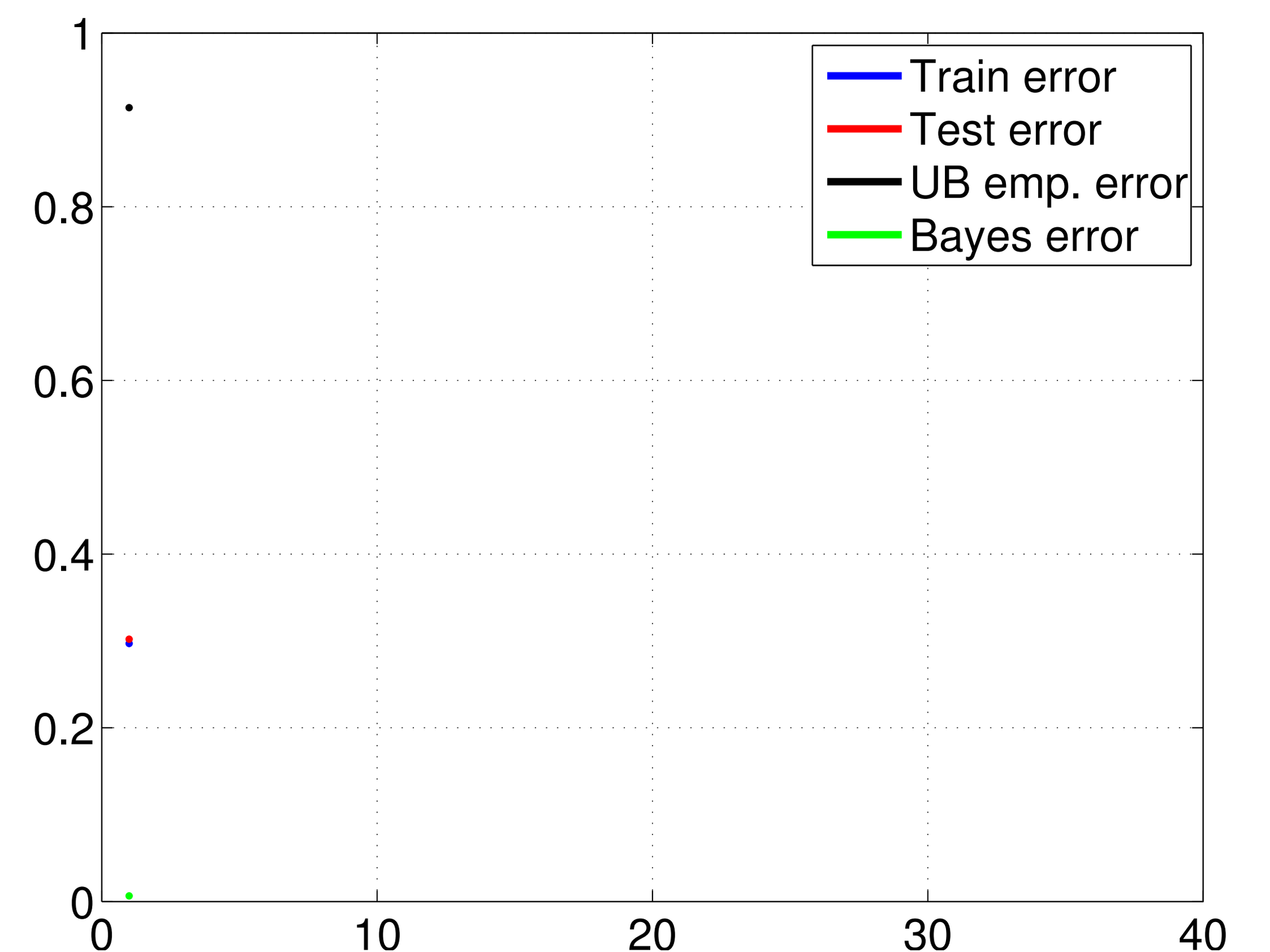


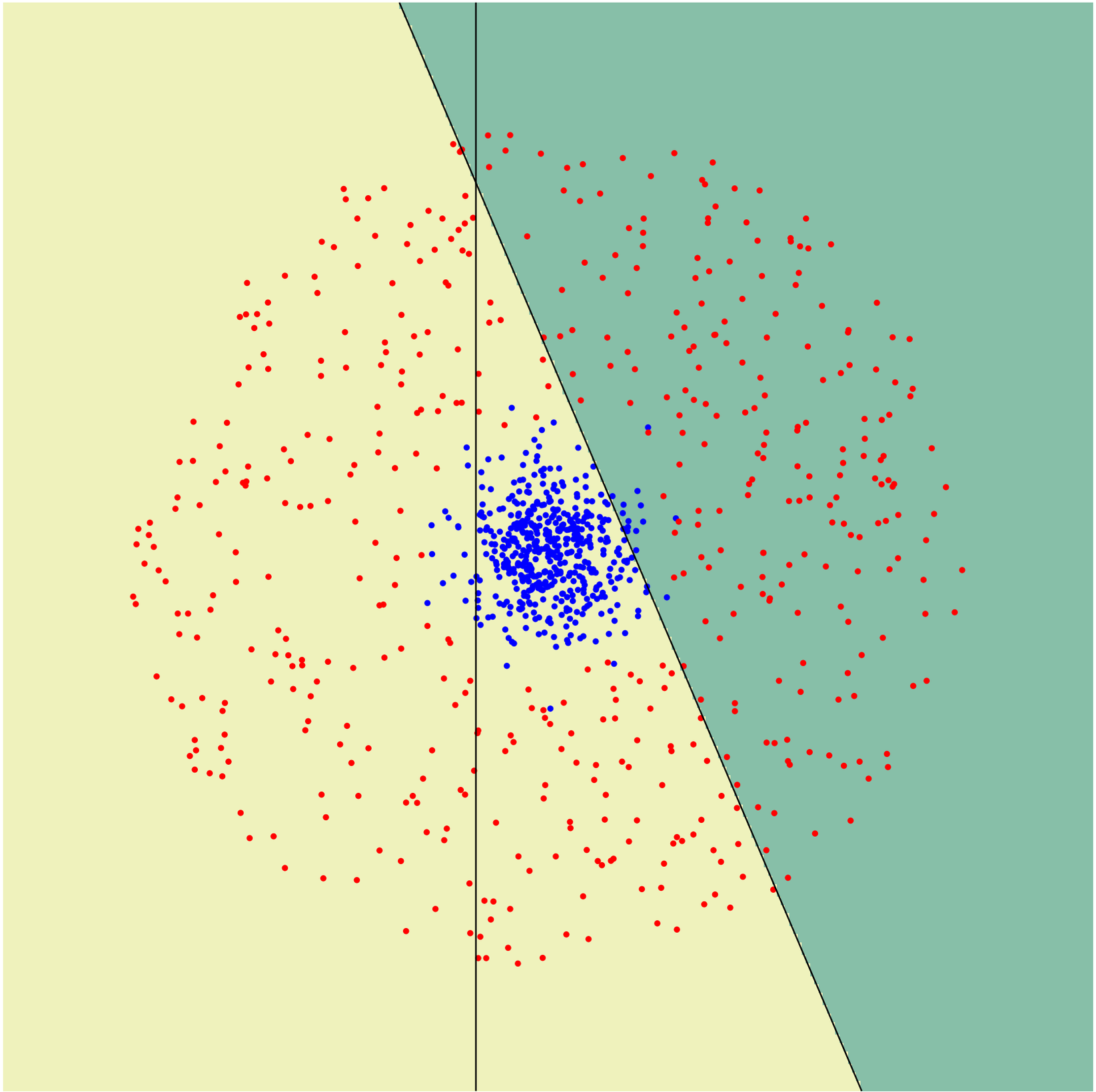


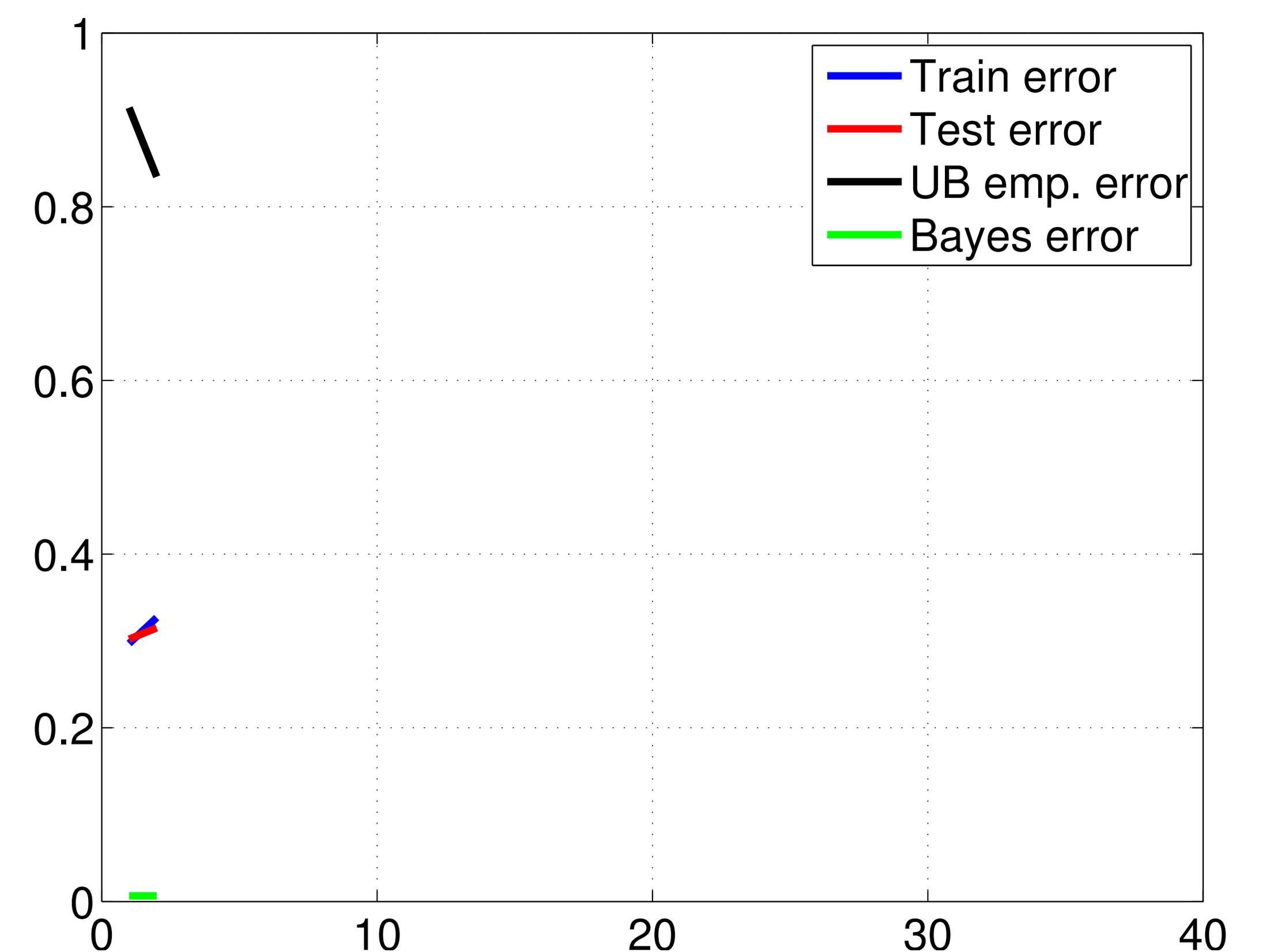


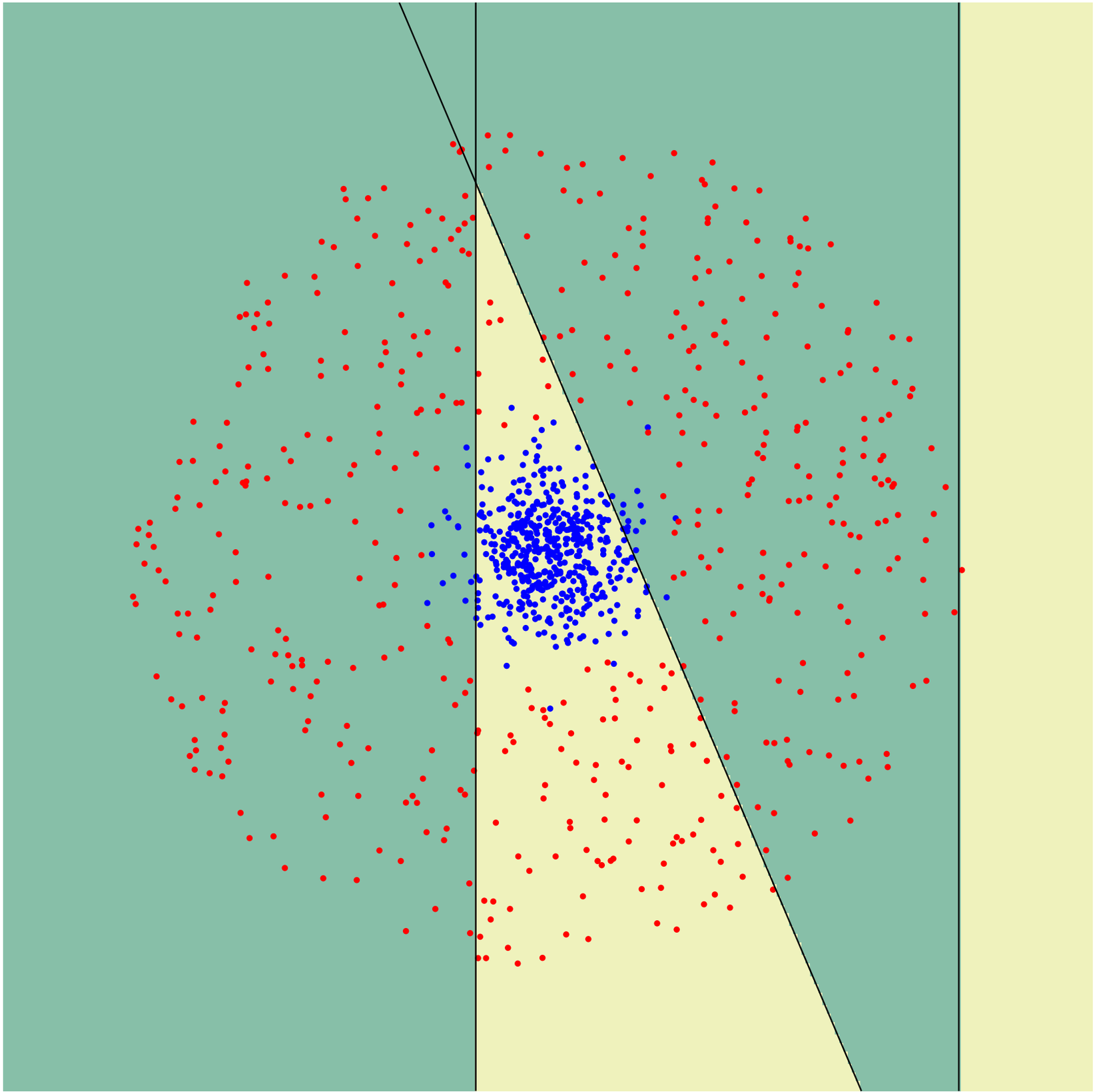


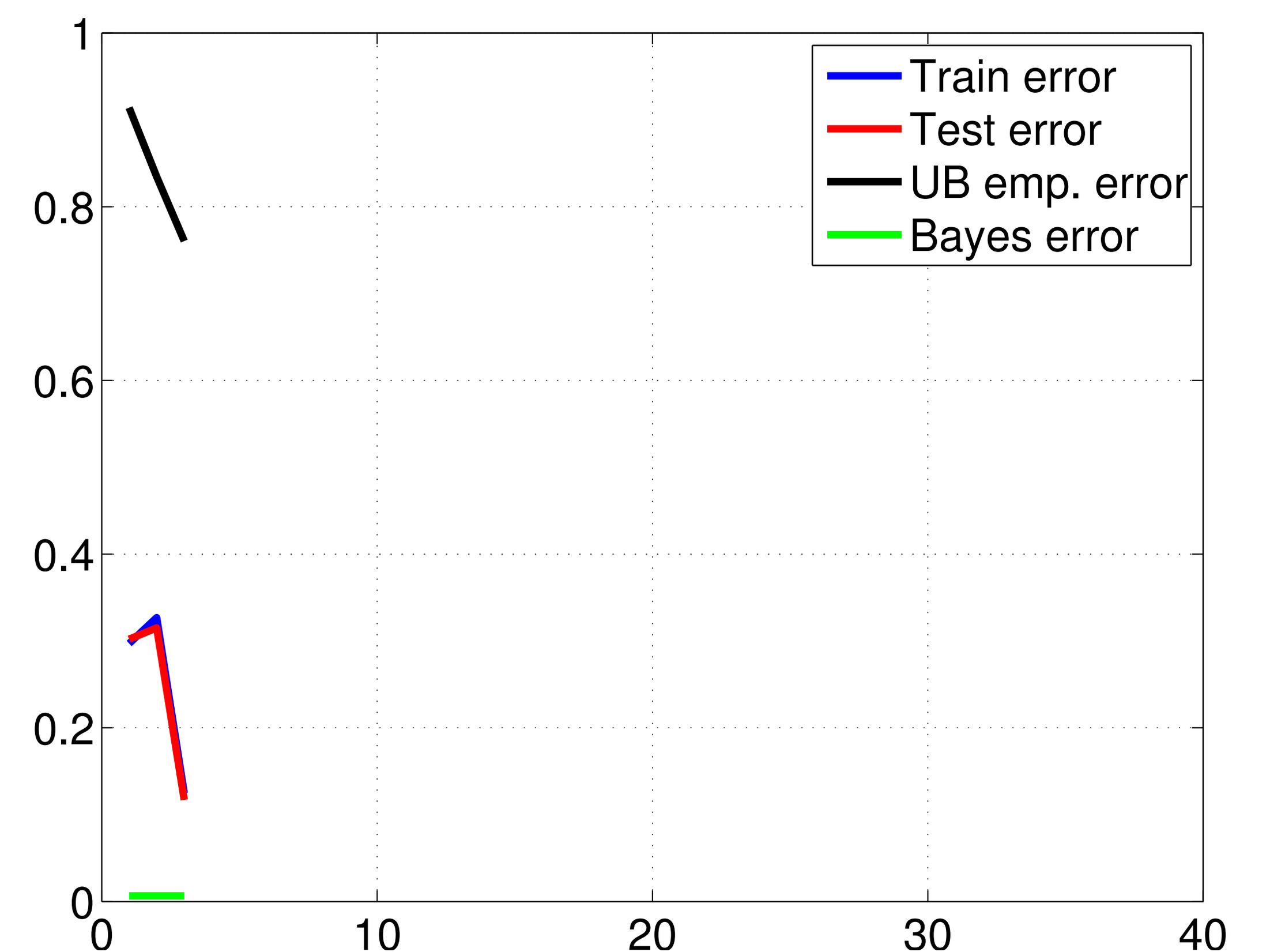


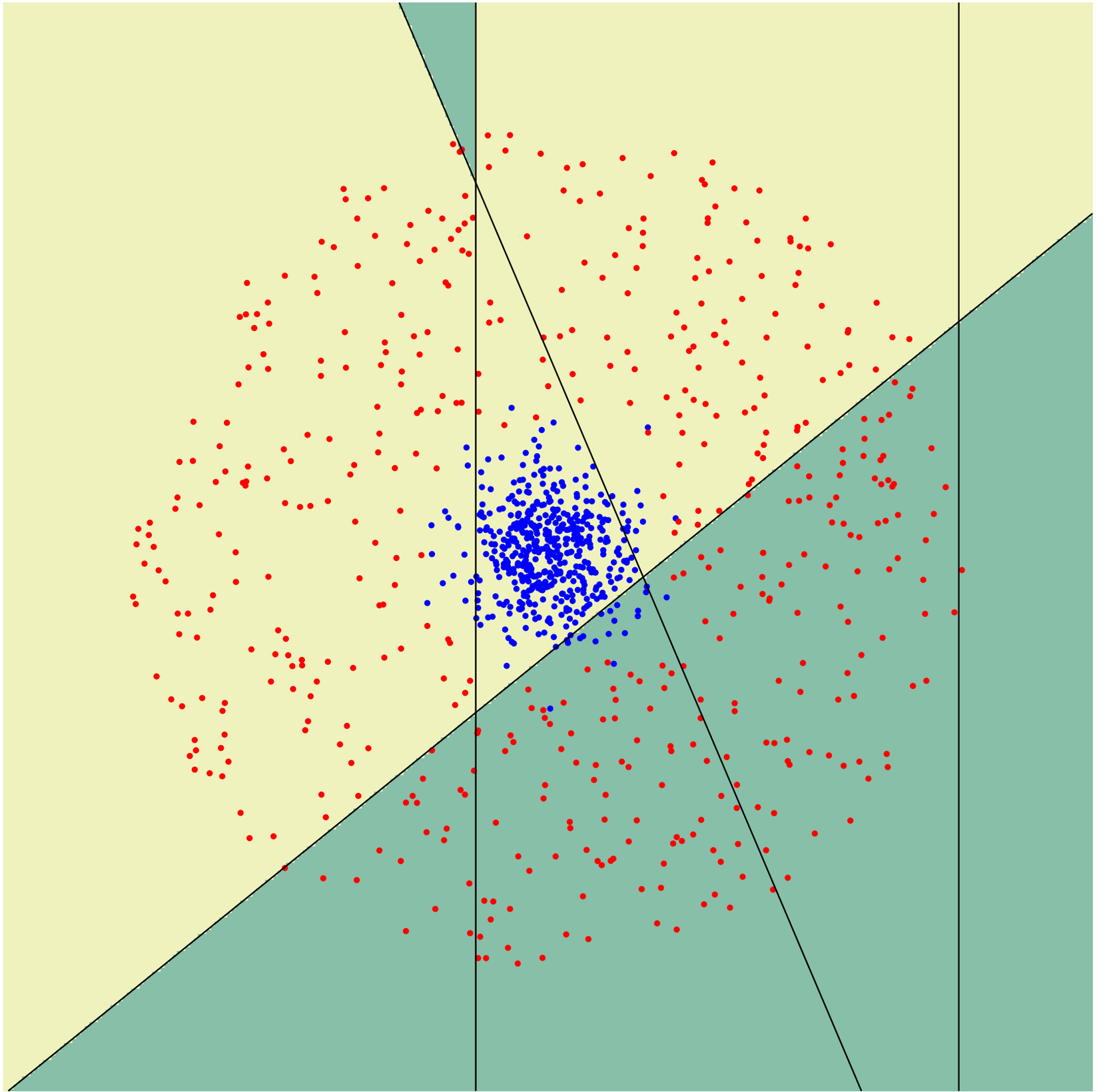

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

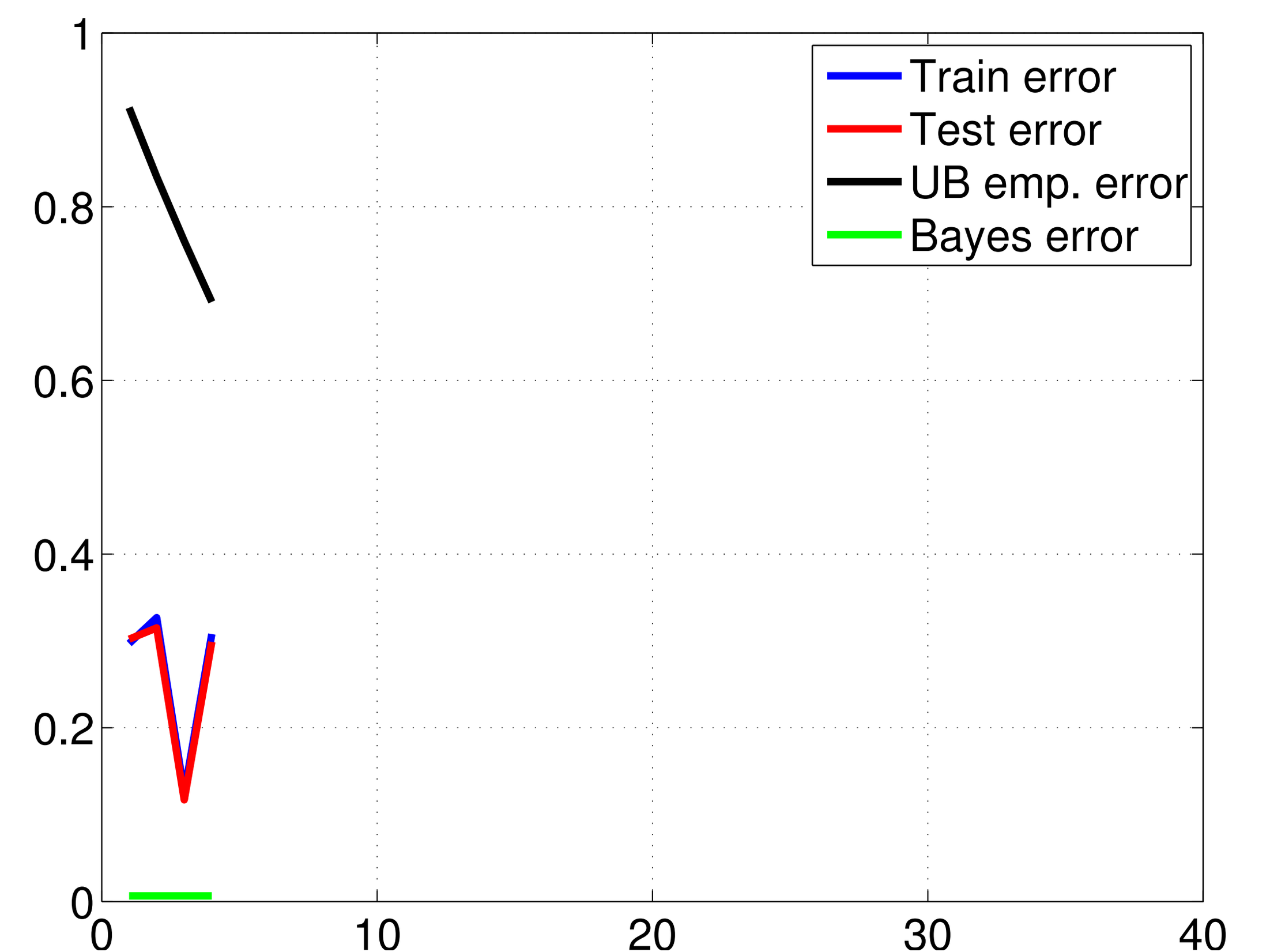


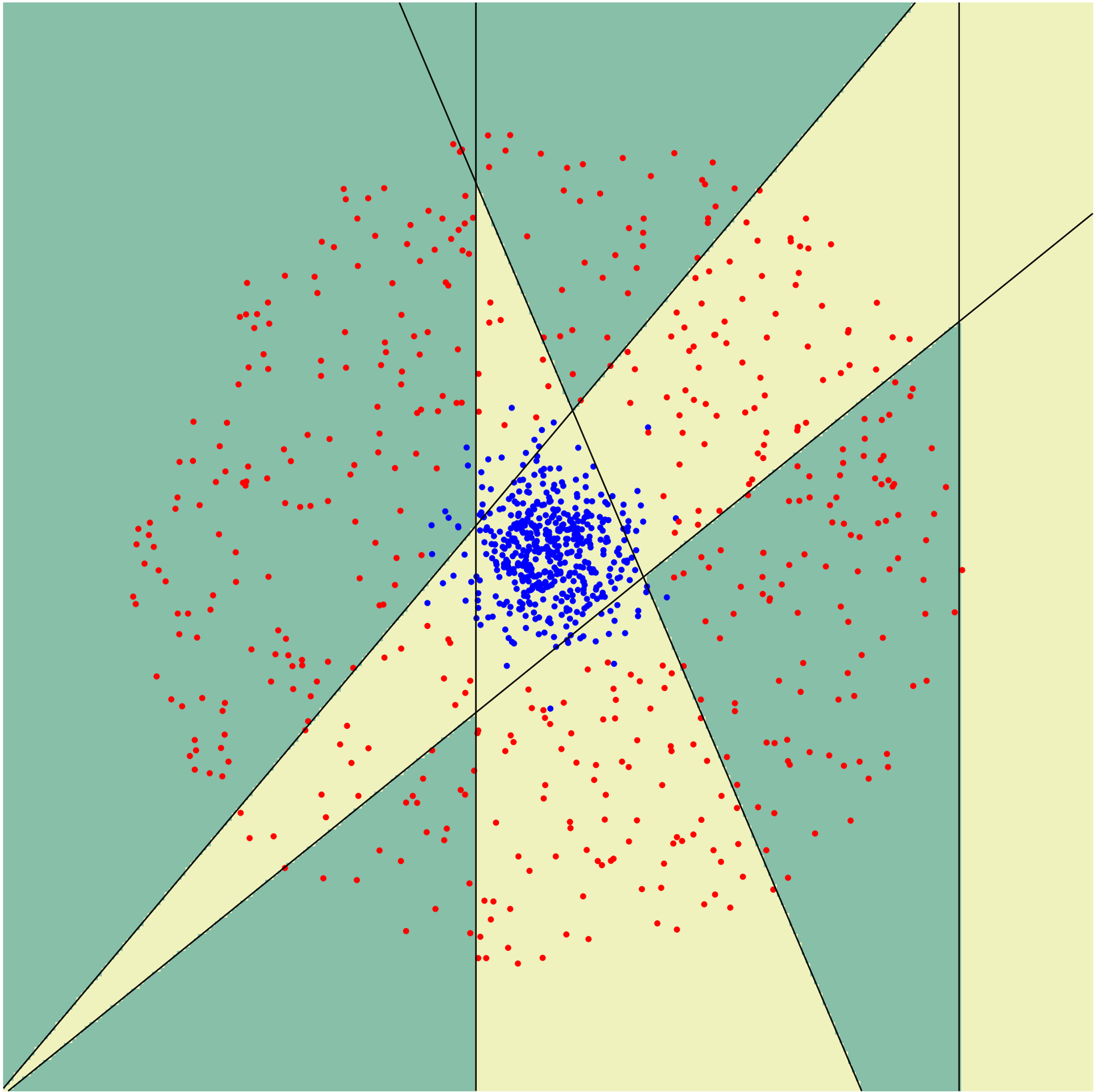

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

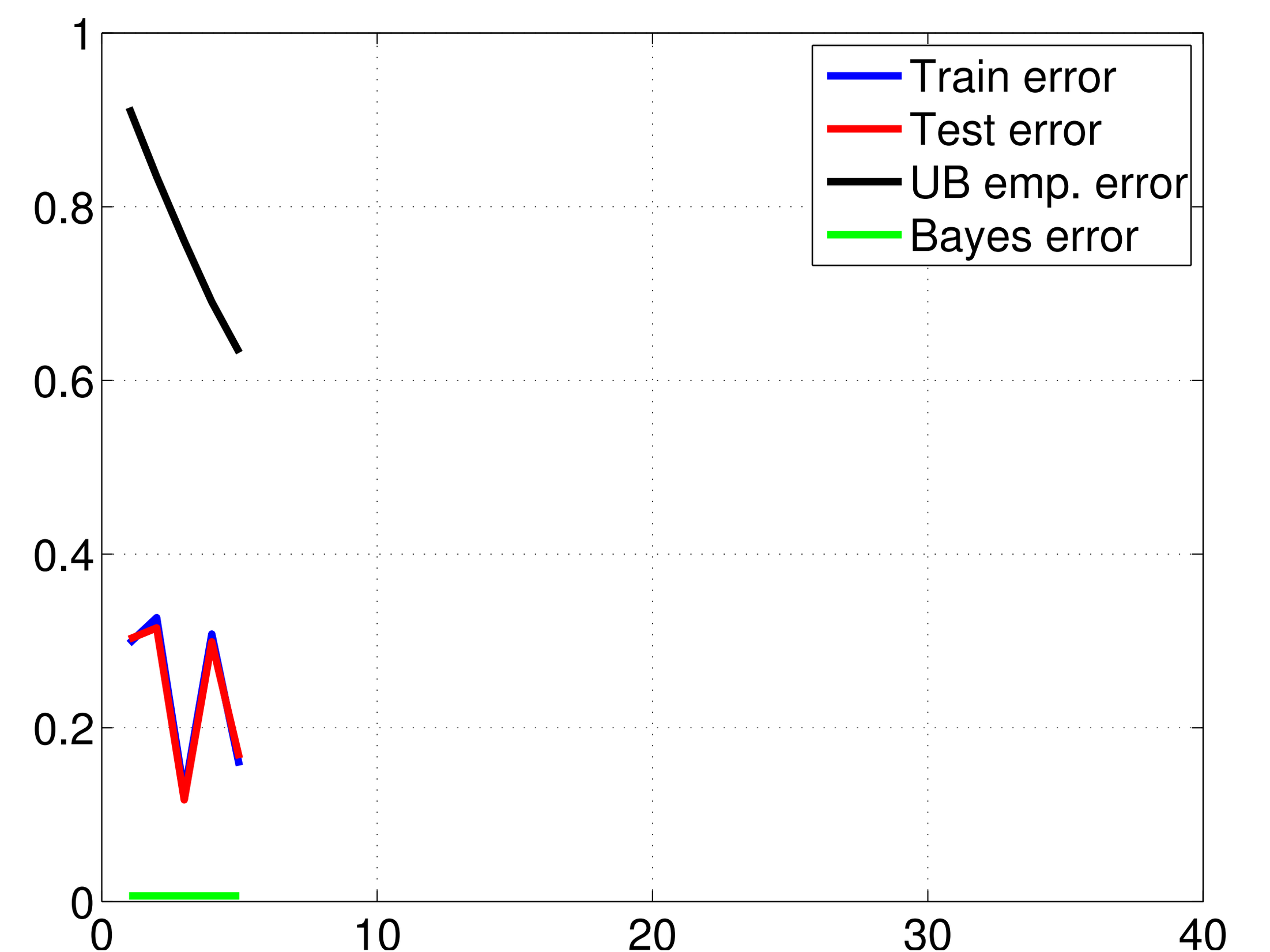


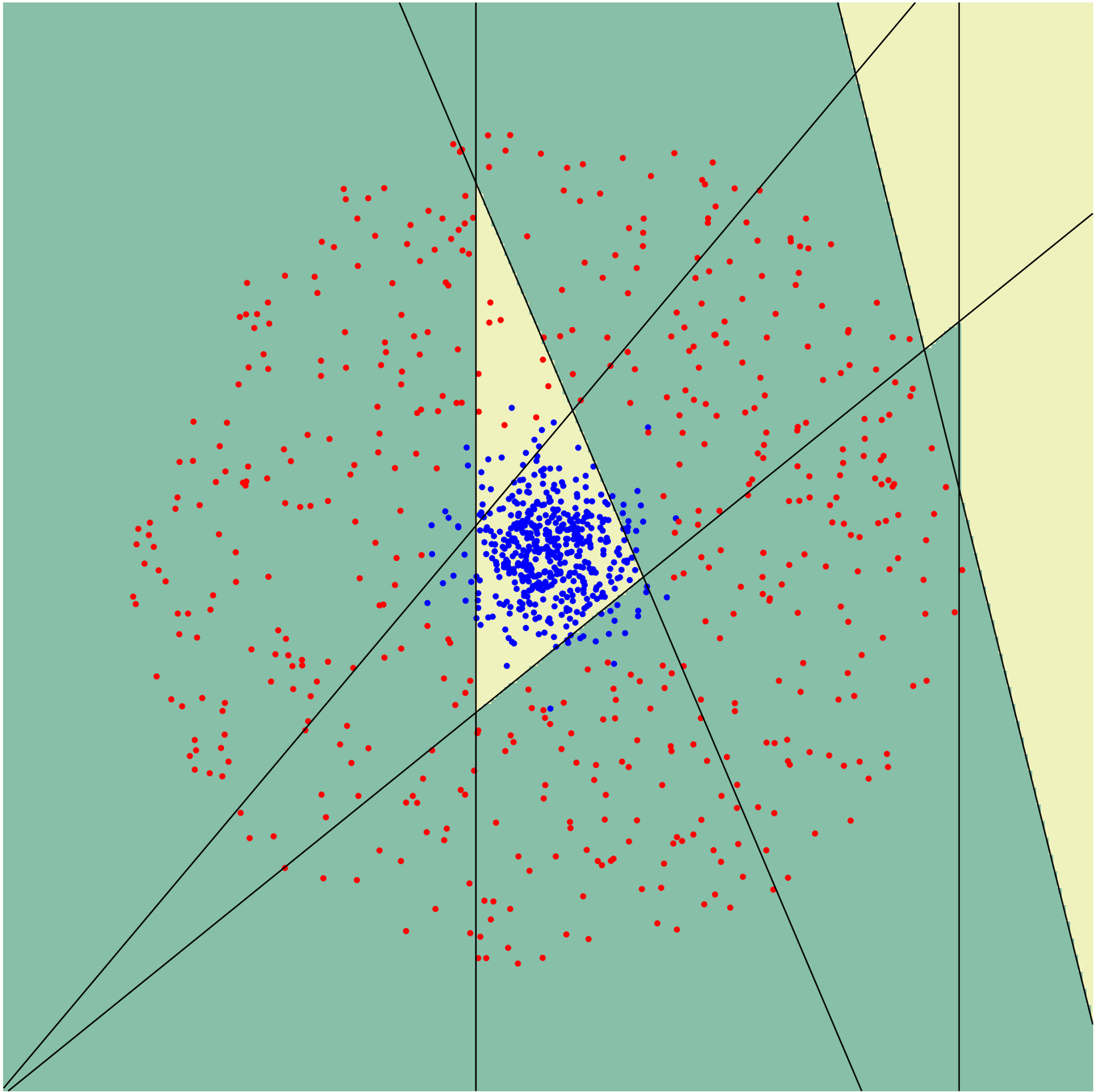

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

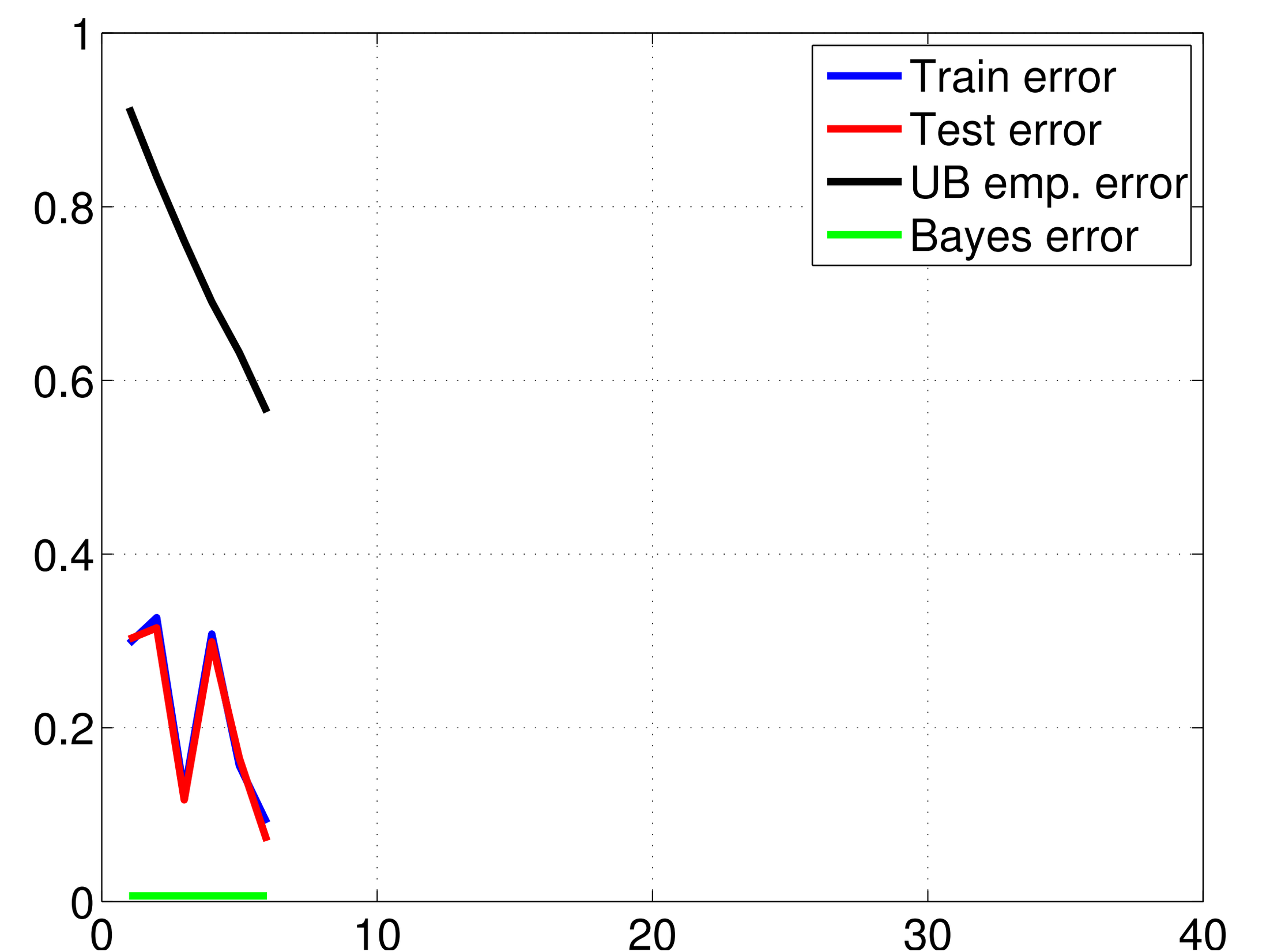


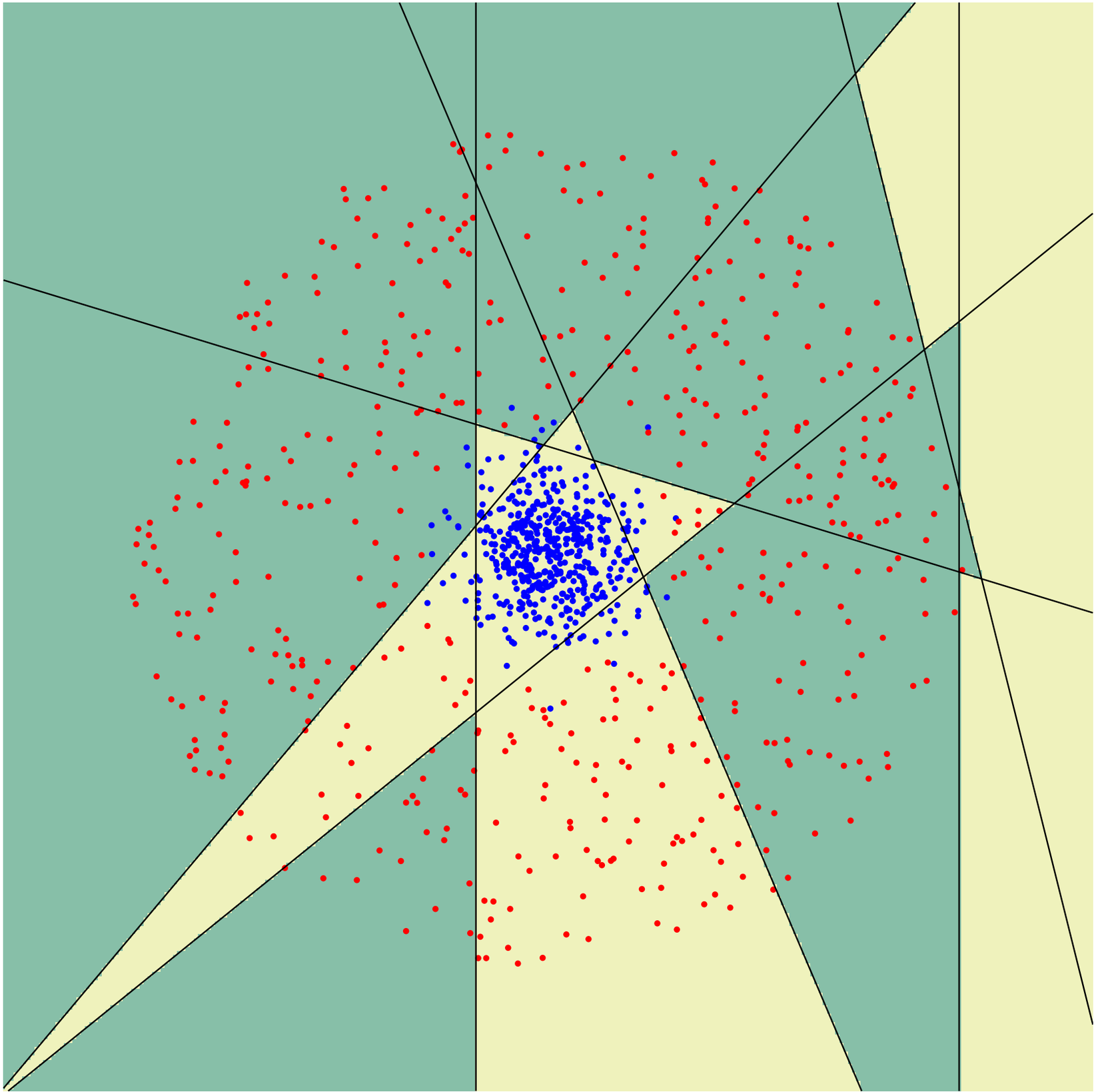

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

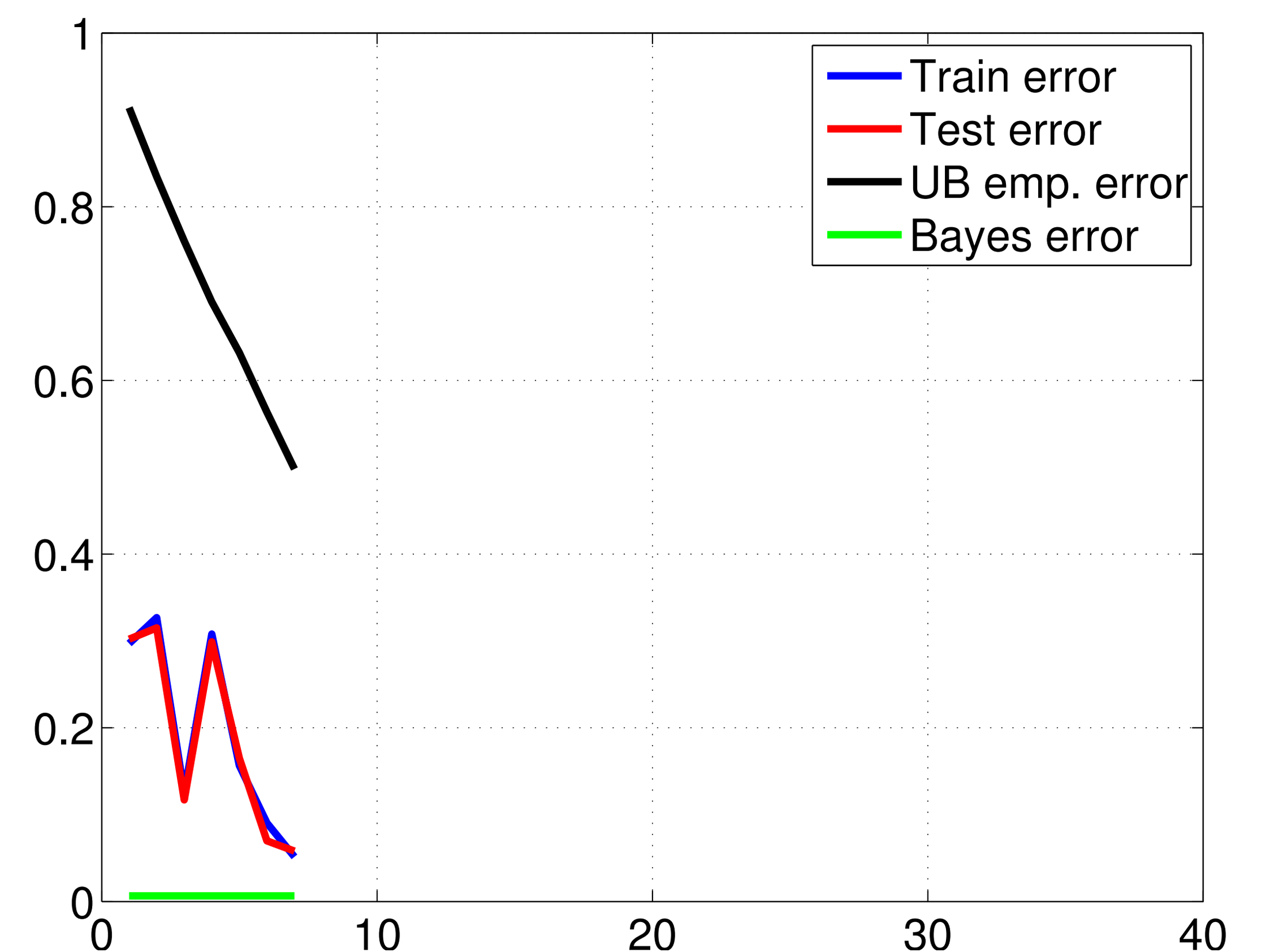


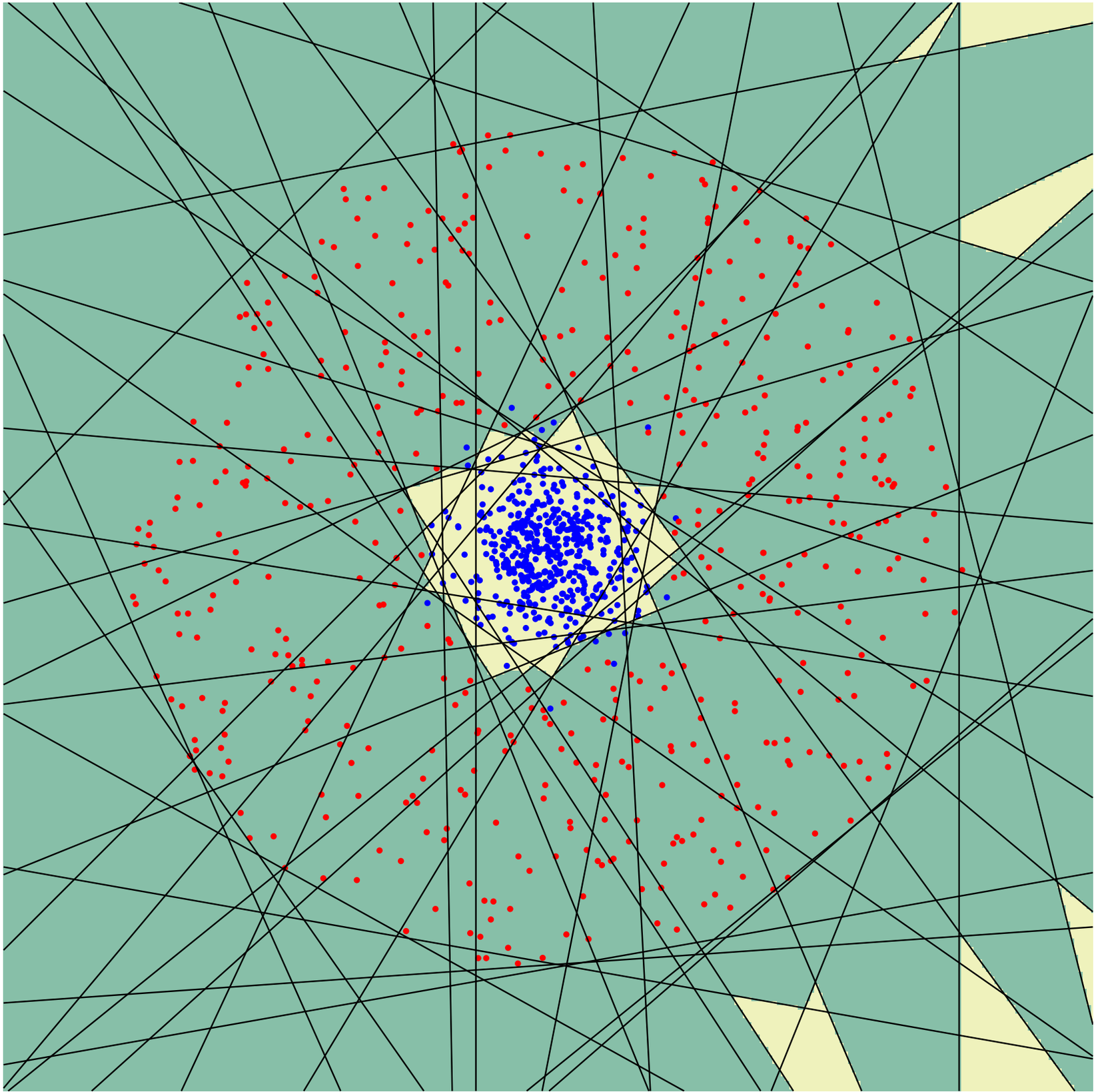

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

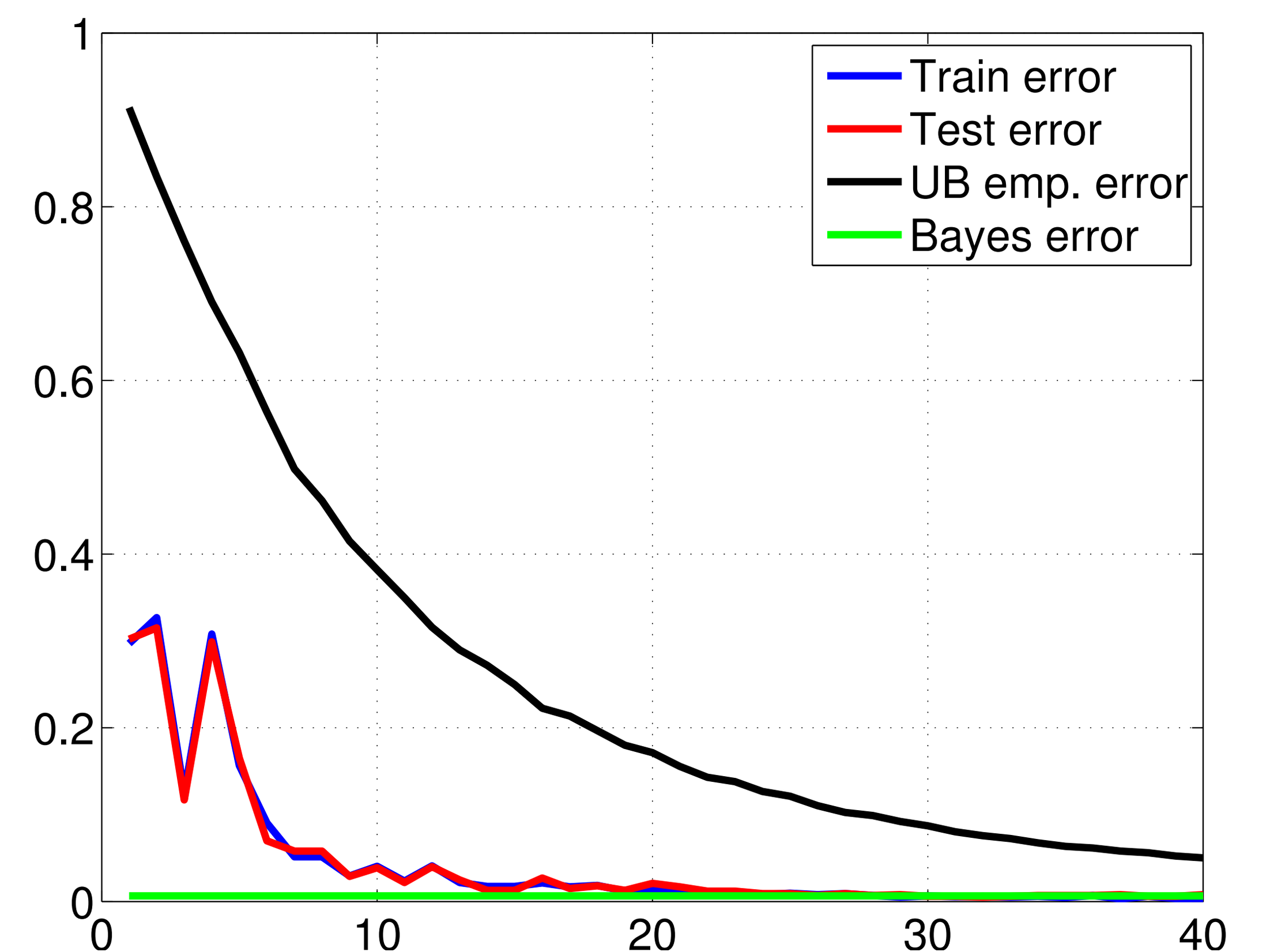


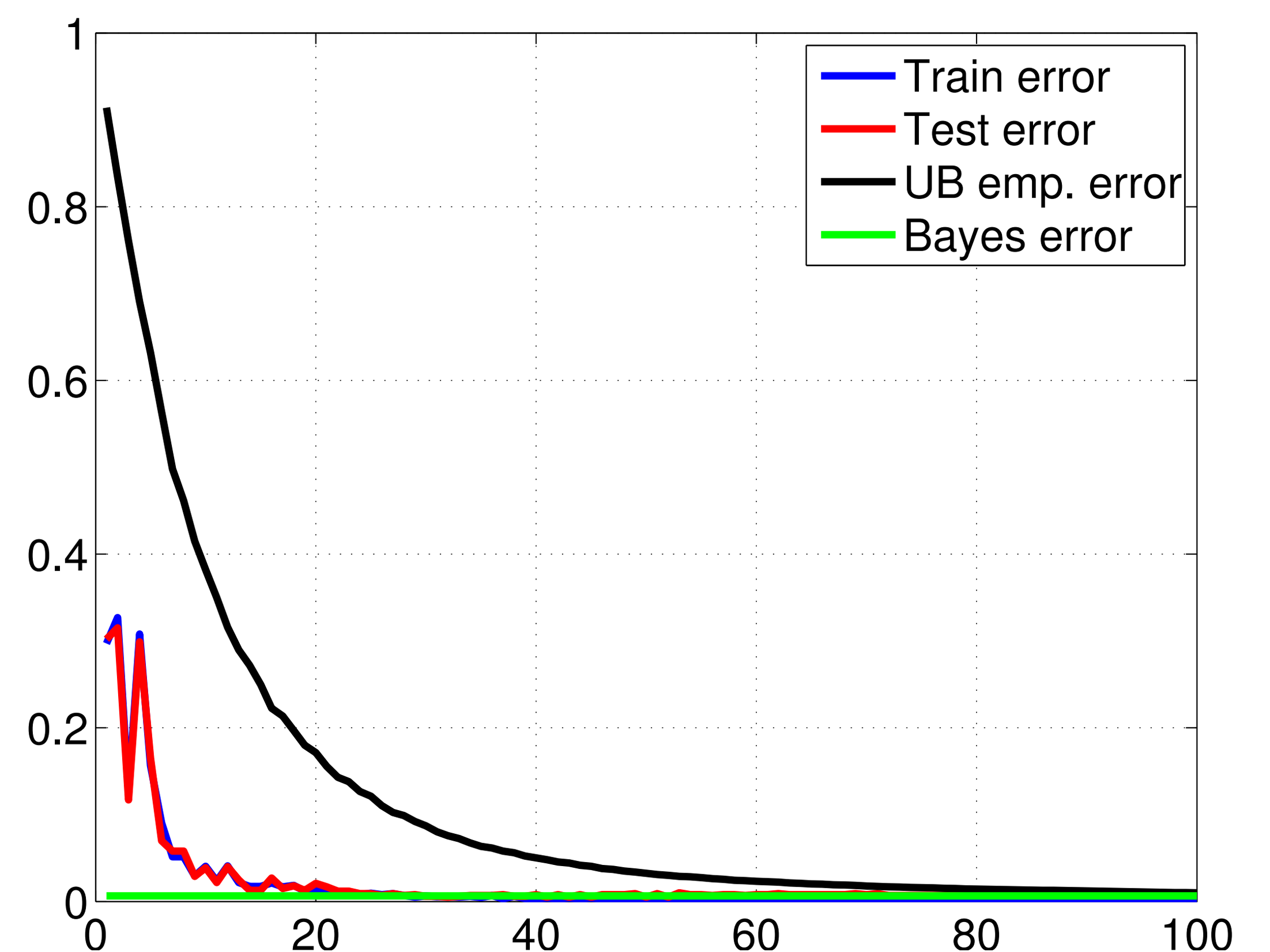

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

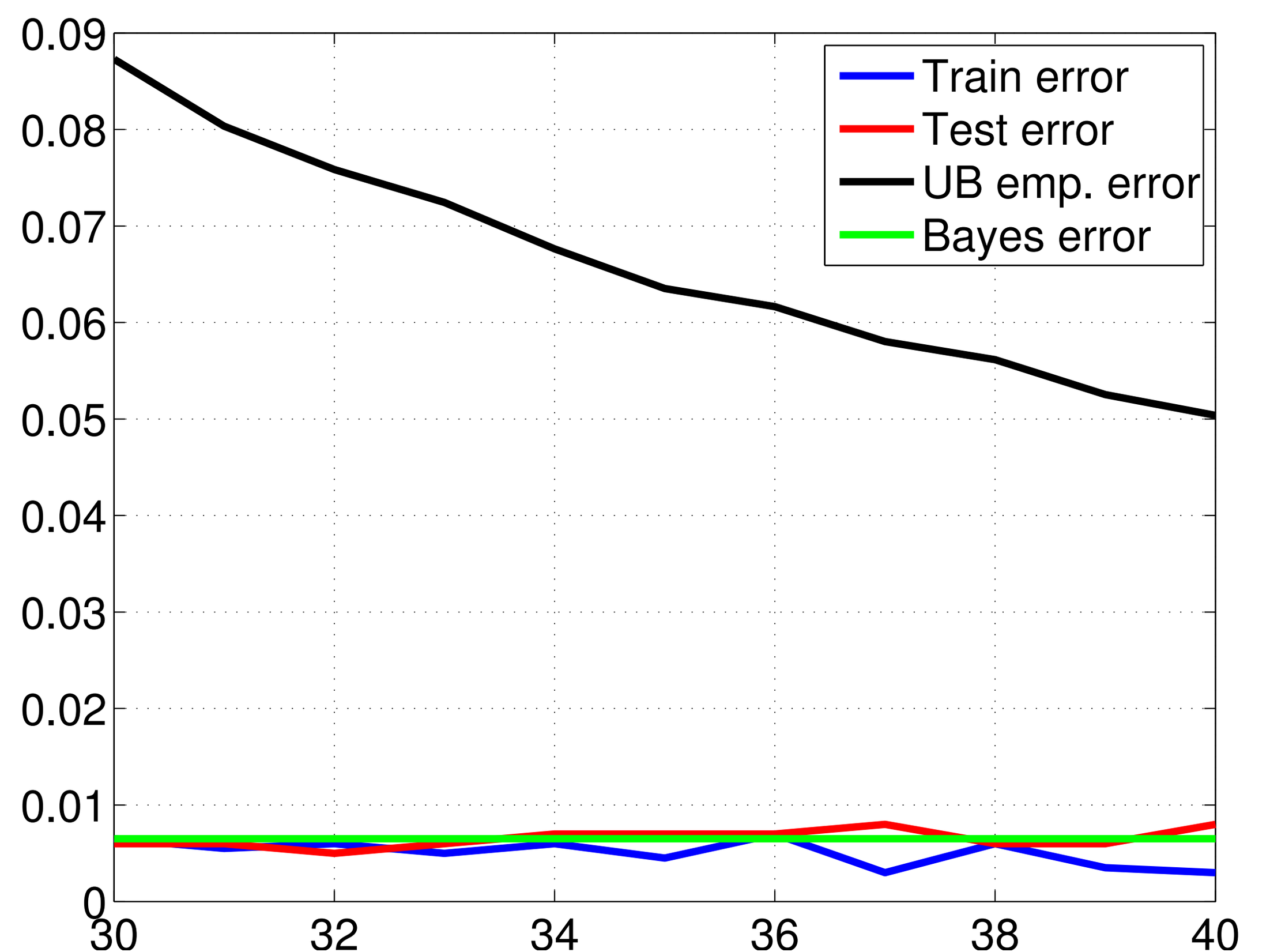


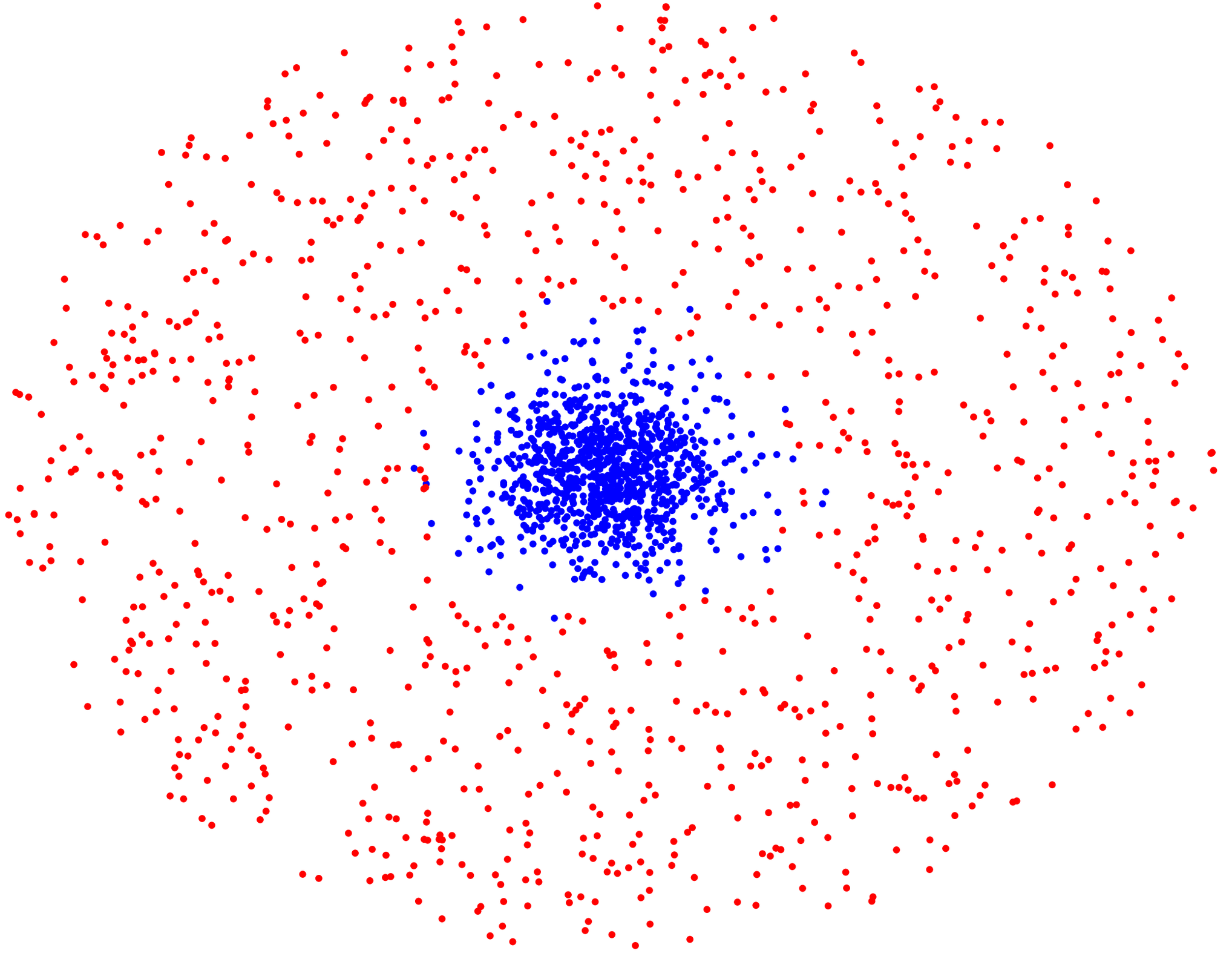

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

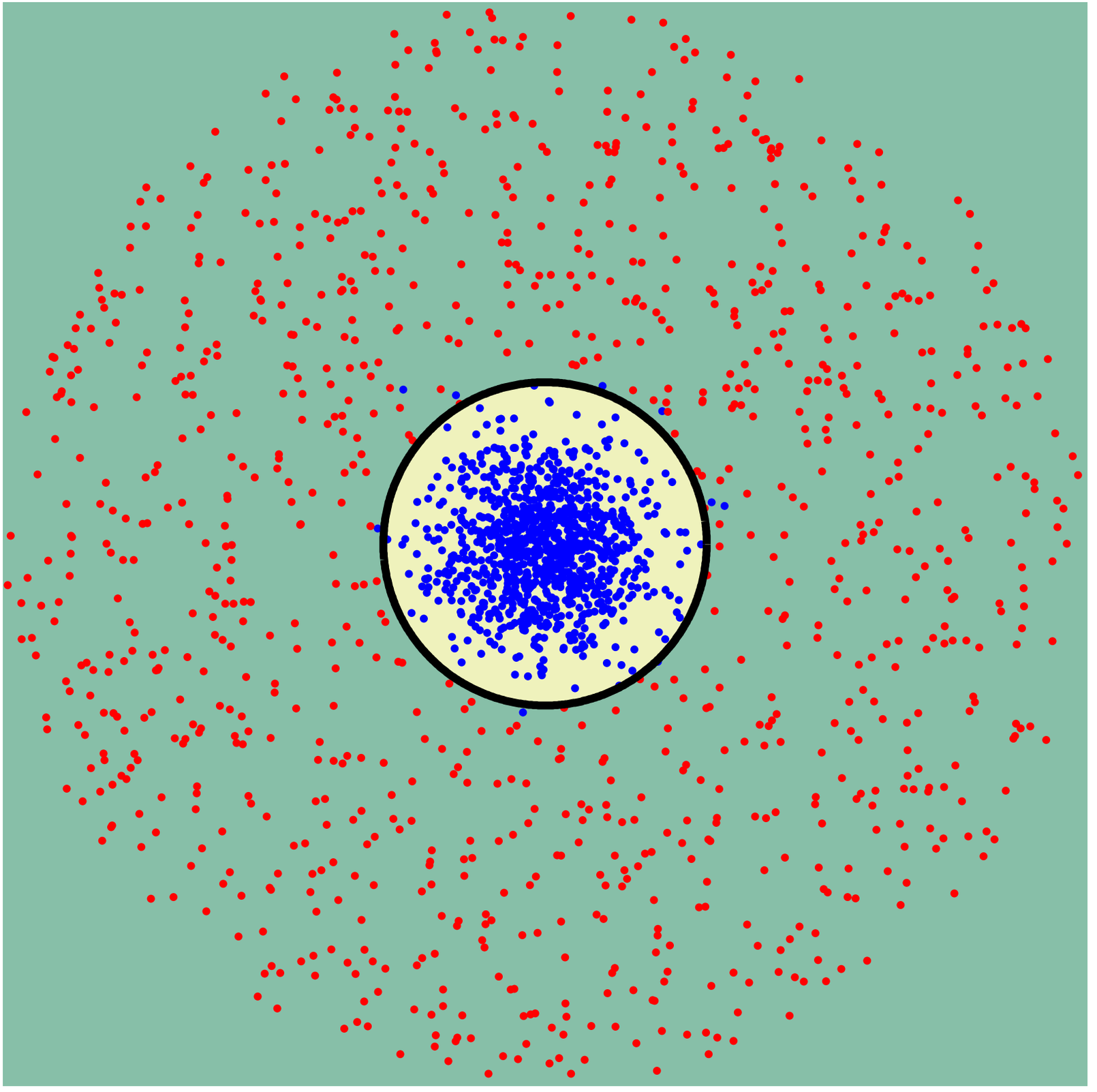


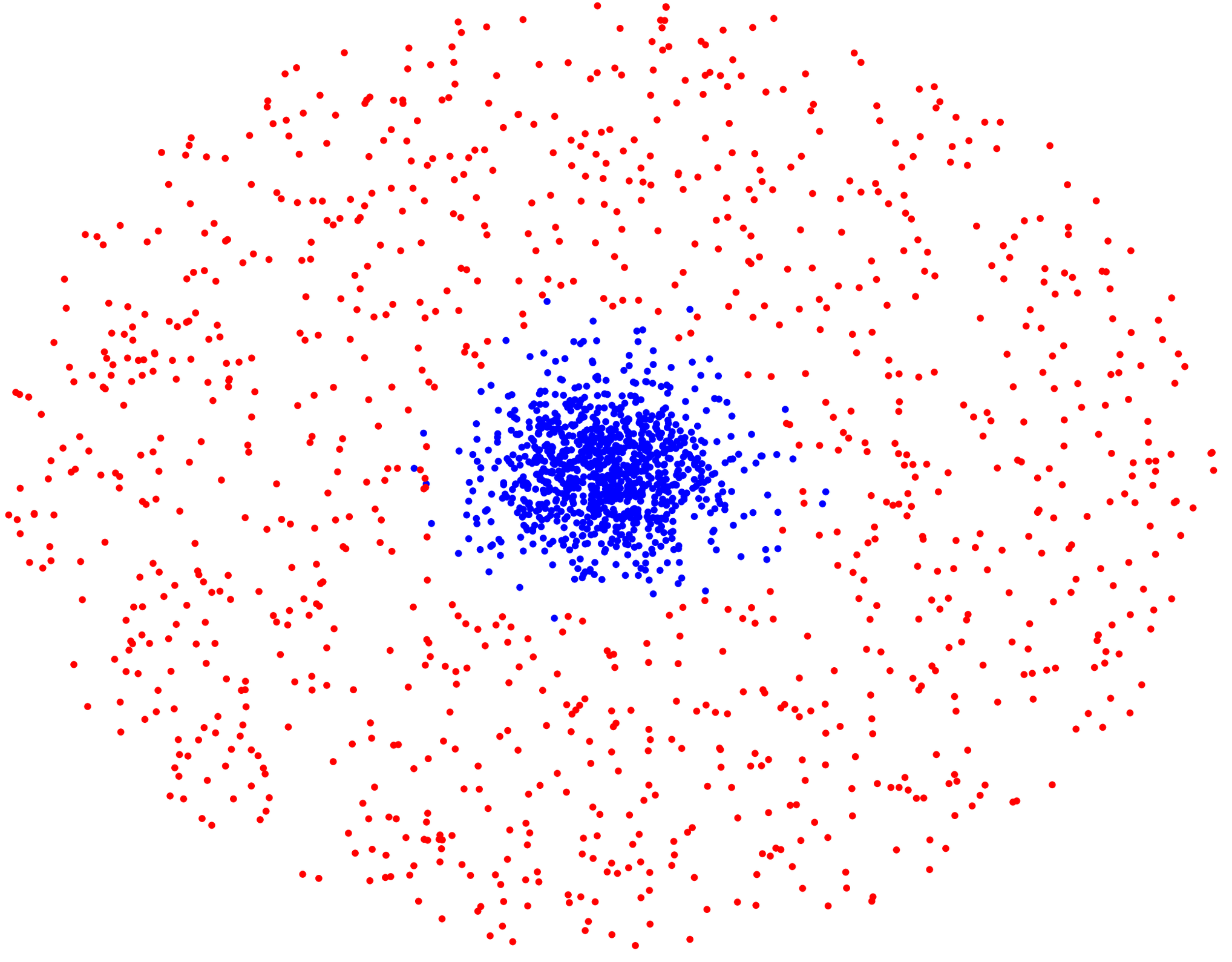

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

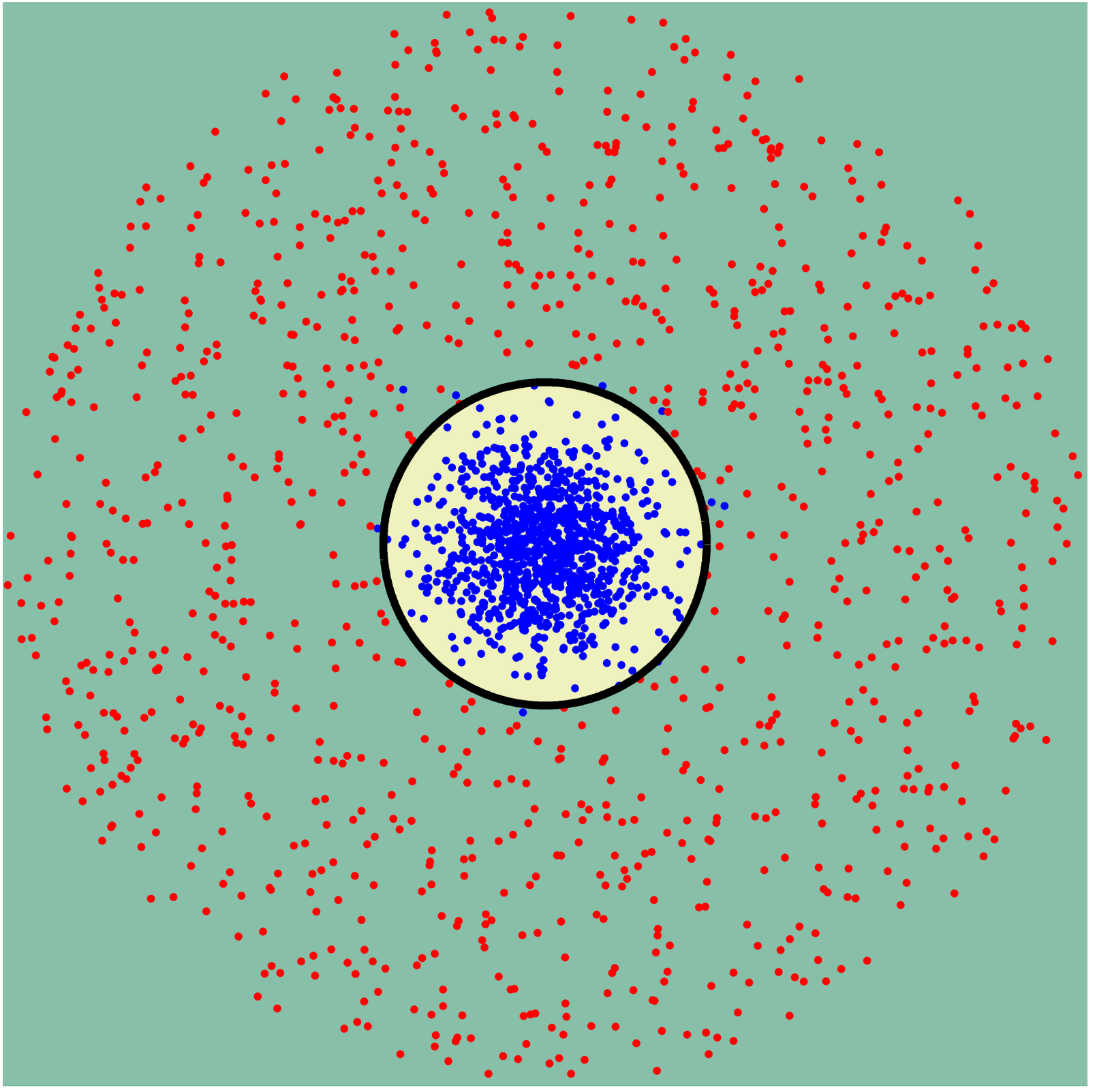

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100

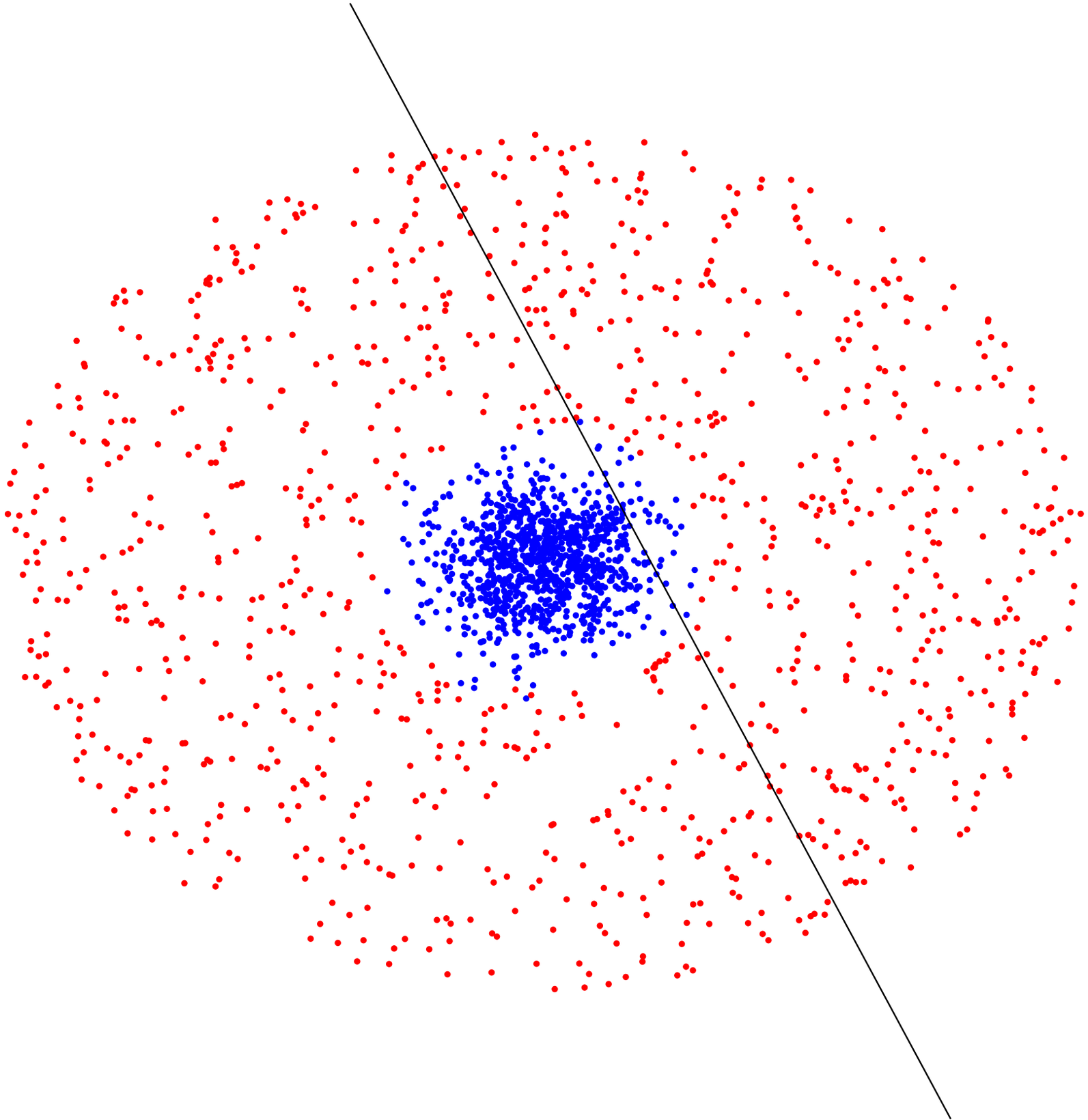

0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
30
32
34
36
38
40
Train error
Test error
UB emp. error
Bayes error

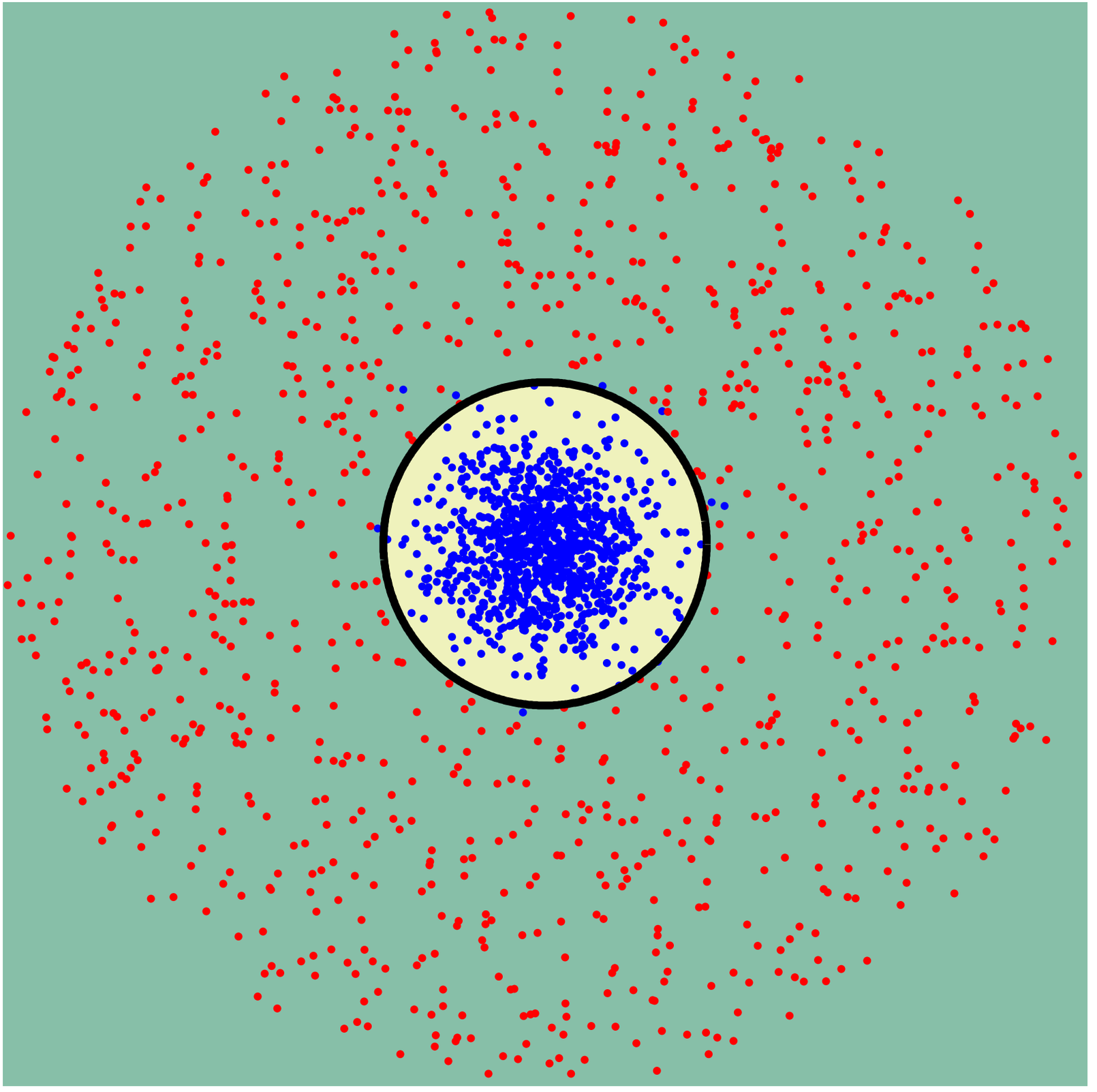


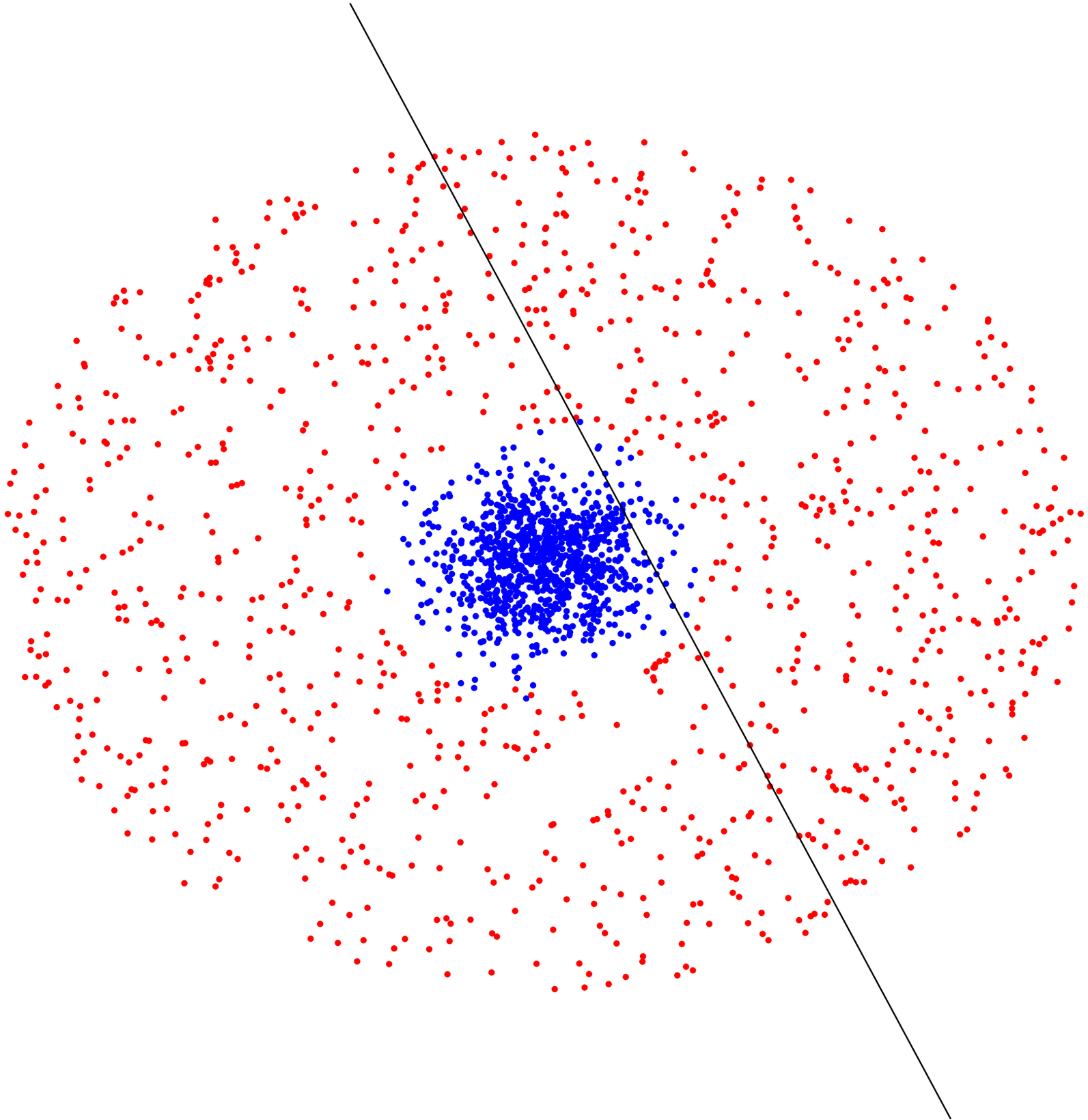


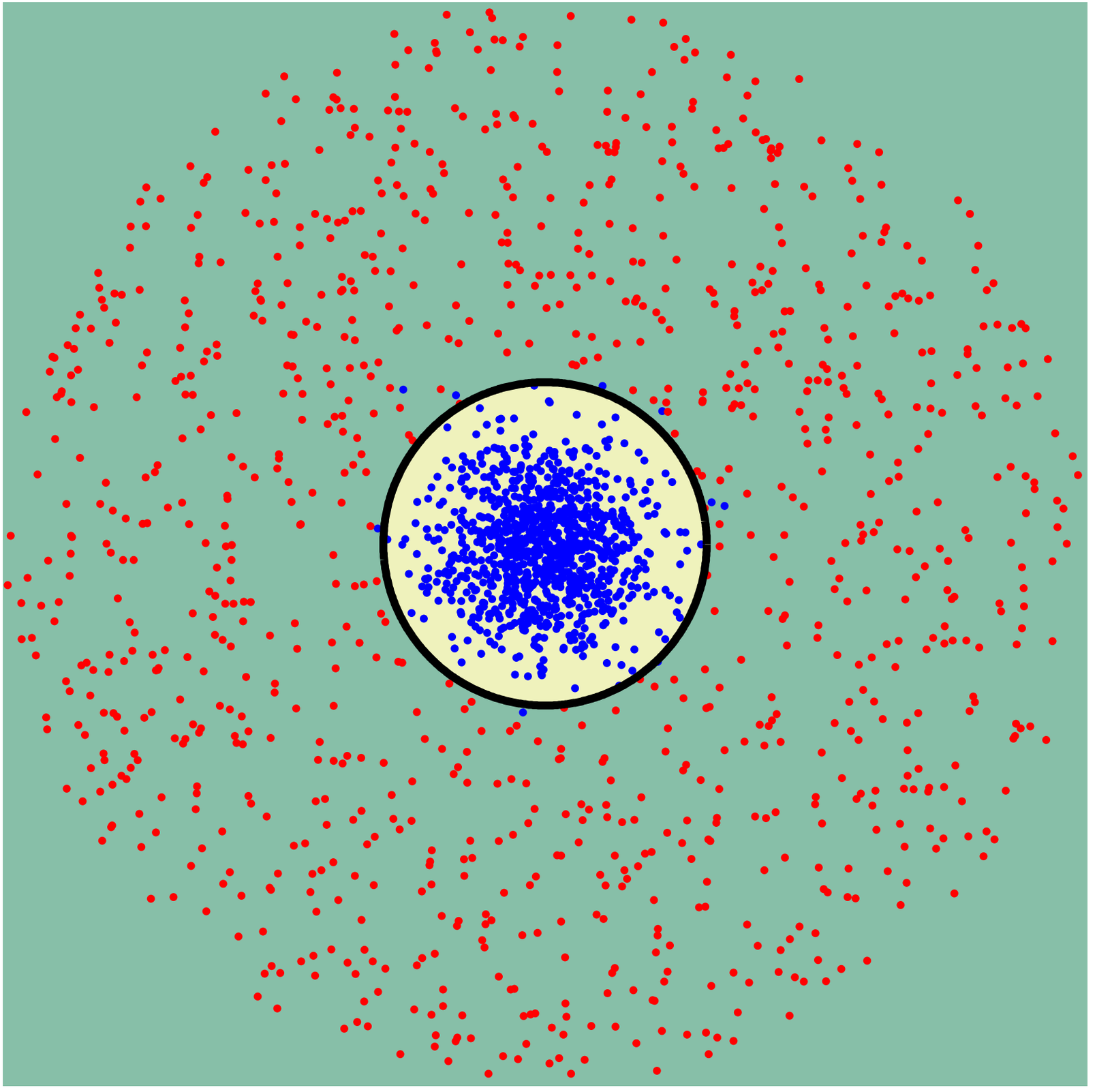


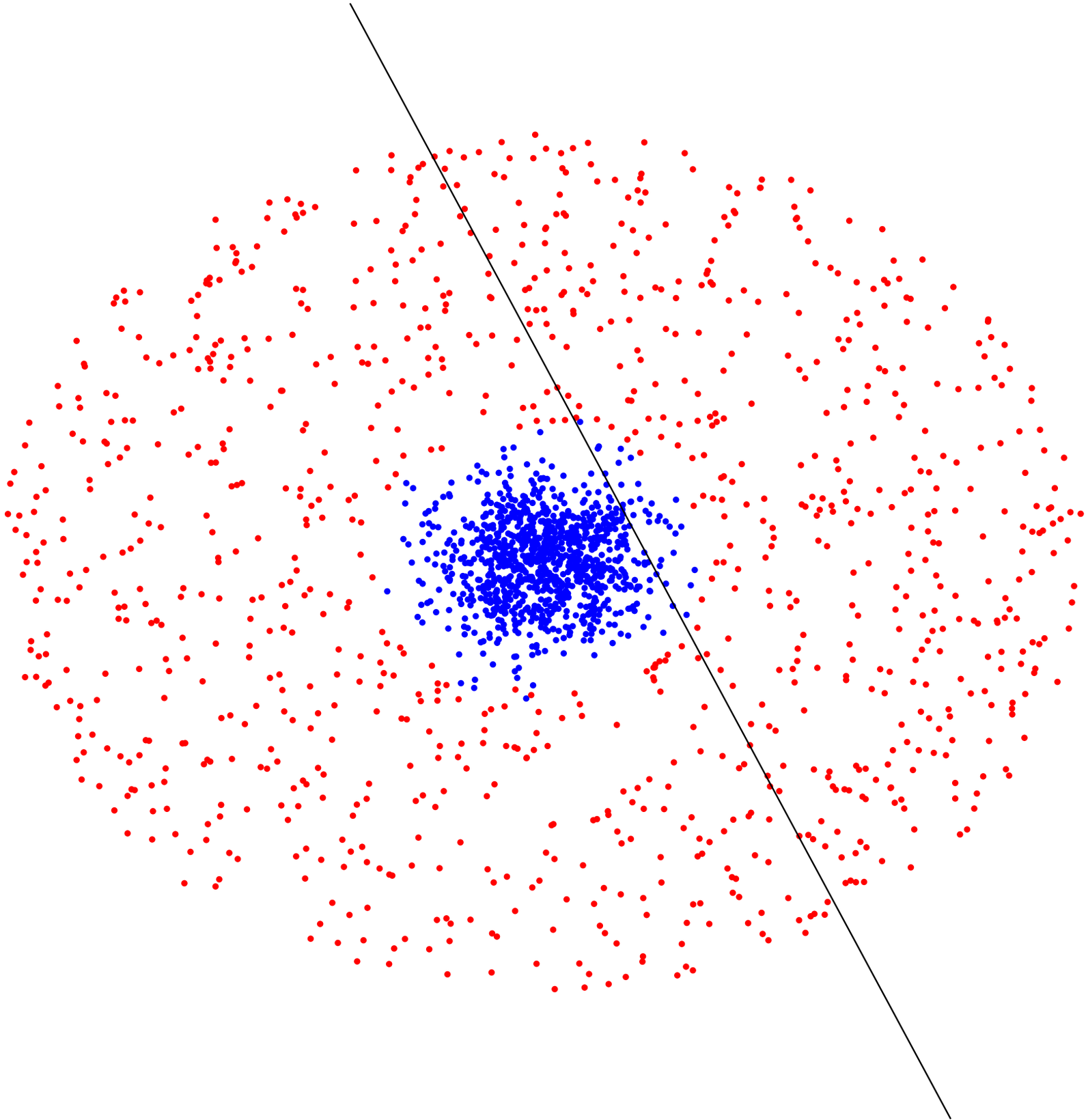


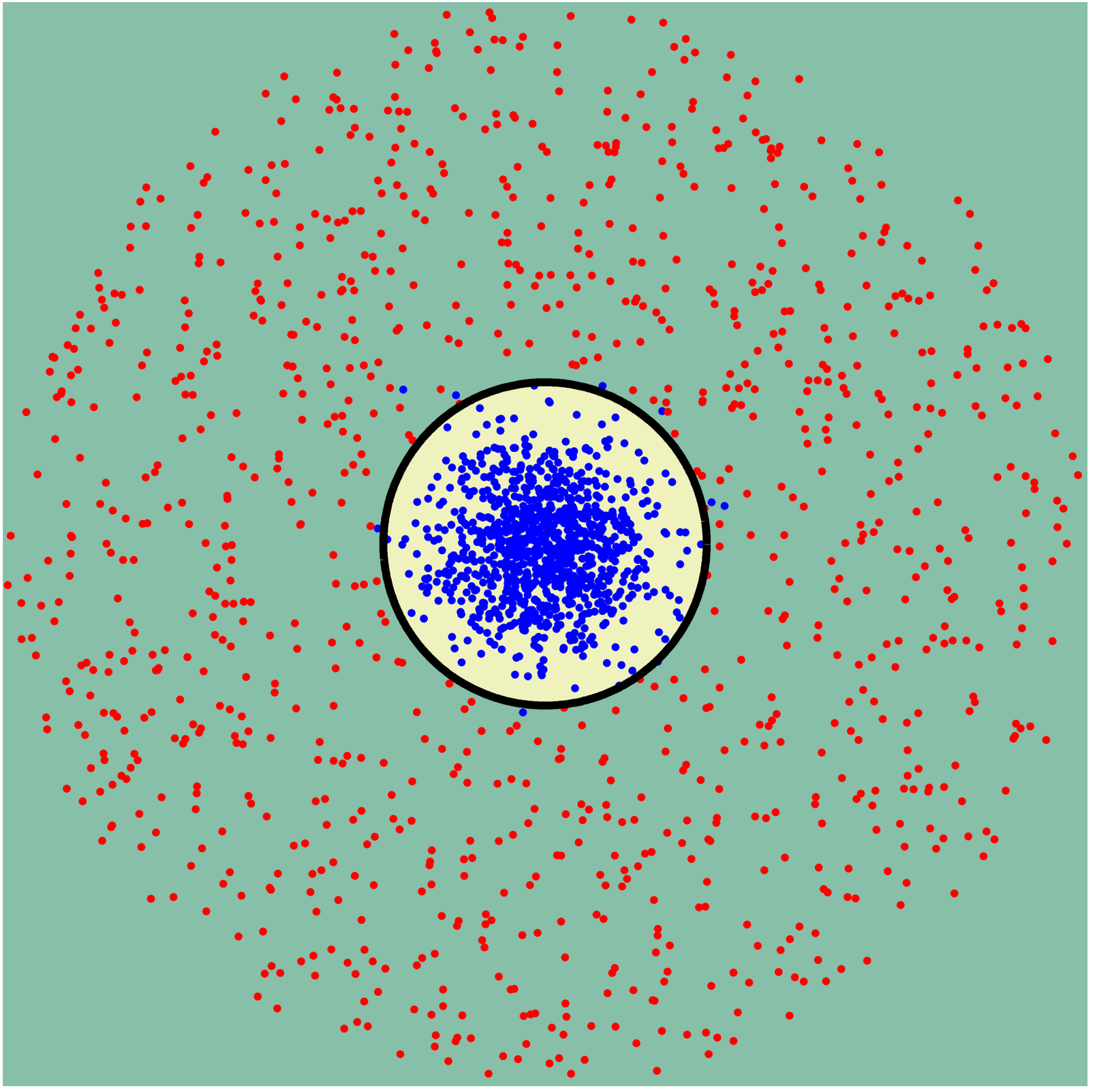


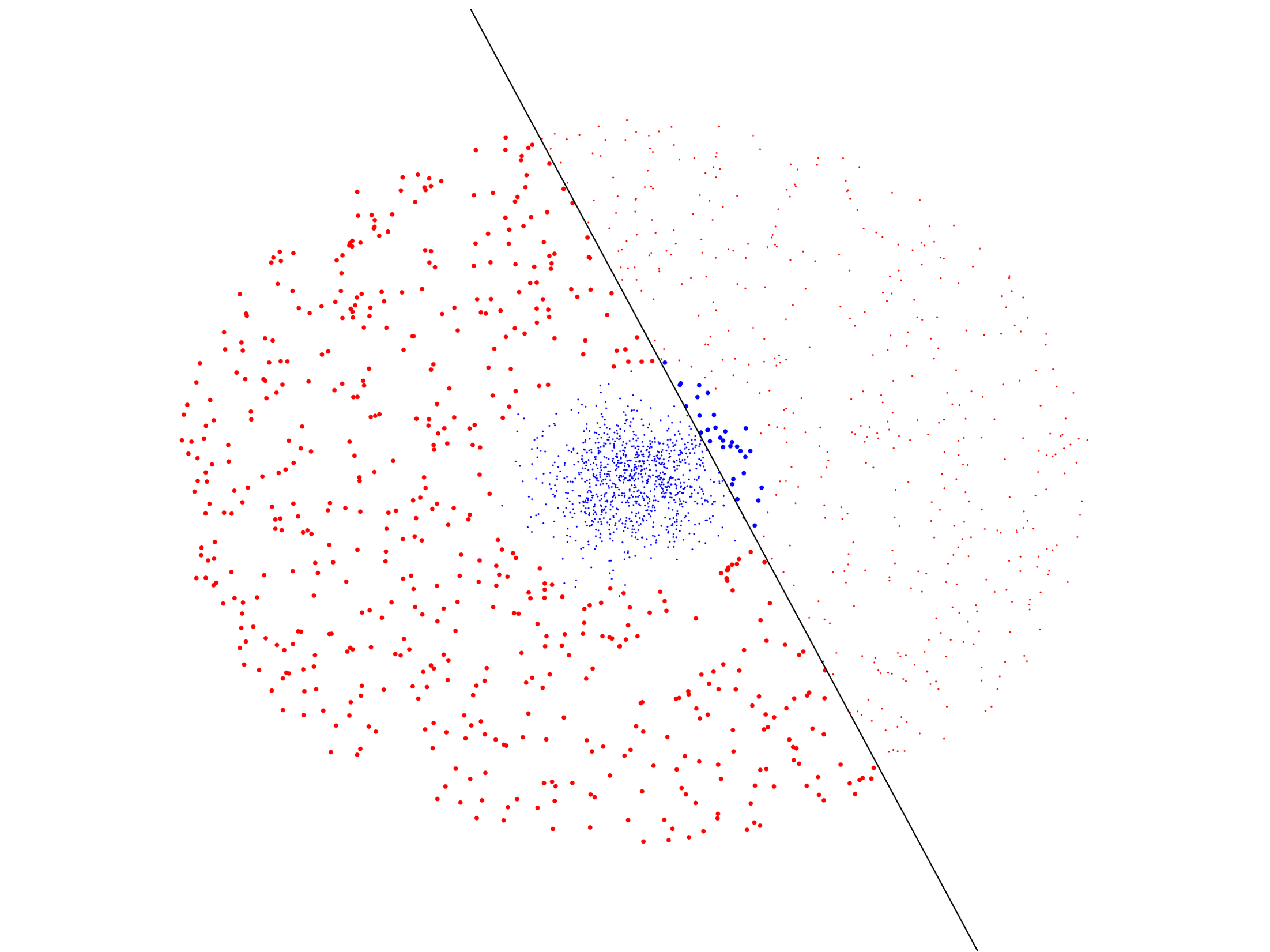


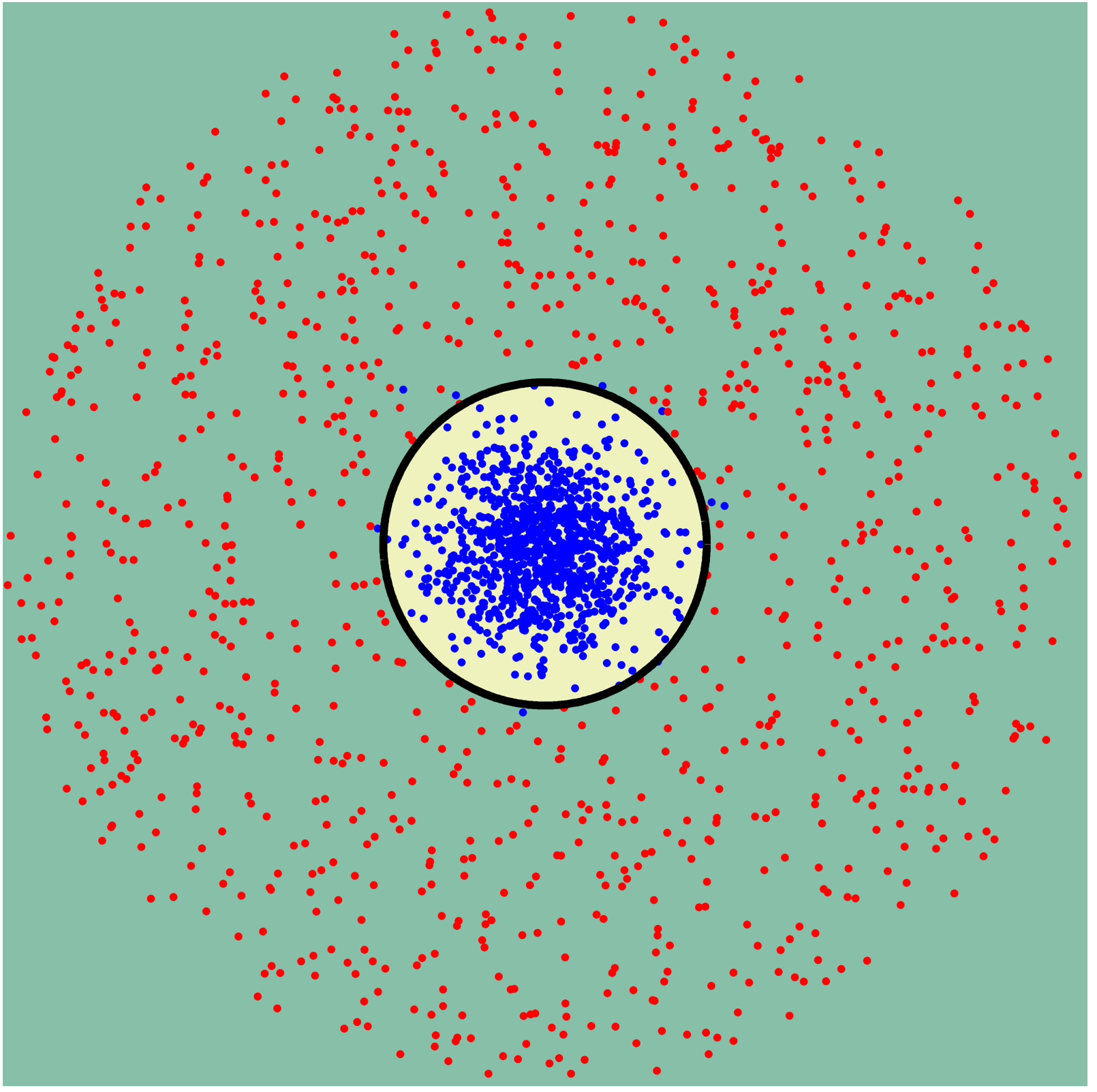


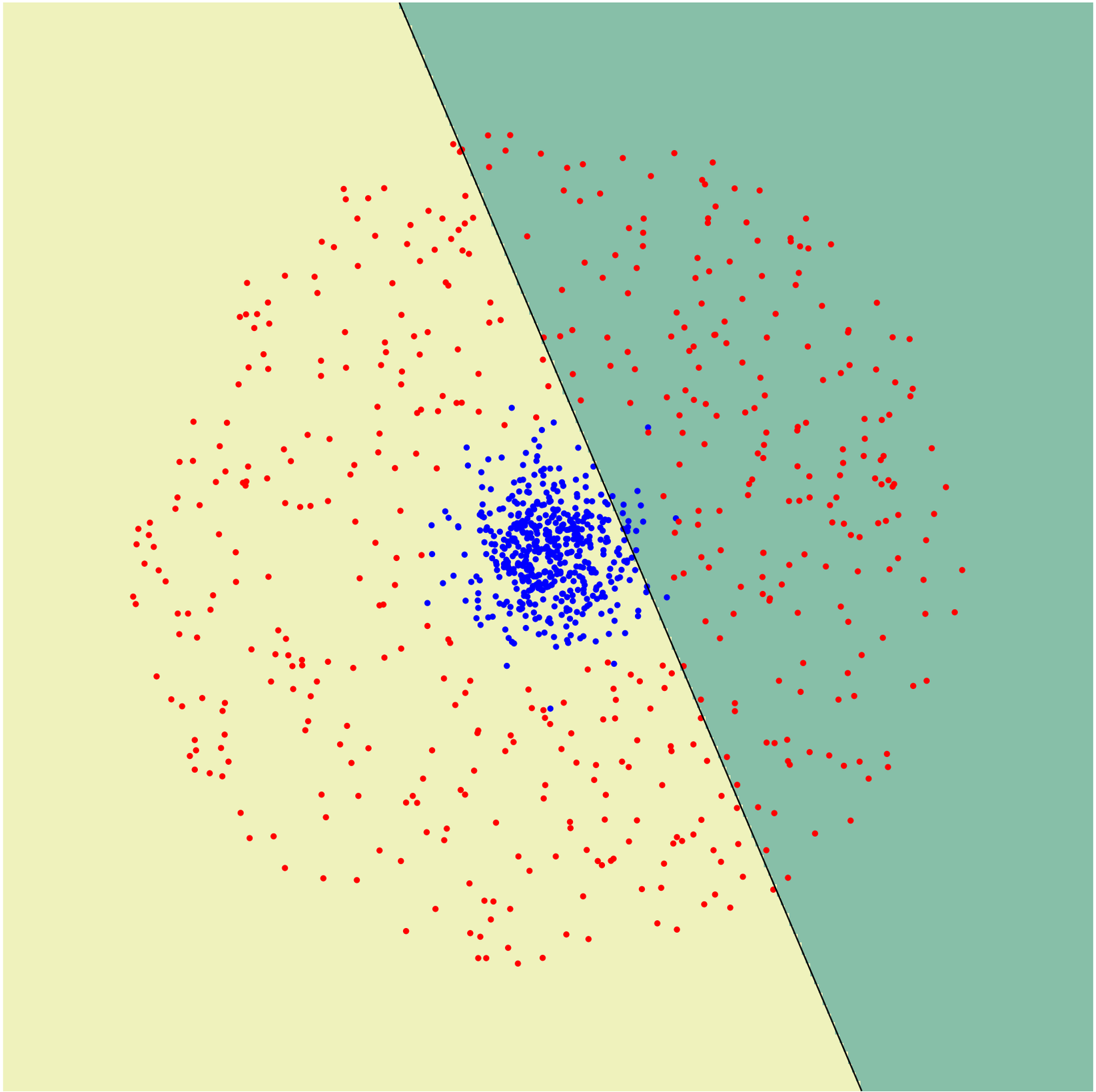


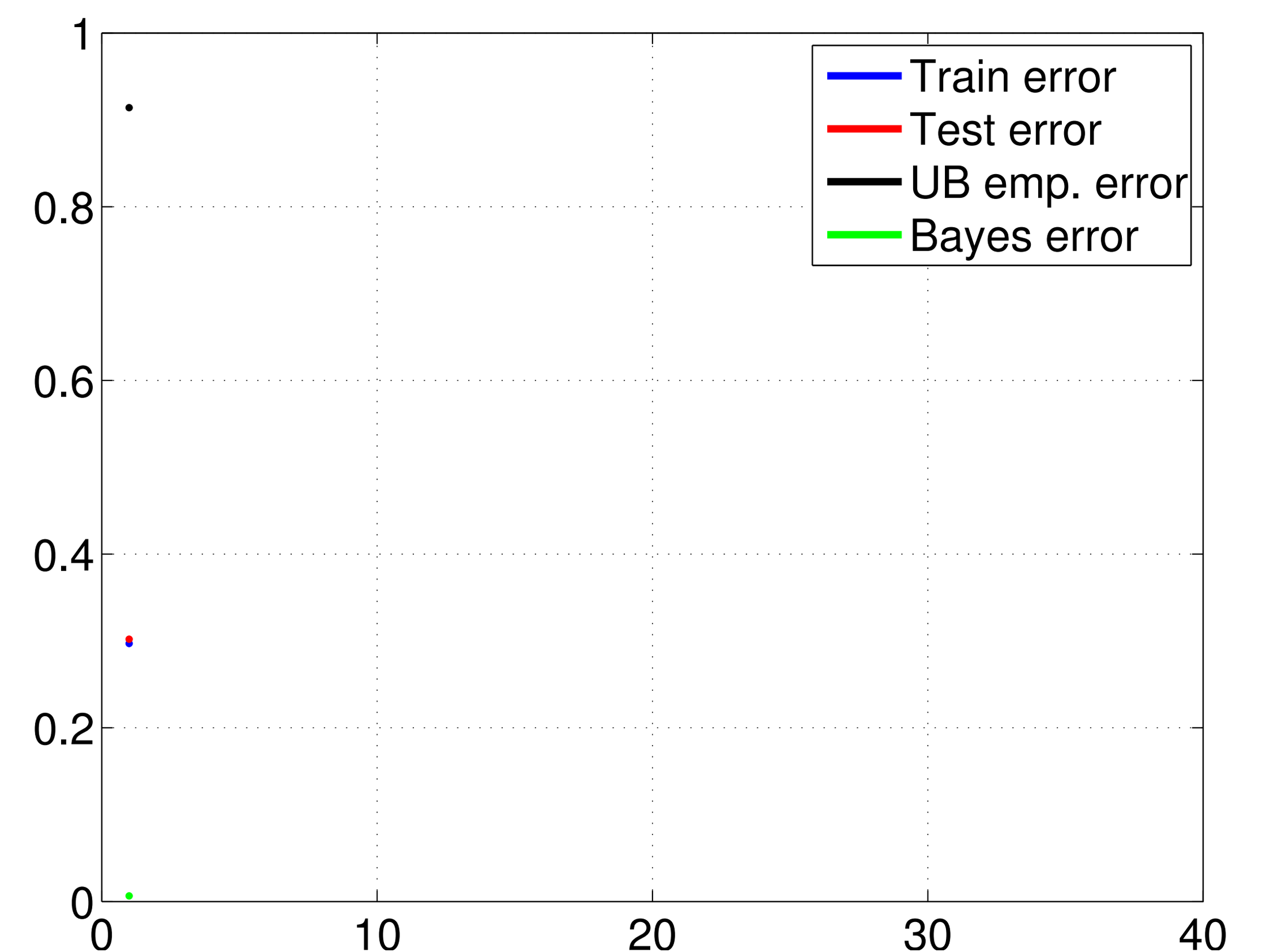


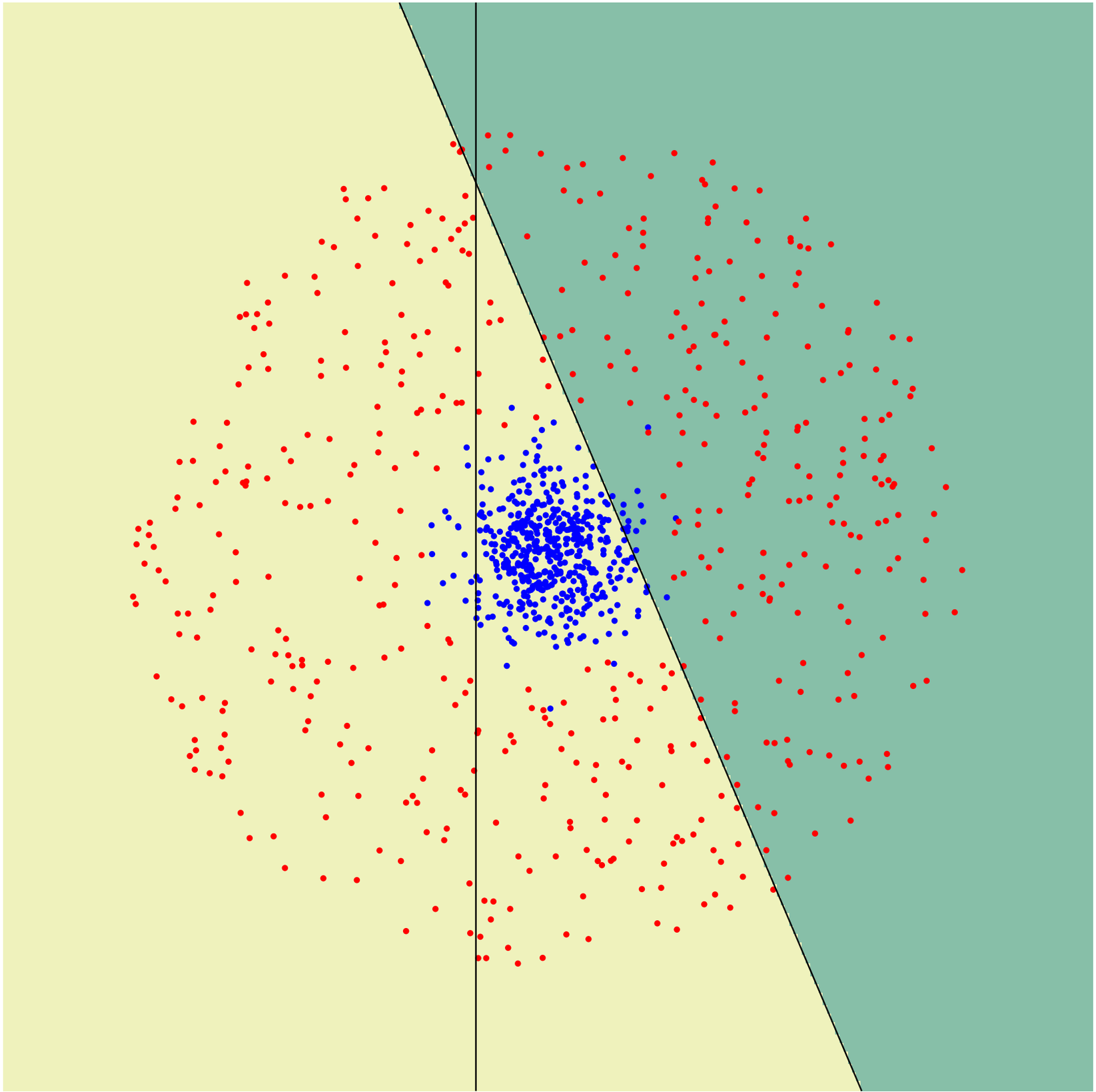


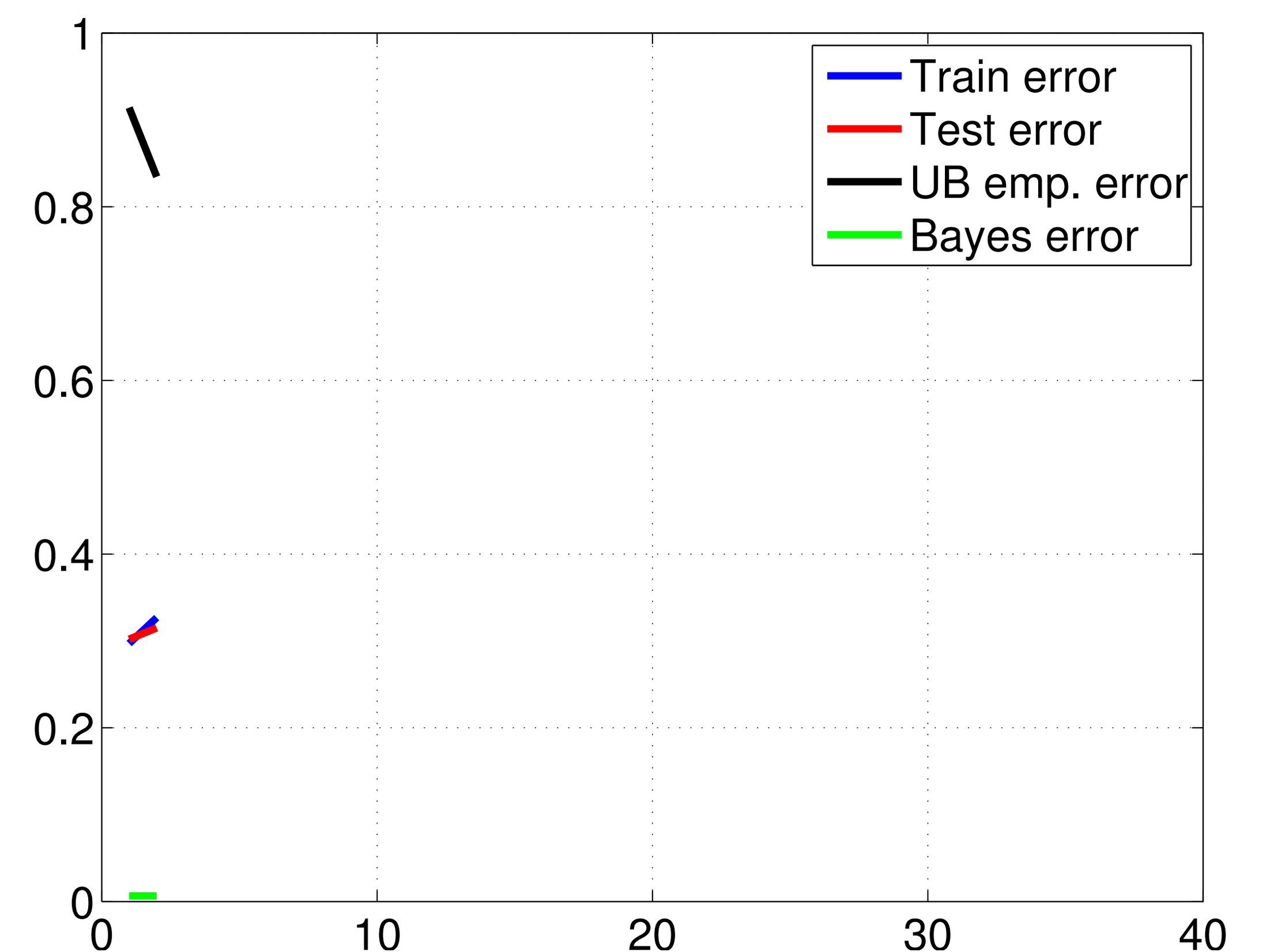


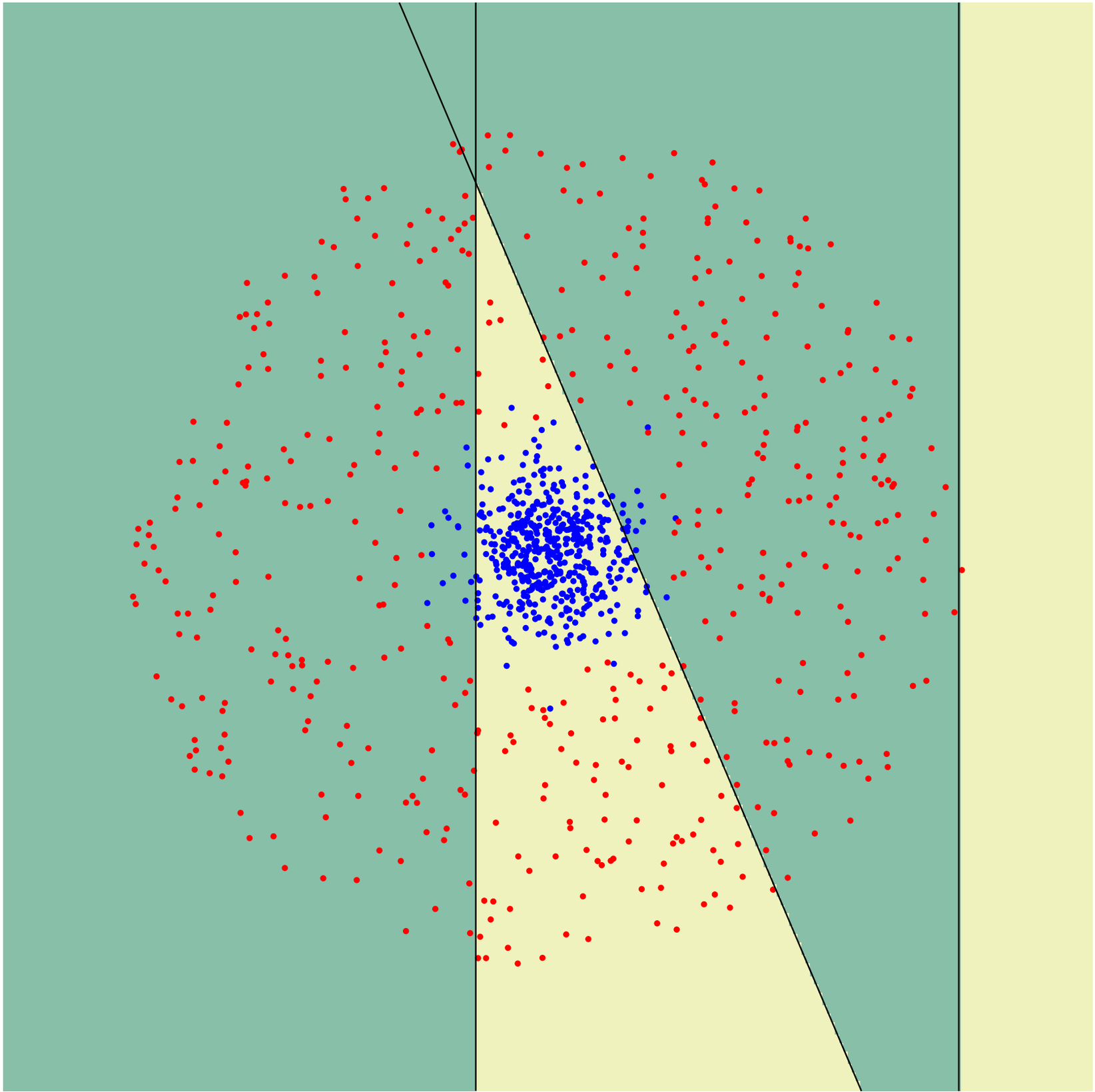


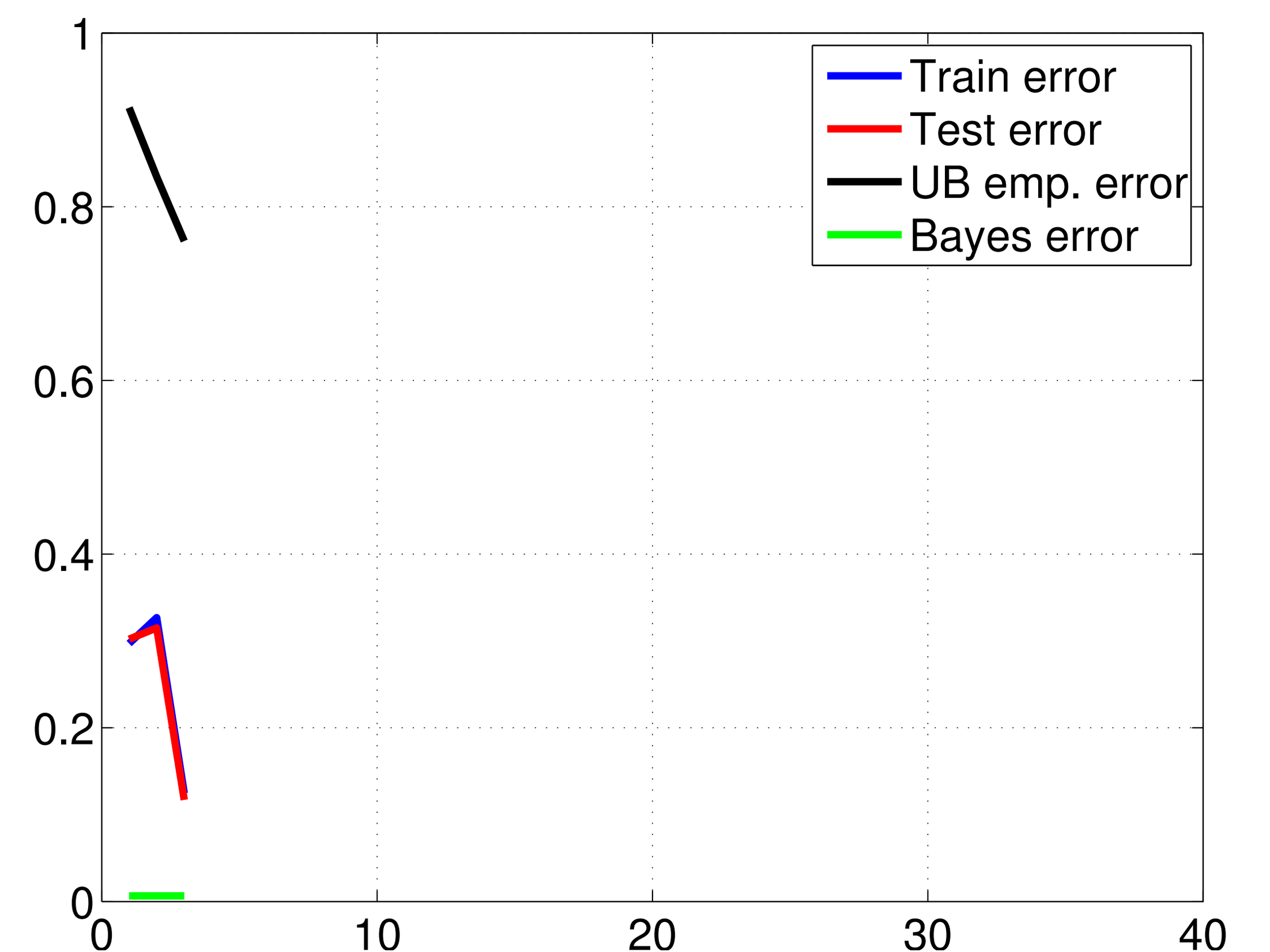


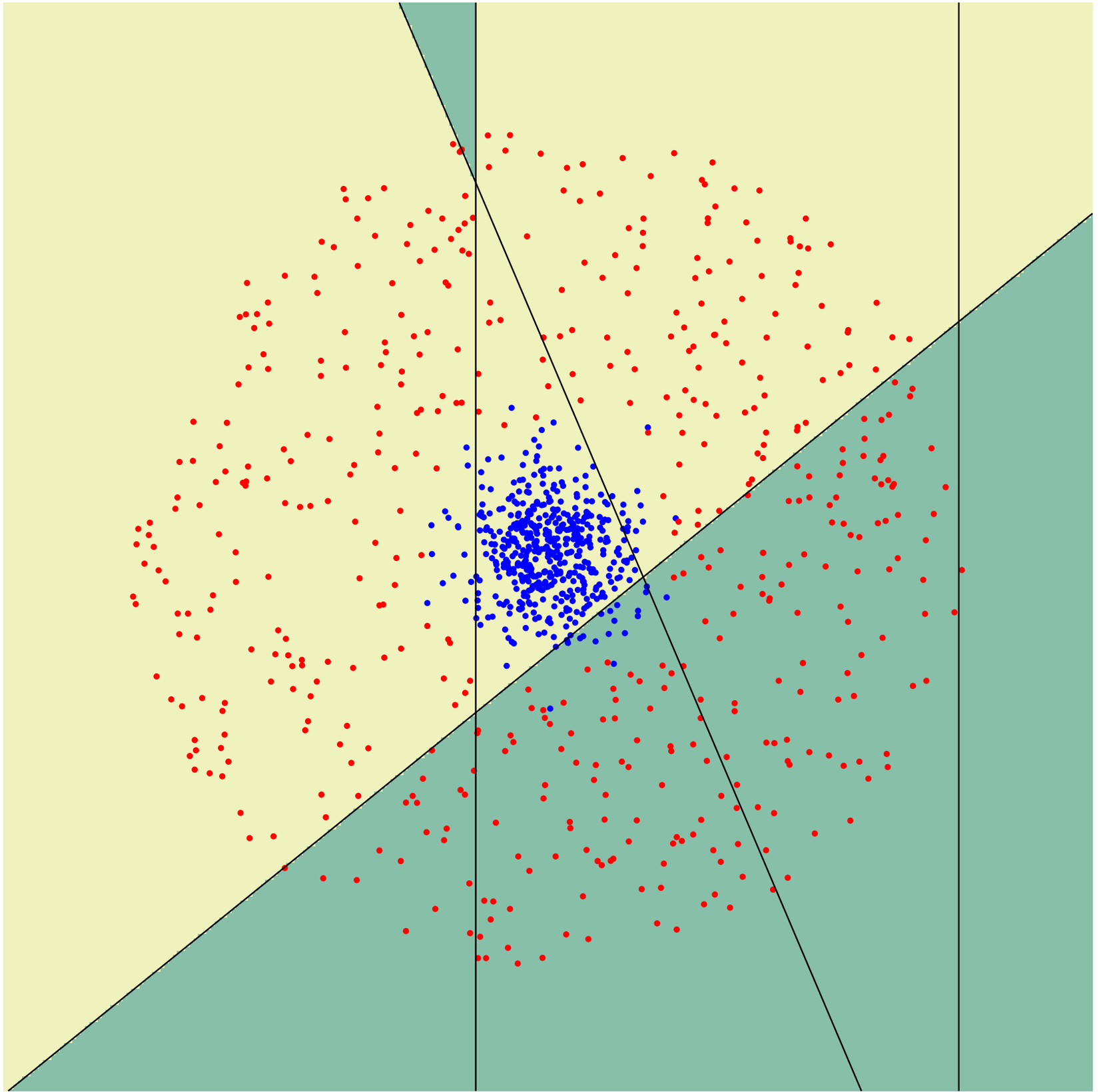

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

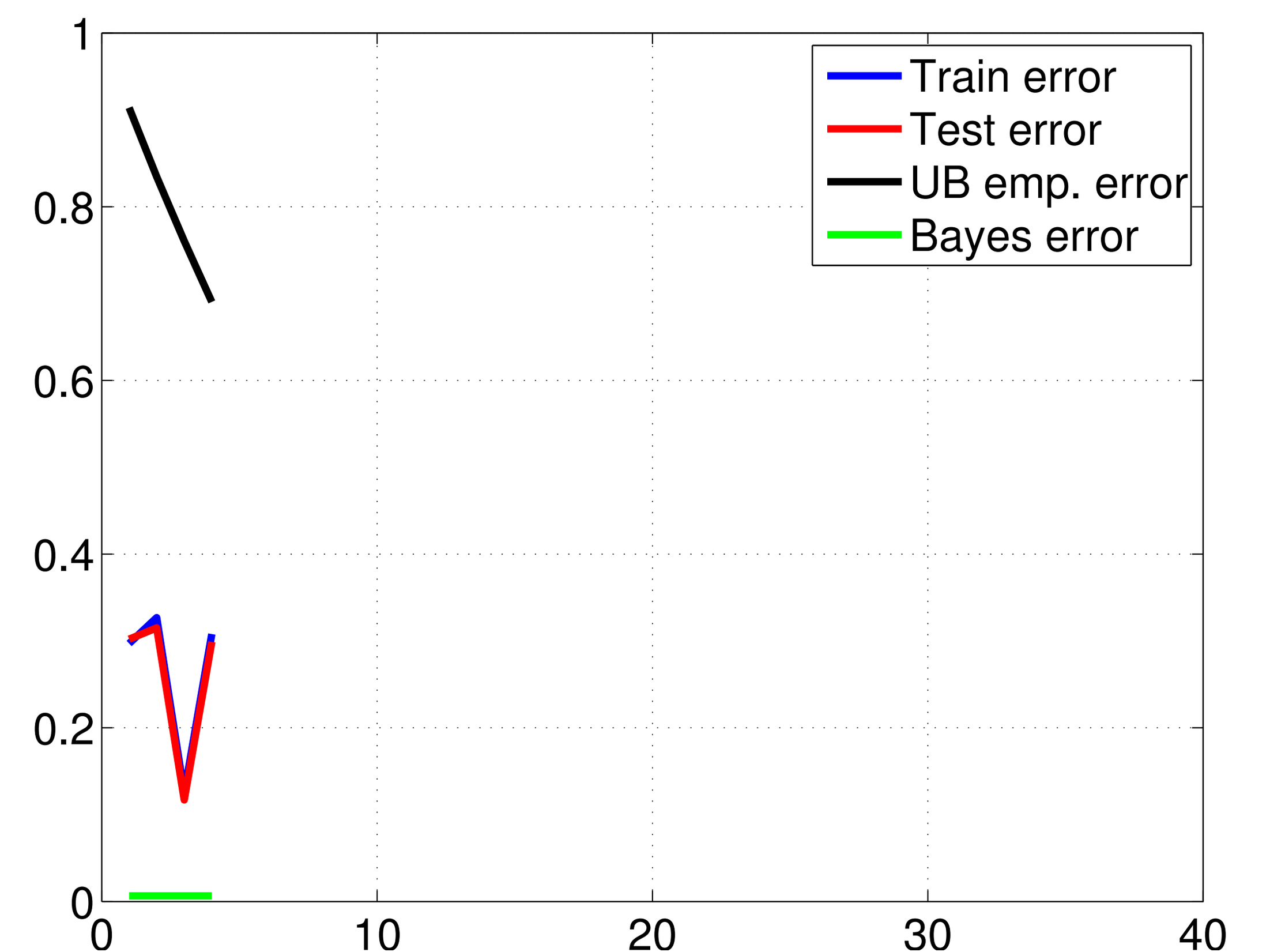


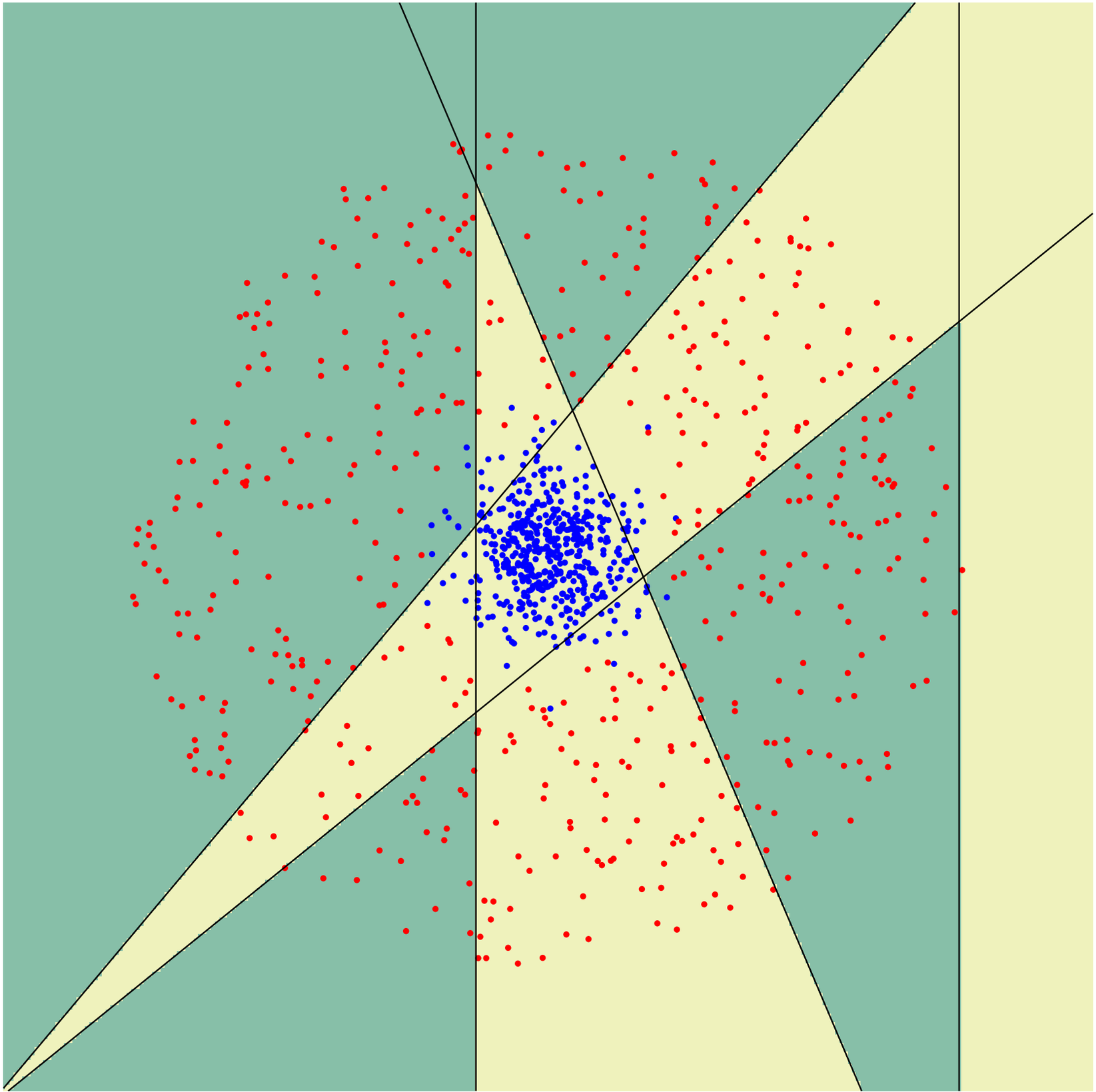

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

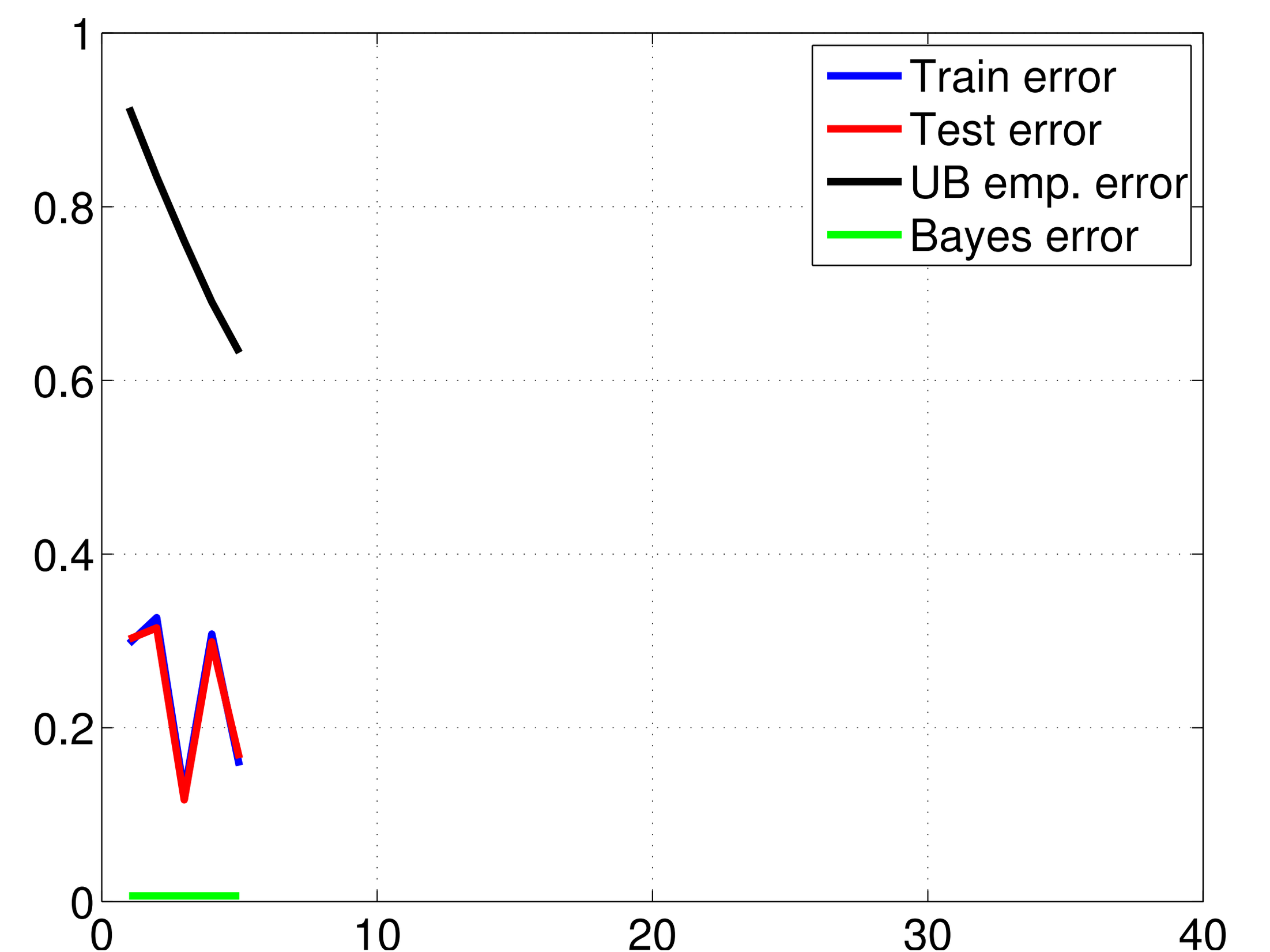


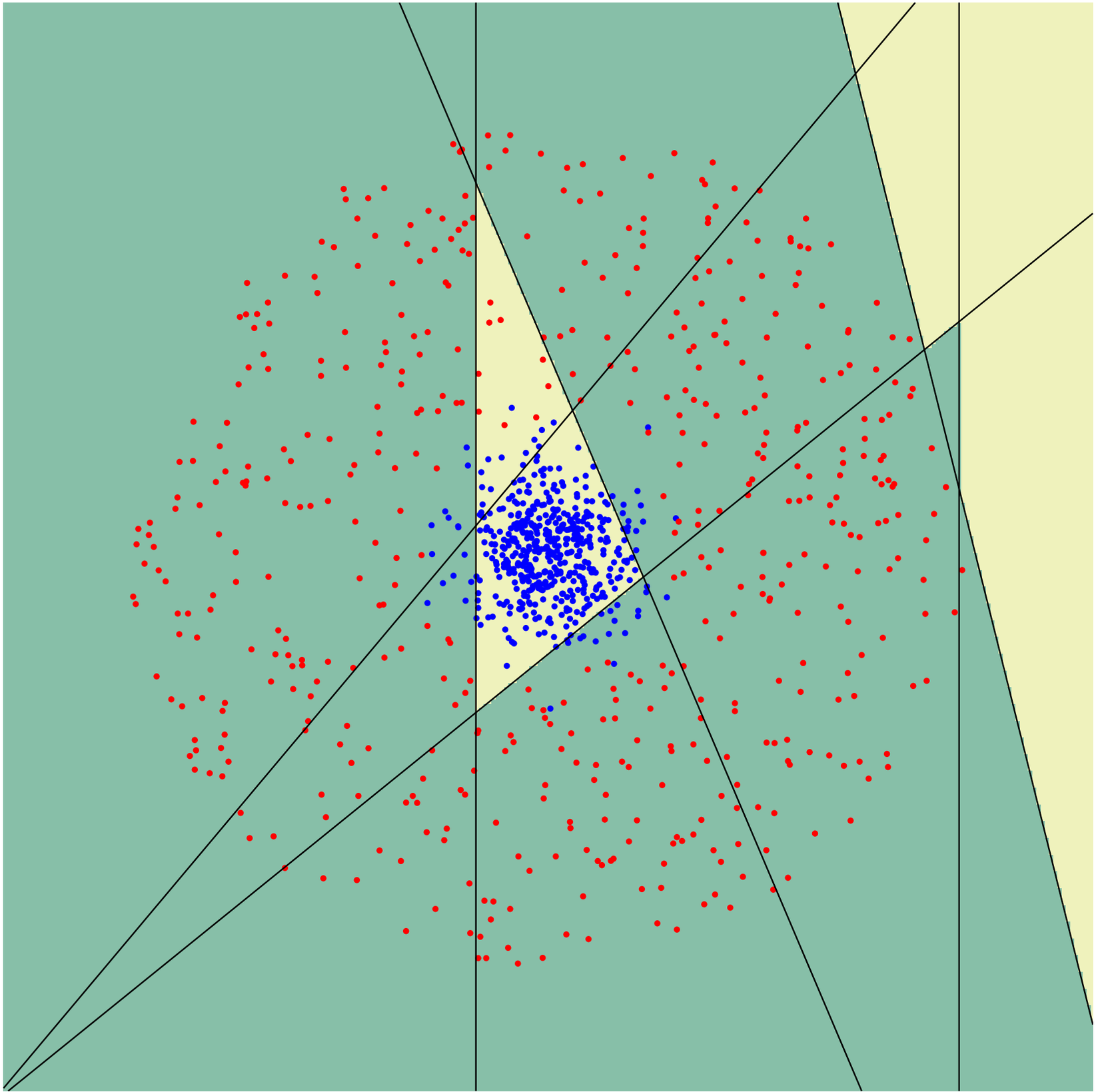

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

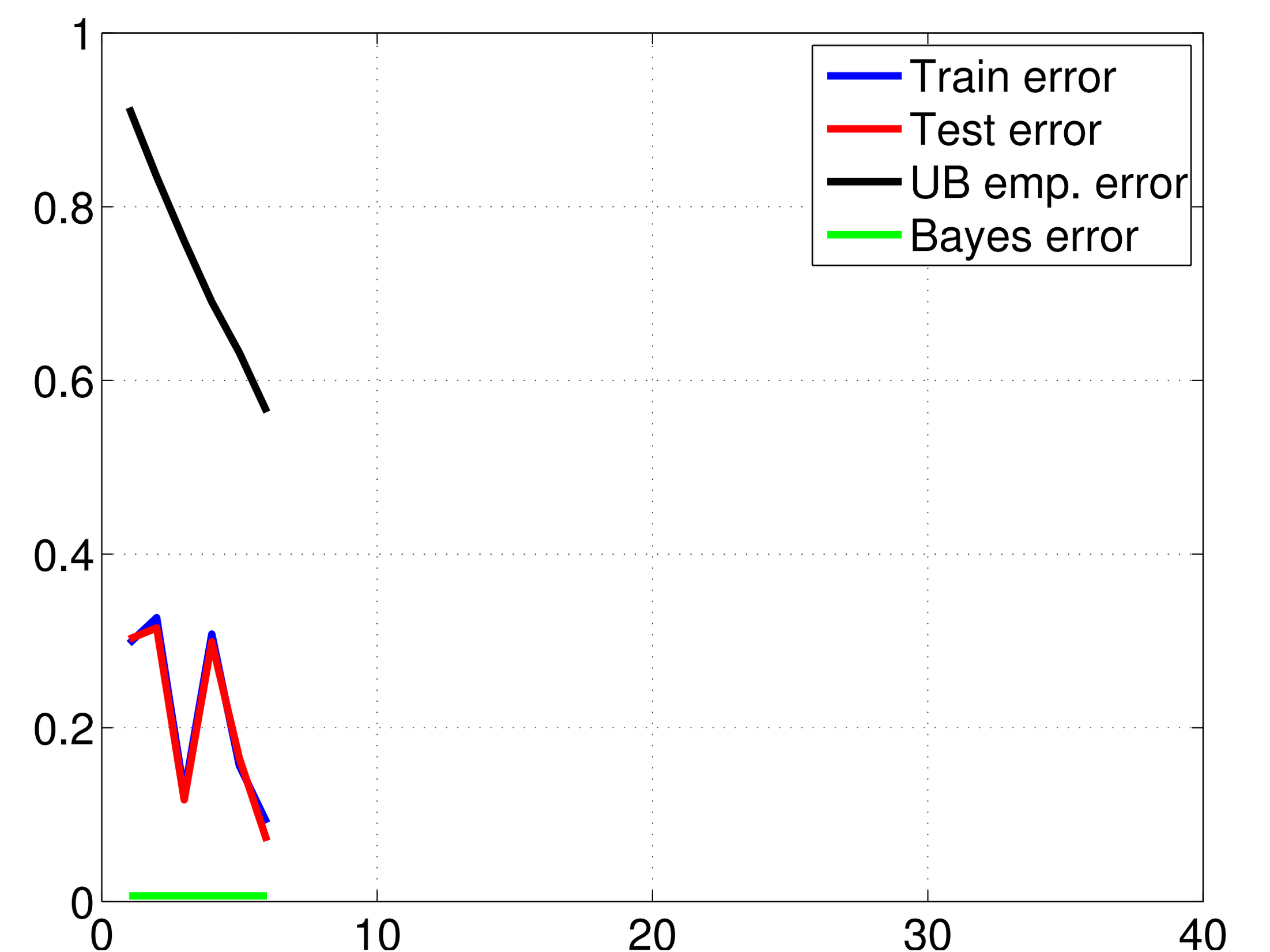


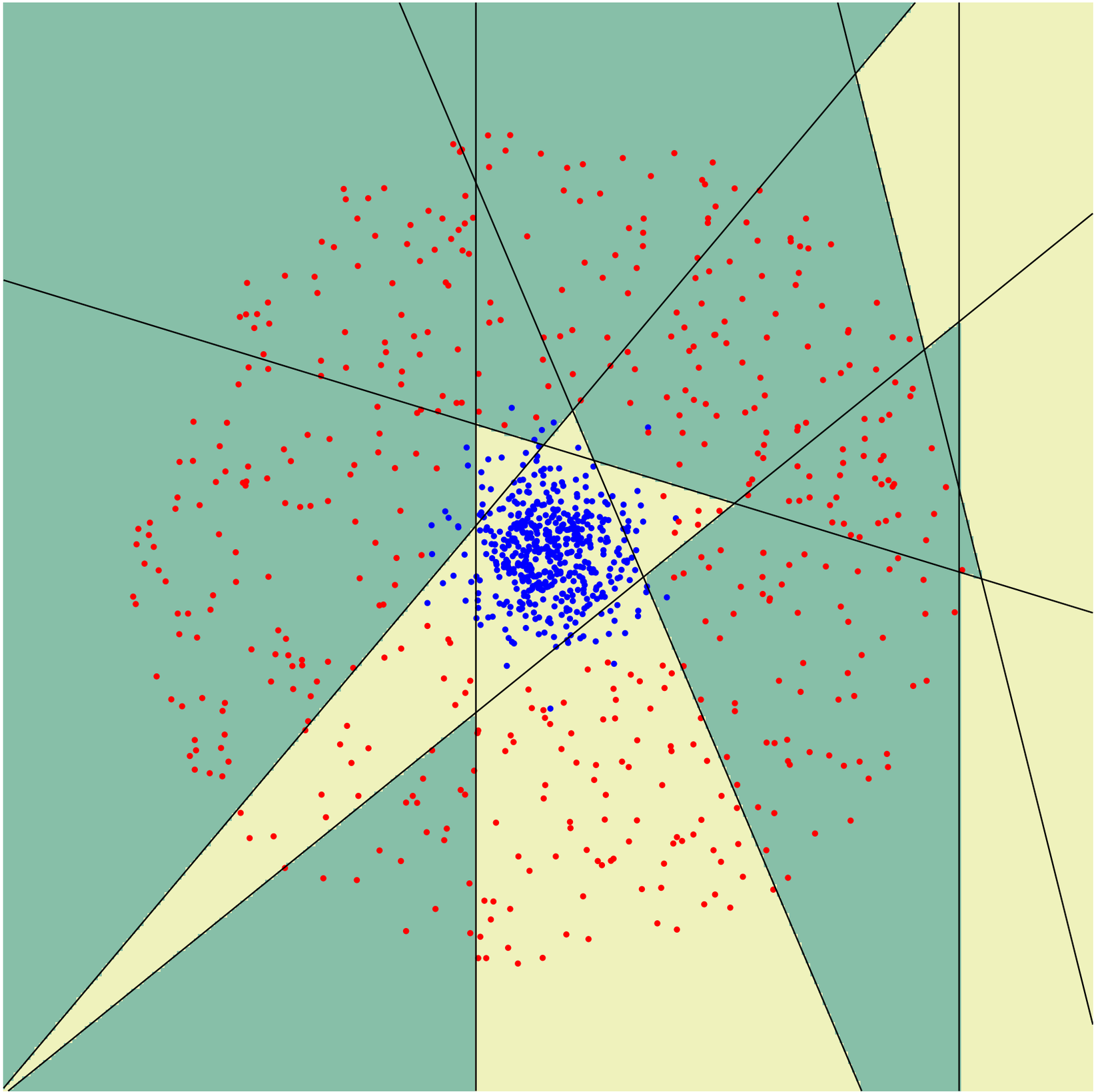

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

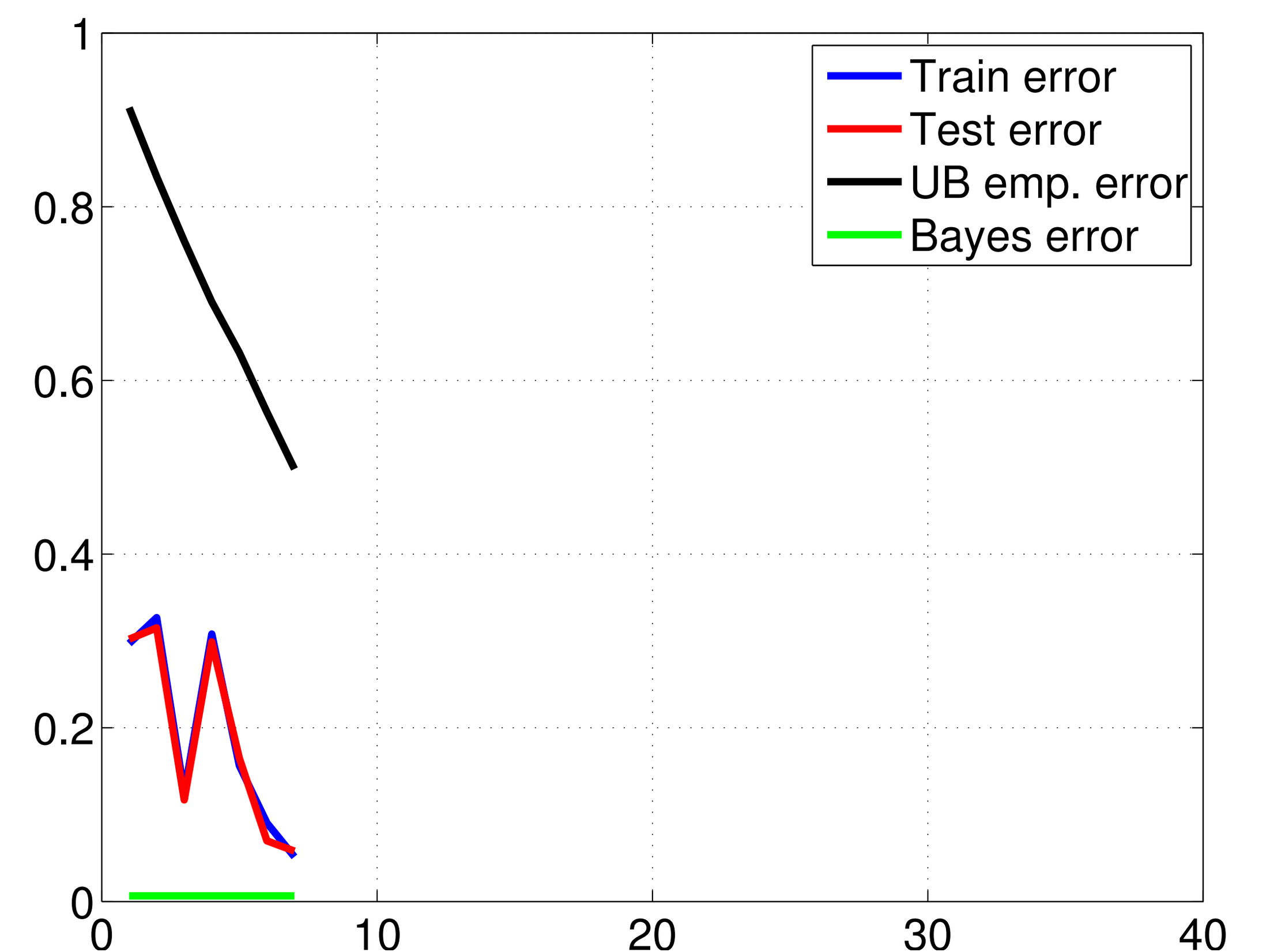


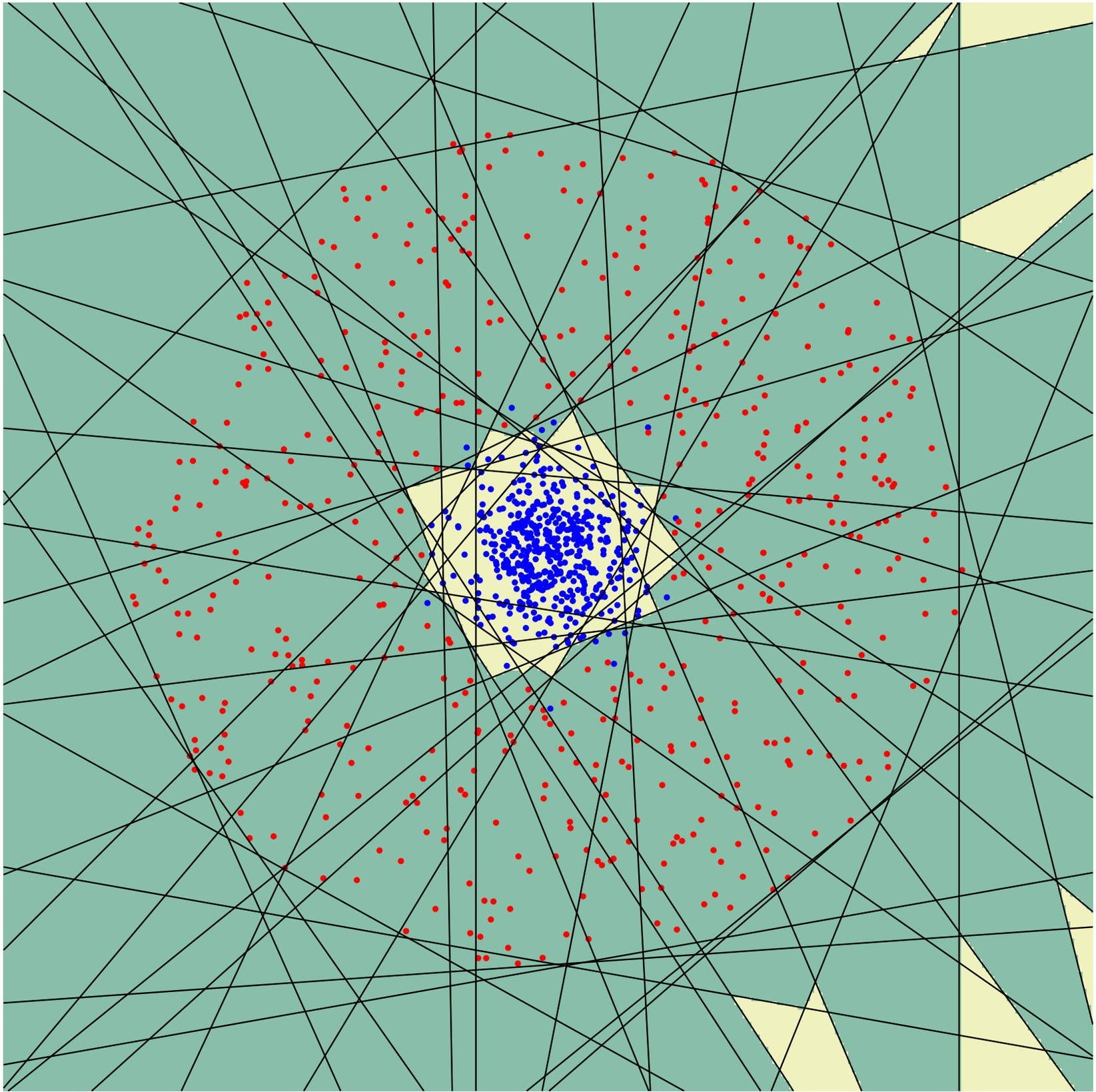

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

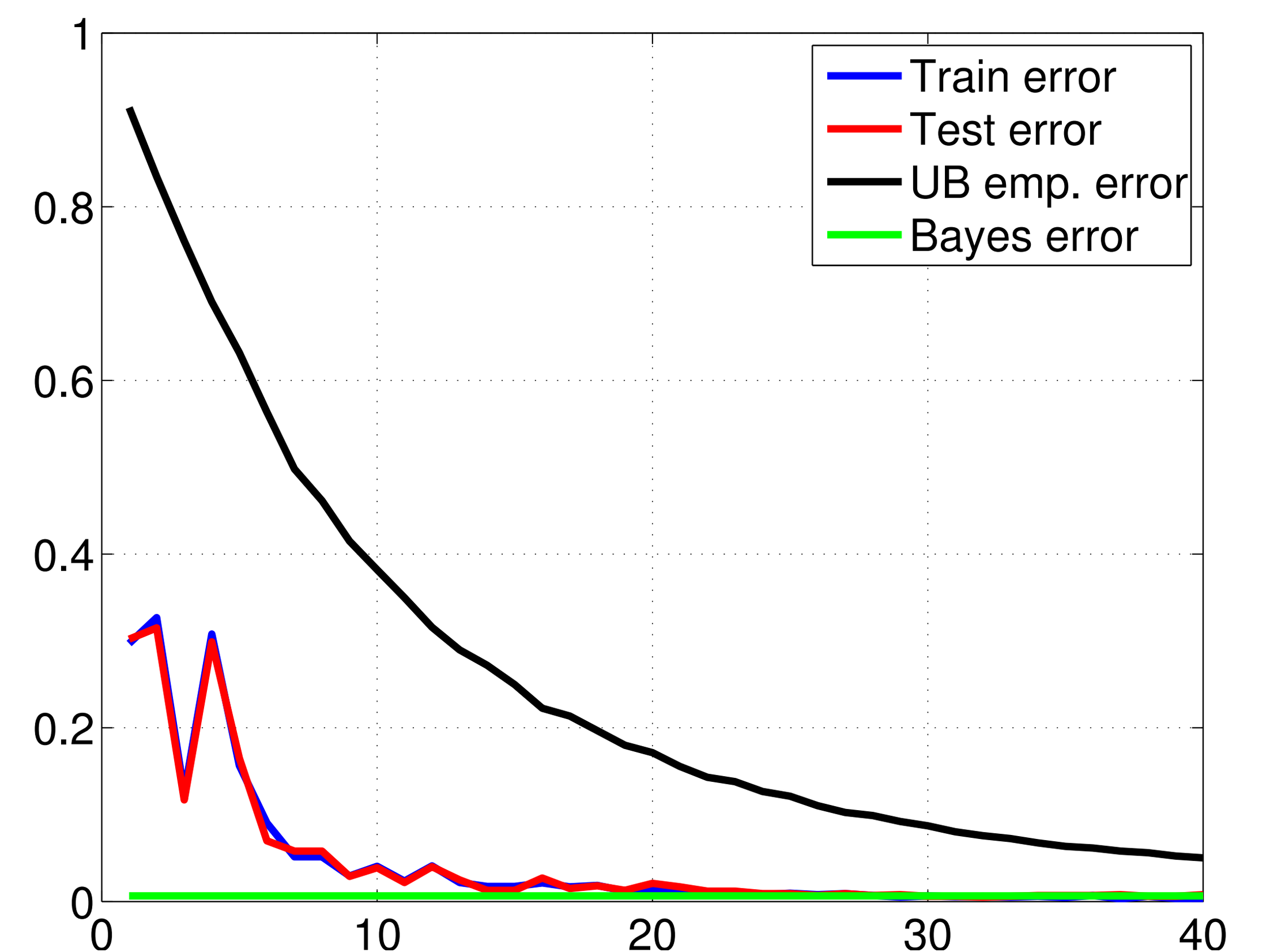


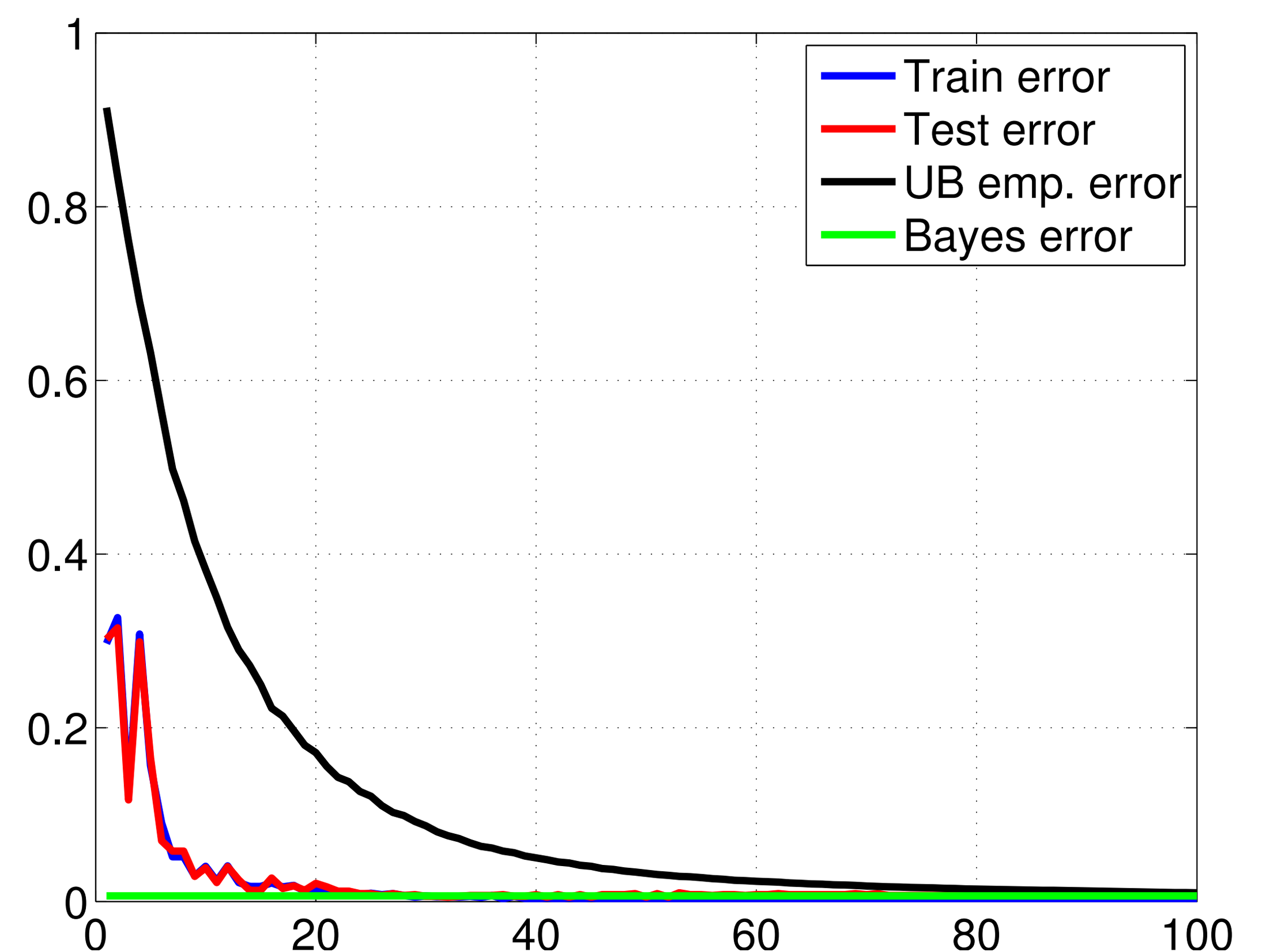

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

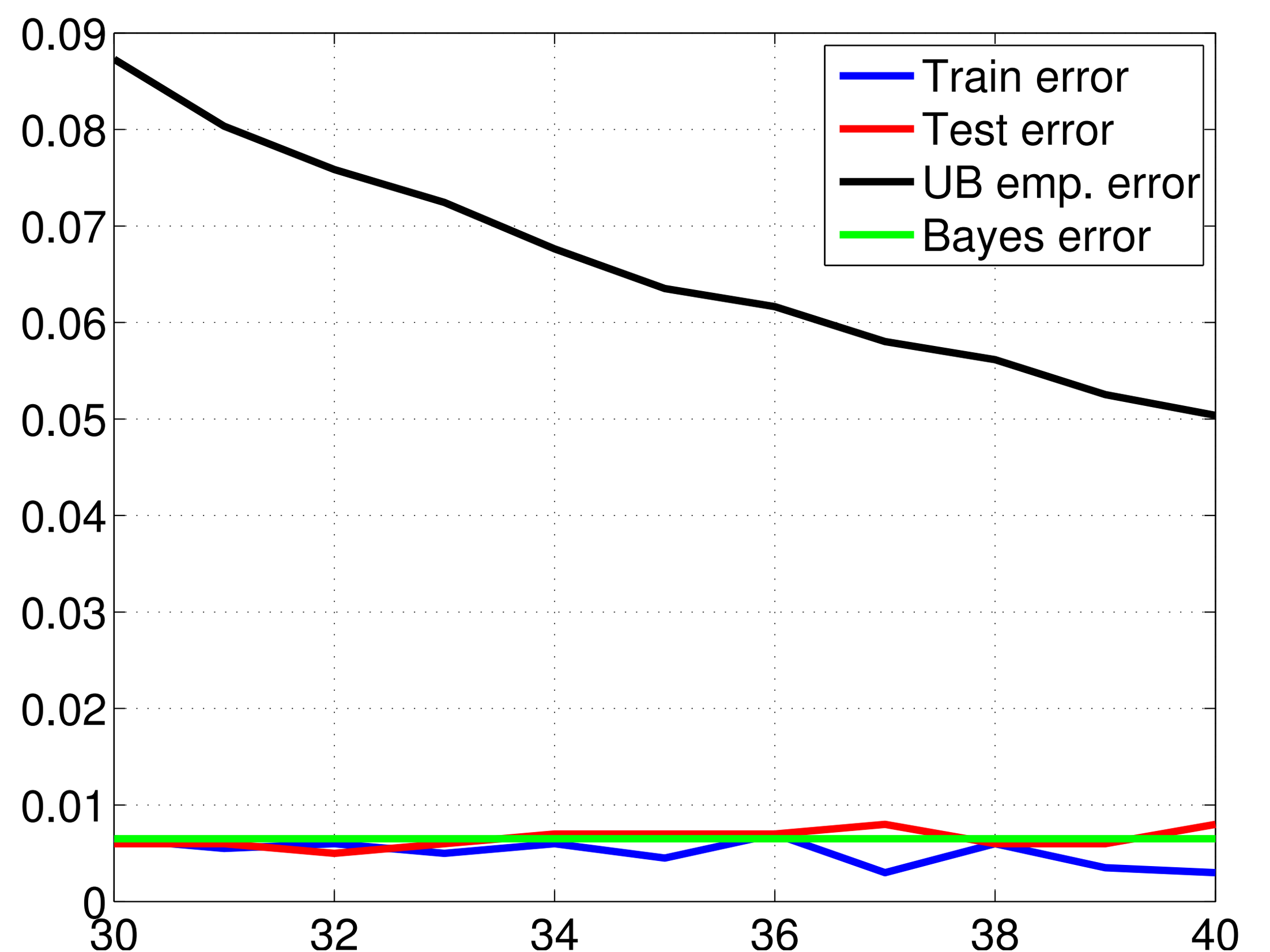



Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40




Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40


Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100


0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
30
32
34
36
38
40
Train error
Test error
UB emp. error
Bayes error