Learning for Time-Constrained Sequential Detection

the algorithmic core of fast (face) detection algorithms

Jiri Matas

work done with Jan Sochman, Michal Perdoch and Jan Cech

Center for Machine Perception
Czech Technical University, Prague

Introduction

- **Time to decision vs. precision trade-off** is inherent in many computer vision problems, e.g. in detection, recognition and matching.

- Often the trade-off is not explicit stated in the problem formulation, but decision time clearly influences both the design and the impact of a method - time is thus implicitly considered an important characteristic of detection and recognition algorithms.

  Example (face detection):

  - Viola-Jones 2002: 20000+ Google Scholar citations.
  - Schneiderman-Kanade 1998: 400 citations.

  Schneiderman-Kanade has (had) smaller error rates, it was one of the first methods to apply machine learning, but *it is 1000x slower*.

Example: van de Sande et al. ICCV 2011

Segmentation as Selective Search for Object Recognition

Koen E. A. van de Sande* Jasper R. R. Uijlings† Theo Gevers* Arnold W. M. Smeulders*
*University of Amsterdam †University of Trento
Amsterdam, The Netherlands Trento, Italy
ksande@uva.nl, jrr@disi.unitn.it, th.gevers@uva.nl, a.w.m.smeulders@uva.nl

Abstract

For object recognition, the current state-of-the-art is based on exhaustive search. However, to enable the use of more expensive features and classifiers and thereby progress beyond the state-of-the-art, a selective search strategy is needed. Therefore, we adapt segmentation as a selective search by reconsidering segmentation: We propose to generate many approximate locations over few and precise object delineations because (1) an object whose location is never generated can not be recognised and (2) appearance and immediate nearby context are most effective for object recognition. Our method is class-independent and is shown to cover 96.7% of all objects in the Pascal VOC 2007 test set using only 1,536 locations per image. Our selective search enables the use of the more expensive bag-of-words method which we use to substantially improve the state-of-the-art by up to 8.5% for 8 out of 20 classes on the Pascal VOC 2010 detection challenge.

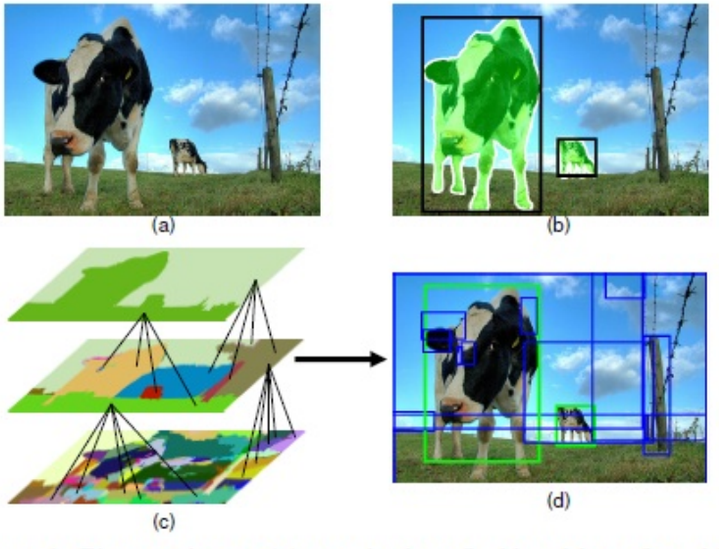


Figure 1. Given an image (a) our aim is to find its objects for which the ground truth is shown in (b). To achieve this, we adapt segmentation as a selective search strategy: We aim for high recall by generating locations at all scales and account for many different scene conditions by employing multiple invariant colour spaces. Example object hypotheses are visualised in (d).

Wald's Sequential Analysis

- Wald's sequential analysis (Wald, 1947) provides a building block for quasi-optimal solutions of time-constrained recognition problems.
- It is not be clear how to measure time to decision.
  Algorithmic complexity $\mathcal{O}(.)$? The "hidden" constants may be huge.
  Running time? Depends on the hardware.
  In Wald's statistical decision theory, the number of measurements is minimized.
- We assume (and show) that the number of measurements is a good substitute for time-to-decision.
- Wald's Sequential Probability Ratio Test (SPRT) deals with the decision error v. time trade-off optimally, but:
  1. ordering of measurements is assumed to be given a priori or is irrelevant (i.i.d. measurements)
  2. unlimited number of i.i.d. measurements available (implies that zero-error decision is possible!)
  3. multidimensional pdf's are assumed to be known (reduced to estimation of a pair of 1D pdfs in the i.i.d. case)

- **WaldBoost** (Sochman and Matas, 2005) integrates feature selection and ordering provided by AdaBoost with SPRT, removing some limitations of SPRT.

Overview

We present two applications of WaldBoost (1. and 2.) and
two of Wald's SPRT (3. and 4.) in computer vision algorithms:

- Application 1: **Optimal Cascaded Detector of the Viola-Jones type**.
  We formalize the problem as constrained optimization, showing that Waldboost algorithm provides a quasi-optimal solution.
  The approach is demonstrated on face detection (Sochman and Matas, CVPR 05).

- Application 2: **Fast Detector of Similarity-invariant Regions.** Given an executable of a carefully designed detector of repeatable "interest points" that provides an unlimited number of positive (interesting region) and negative examples, Waldboost learns an approximation of the detector whose output differs slightly from the "supervisor" and is significantly faster. (Matas and Sochman, ICRA 2007)
  Might generalise to "fast approximation of a black box algorithm".

- Application 3: **Correspondence Selection with Sequential Cosegmentation** (Cech, Matas, Perdoch, PAMI 09). Wald's analysis applies directly, measurements are ordered.

- Application 4: **Optimal Randomised RANSAC**(Matas and Chum, ICCV 05). Wald's analysis applies directly, samples are i.i.d.

Problem Formulation

Basic notions:

$x_1, x_2, \ldots$ ordered sequence of measurements

$S = \{S_1, \ldots, S_t\}$ sequential strategy

where $S_i : \{x_1, \ldots, x_i\} \to \{-1, +1, \sharp\}$

Sequential strategy S is characterised by:

1. α_S and β_S errors of the first and the second kind
2. $\bar{T}_S = E_X(T_S(x))$ average evaluation time
   where $T_S(x) = \arg\min_i(S_i(x) \neq \sharp)$ time-to-decision

Problem formulation:

$$S^* = \arg\min_S \bar{T}_S \quad \text{s.t.} \quad \begin{aligned} \beta_S &\leq \beta, \\ \alpha_S &\leq \alpha \end{aligned}$$

for specified α and β

Note: The formulation above assumes $P(k), k \in \{-1, 1\}$ exist and are known.
A formulation, requiring only $P(x|k)$, where $\bar{T}_S = \max_k E_X(T_S(x|k))$ is the maximum of the expected decision times for classes $\{1, -1\}$, leads to identical results.

Wald's Sequential Probability Ratio Test

Sequential Probability Ratio Test (SPRT)

SPRT is a sequential strategy S^*

$$S_t^* = \begin{cases} +1, & R_t \leq B \\ -1, & R_t \geq A \\ \sharp, & B < R_t < A \end{cases} \qquad R_t = \frac{p(x_1, ..., x_t | y = -1)}{p(x_1, ..., x_t | y = +1)}.$$

Practical (and nearly optimal) setting for the thresholds A and B

$$A = \frac{1-\beta}{\alpha}, \qquad B = \frac{\beta}{1-\alpha}.$$

Difficulties

SPRT is an optimal sequential test for the formulated problem **but**:

1. if measurements not i.i.d then their order must be given.
2. multidimensional pdf's $p(x_1, ..., x_t | y = \mathit{class})$ have to be computed in non i.i.d case.

Discrete AdaBoost (Freund and Schapire 1997)
(Gödel prize 2003)

Given: $(x_1, y_1), \ldots, (x_L, y_L); x_i \in \mathcal{X}, y_i \in \{-1, +1\}; \mathcal{H} = \{h(x)\}$

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

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

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

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

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

Output the strong classifier:

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

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

- The objective is to minimize $\epsilon_{tr} = \frac{1}{L}|\{i : H(x_i) \neq y_i\}|$
- It is upperbounded by $\epsilon_{tr}(H) \leq \Pi_{t=1}^{T} Z_t$

How to select α_t and h_t

- Given h_t select α_t to greedily minimize $Z_t(\alpha)$ in each step
- $Z_t(\alpha)$ is convex differentiable function with one extremum
  $\Rightarrow$ $h_t(x) \in \{-1, 1\}$ then optimal $\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 1$ for optimal α_t
  $\Rightarrow$ Justification of selection of h_t according to ϵ_t
- 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:

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

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

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

Effect on h_t

- α_t minimize $Z_t \Rightarrow \sum_{i:h_t(x_i)=y_i} D_{t+1}(i) = \sum_{i:h_t(x_i)\neq y_i} D_{t+1}(i)$
- Error of h_t on D_{i+1} is $1/2$
- Next weak classifier is the most "independent" one

Upper Bound Theorem

Theorem: The following upper bound holds on the training error of H

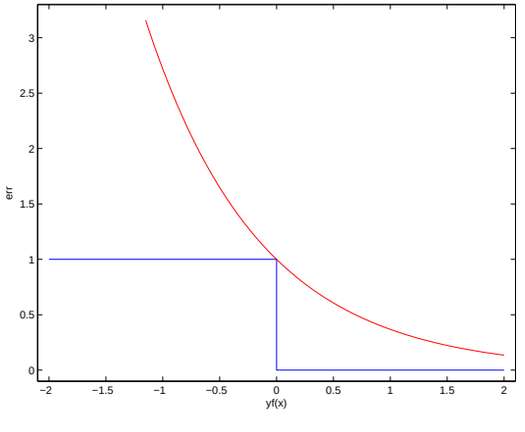
$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leq \prod_{t=1}^{T} Z_t$$

Proof: By unravelling the update rule

$$\begin{aligned} D_{T+1}(i) &= \frac{exp(-\sum_t \alpha_t y_i h_t(x_i))}{m \prod_t Z_t} \\ &= \frac{exp(-y_i f(x_i))}{m \prod_t Z_t}. \end{aligned}$$

If $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $exp(-y_i f(x_i)) > 1$, thus

$$\begin{aligned} [\![H(x_i) \neq y_i]\!] &\leq exp(-y_i f(x_i)) \\ \frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i exp(-y_i f(x_i)) \\ &= \sum_i (\prod_t Z_t) D_{T+1}(i) = \prod_t Z_t \end{aligned}$$



Choosing α_t

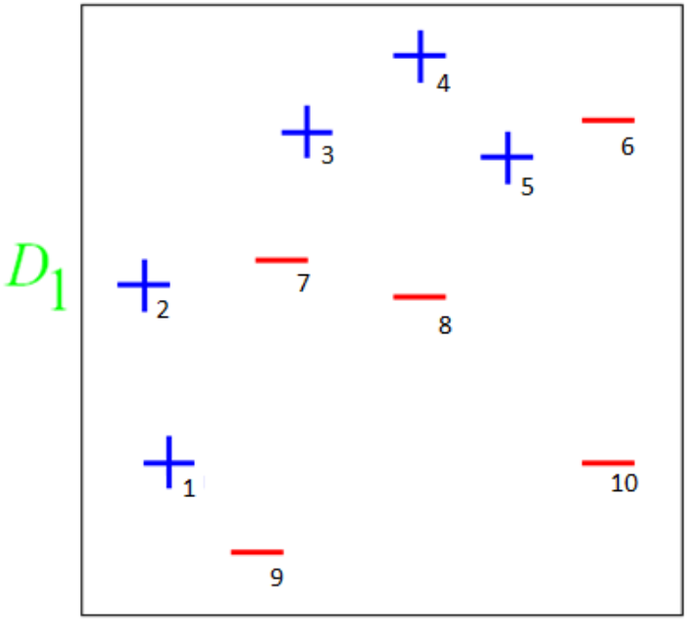
We attempt to minimise $Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}$:

$$\begin{aligned}
\frac{dZ}{d\alpha} = -\sum_{i=1}^{m} D(i) y_i h(x_i) e^{-y_i \alpha_i h(x_i)} &= 0 \\
-\sum_{i:y_i = h(x_i)} D(i) e^{-\alpha} + \sum_{i:y_i \neq h(x_i)} D(i) e^{\alpha} &= 0 \\
-e^{-\alpha}(1-\epsilon) + e^{\alpha}\epsilon &= 0 \\
\alpha &= \frac{1}{2} \log \frac{1-\epsilon}{\epsilon}
\end{aligned}$$

$\Rightarrow$ The minimisator of the upper bound is $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.

AdaBoost Example

Data	1	2	3	4	5	6	7	8	9	10
Class	+	+	+	+	+	-	-	-	-	-
D_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1



AdaBoost Example

Data	1	2	3	4	5	6	7	8	9	10
Class	+	+	+	+	+	-	-	-	-	-
$D_2.Z_2$	0.07	0.07	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07

$Z_2 = 0.92$

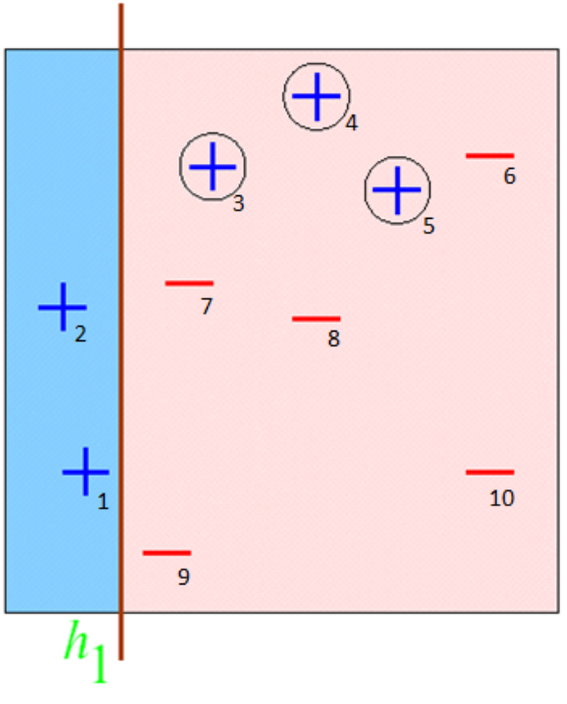
Round 1

e_1... weights of incorrect samples

$\alpha_1 = \frac{1}{2} \log \frac{1-e_1}{e_1}$

$D_2 \approx D_1 \sqrt{e_1/(1-e_1)}$ correct

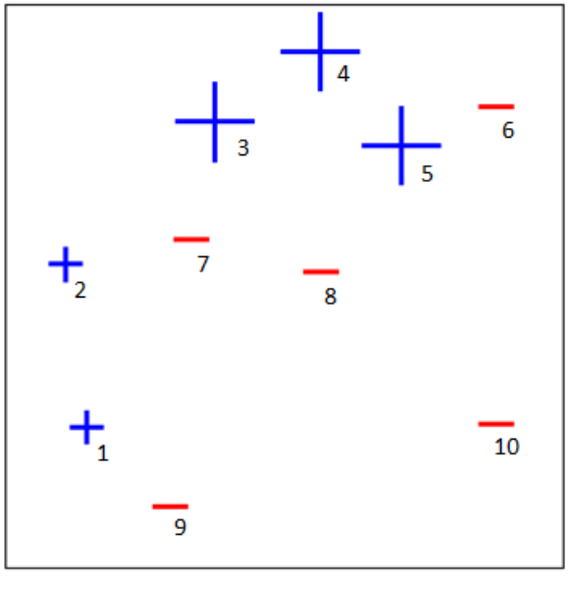
$D_2 \approx D_1 \sqrt{(1-e_1)/e_1}$ incorrect



$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

D_2

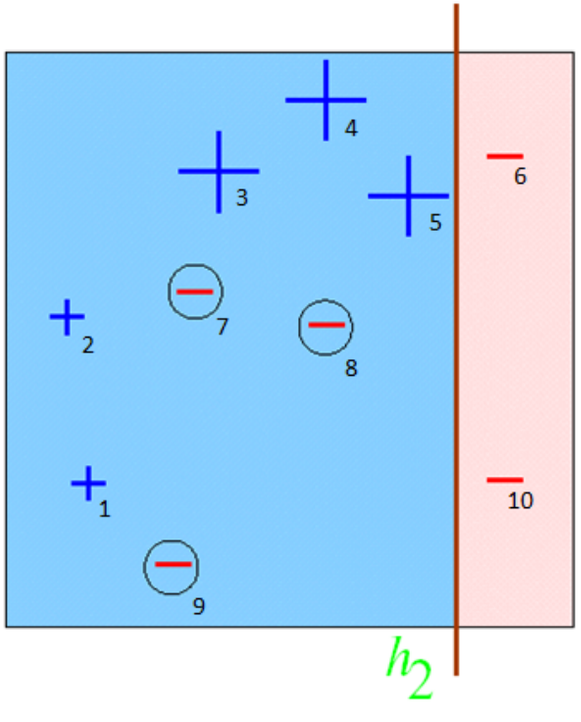


AdaBoost Example

Data	1	2	3	4	5	6	7	8	9	10
Class	+	+	+	+	+	-	-	-	-	-
$D_3.Z_3$	0.04	0.04	0.09	0.09	0.09	0.04	0.14	0.14	0.14	0.14

$Z_3 = 0.82$

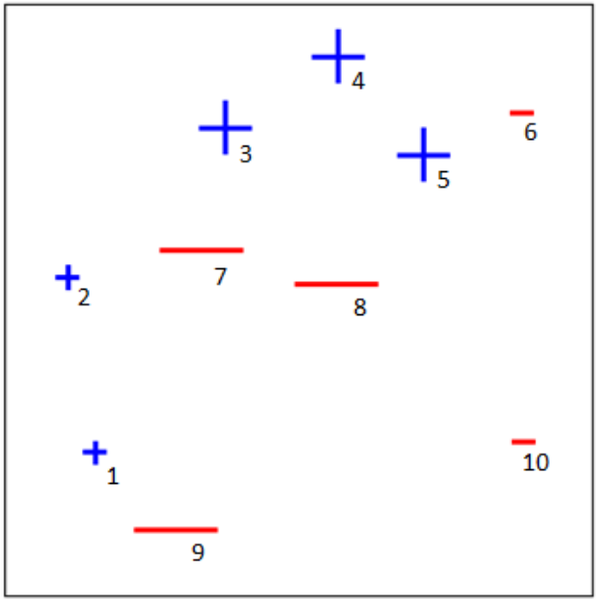
Round 2



$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

D_3

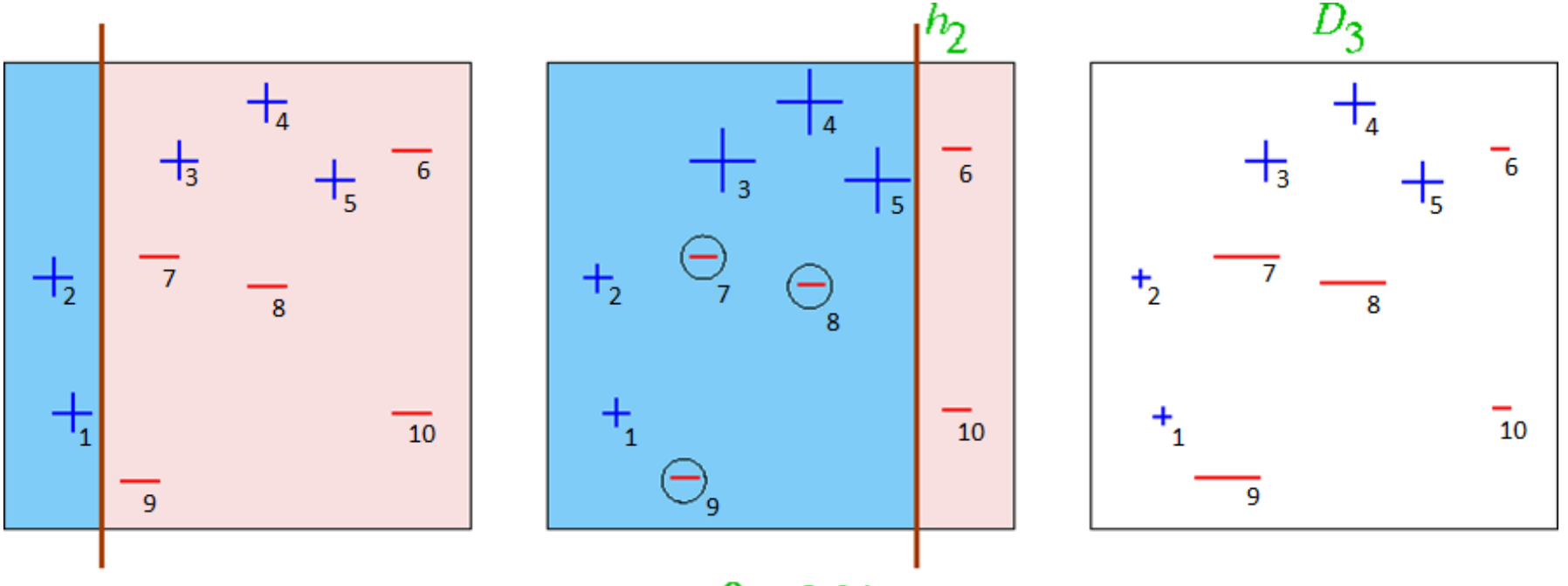


AdaBoost Example

Data	1	2	3	4	5	6	7	8	9	10
Class	+	+	+	+	+	-	-	-	-	-
$D_3.Z_3$	0.04	0.04	0.09	0.09	0.09	0.04	0.14	0.14	0.14	0.14

$Z_3 = 0.82$

Round 2



$\varepsilon_2 = 0.21$

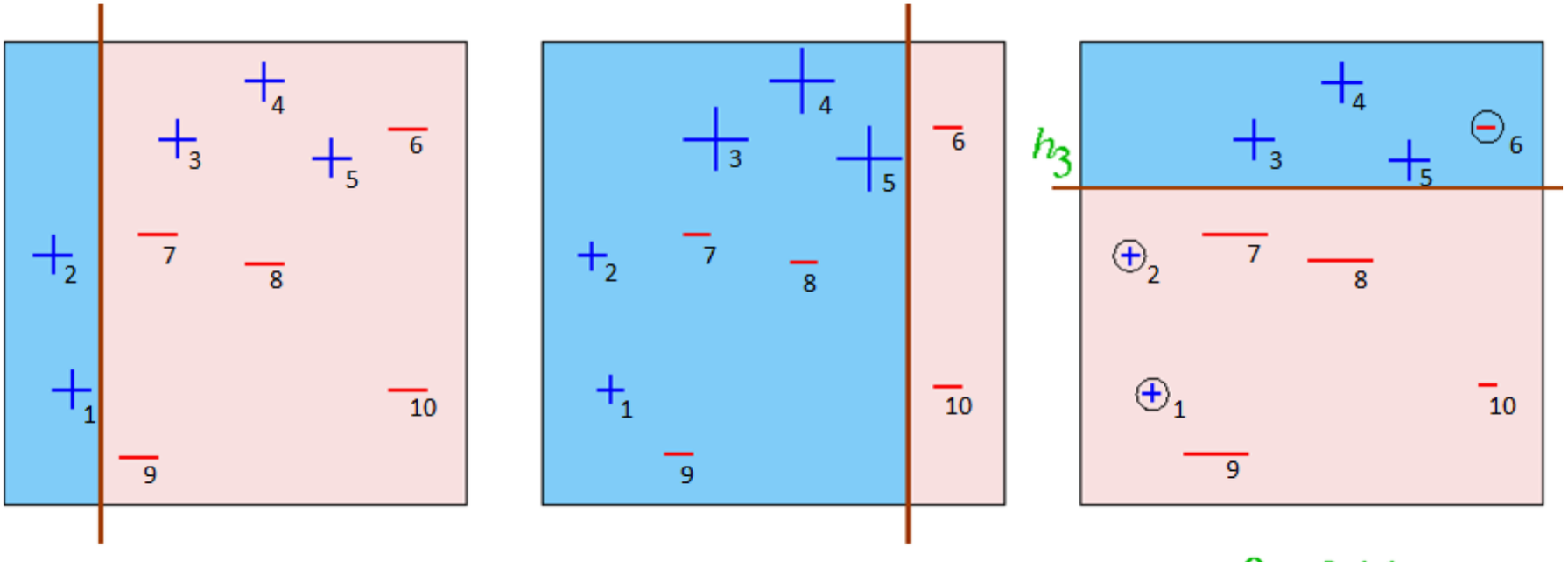
$\alpha_2 = 0.65$

AdaBoost Example

Data	1	2	3	4	5	6	7	8	9	10
Class	+	+	+	+	+	-	-	-	-	-
$D_4.Z_4$	0.11	0.11	0.04	0.04	0.04	0.11	0.07	0.07	0.07	0.02

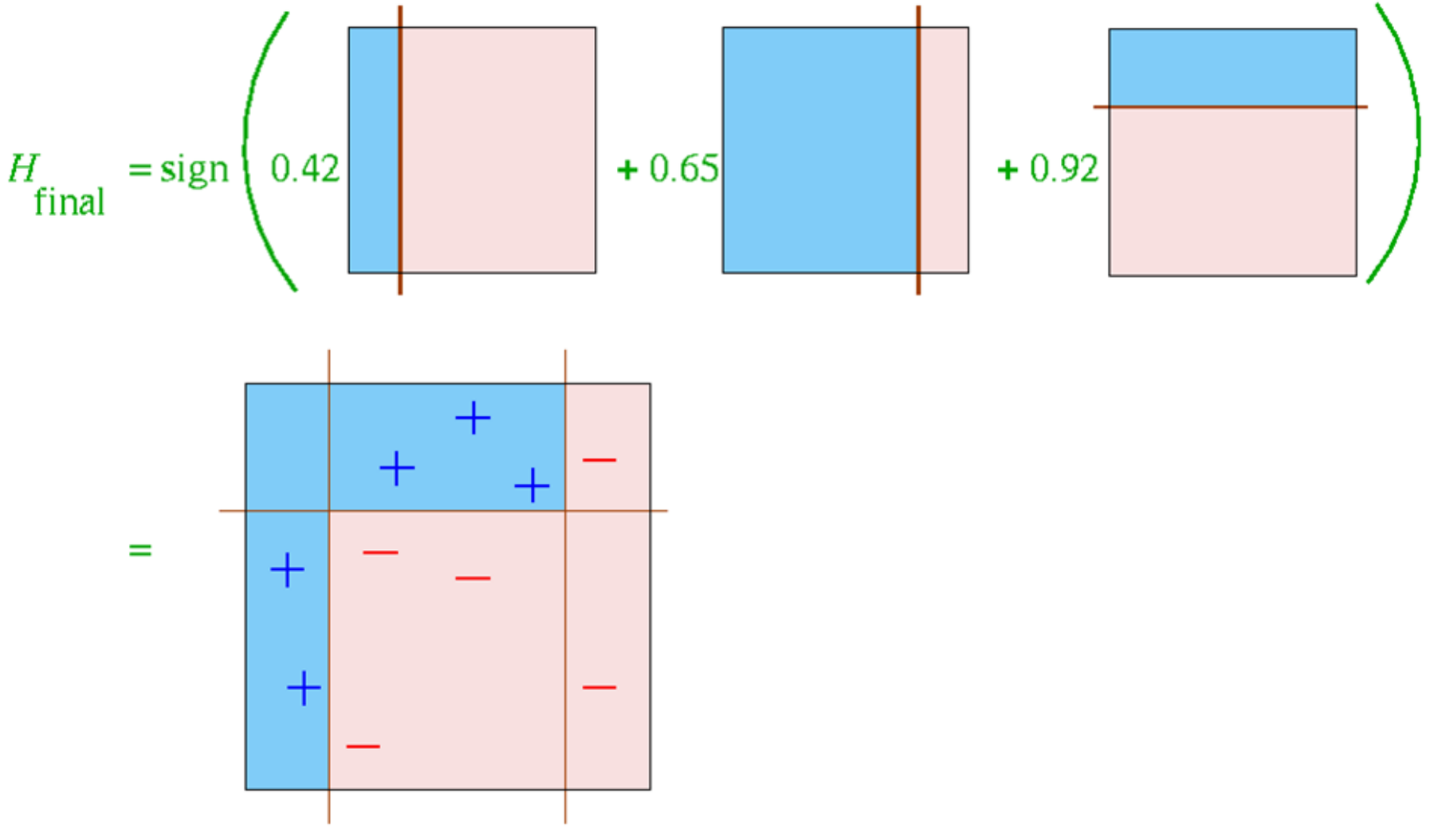
$Z_4 = 0.68$

Round 3



AdaBoost Example

Final Classifier

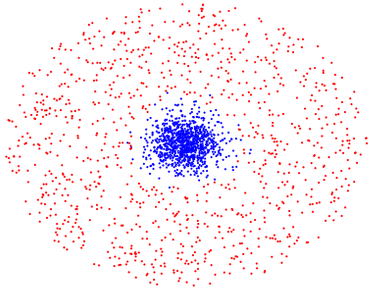


AdaBoost: Summary of the Algorithm

Example:

$h(x)$: perceptron

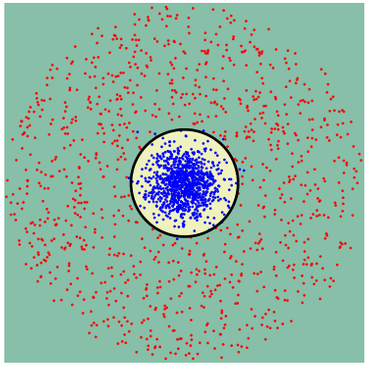
Data:
Training set



$\bullet \sim N(0,1)$ $\quad \bullet \sim \frac{1}{r\sqrt{8\pi^3}} e^{-1/2(r-4)^2}$

P($\bullet$) = 0.5 $\quad$ P($\bullet$) = 0.5

GroundTruth

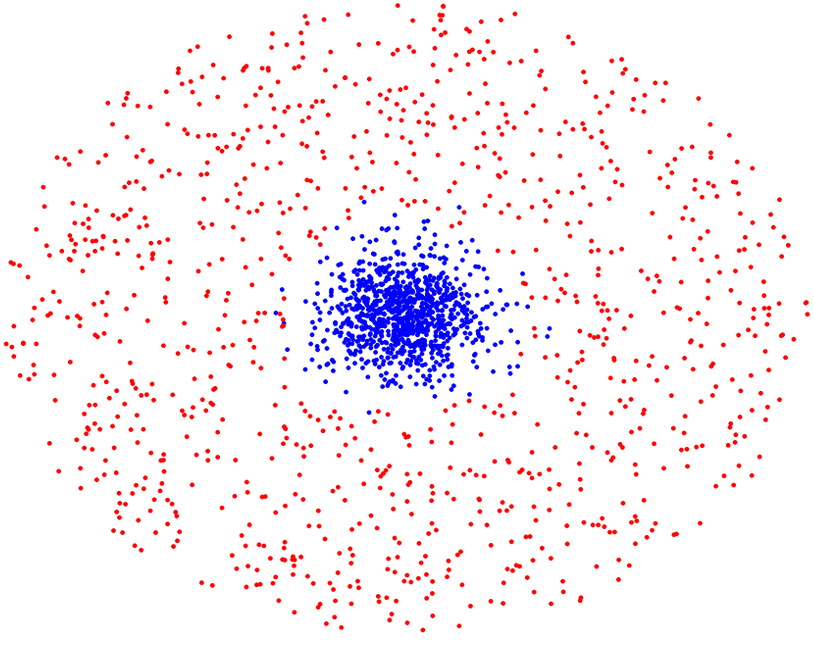


AdaBoost: Summary of the Algorithm

Initialization...

AdaBoost: Summary of the Algorithm

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

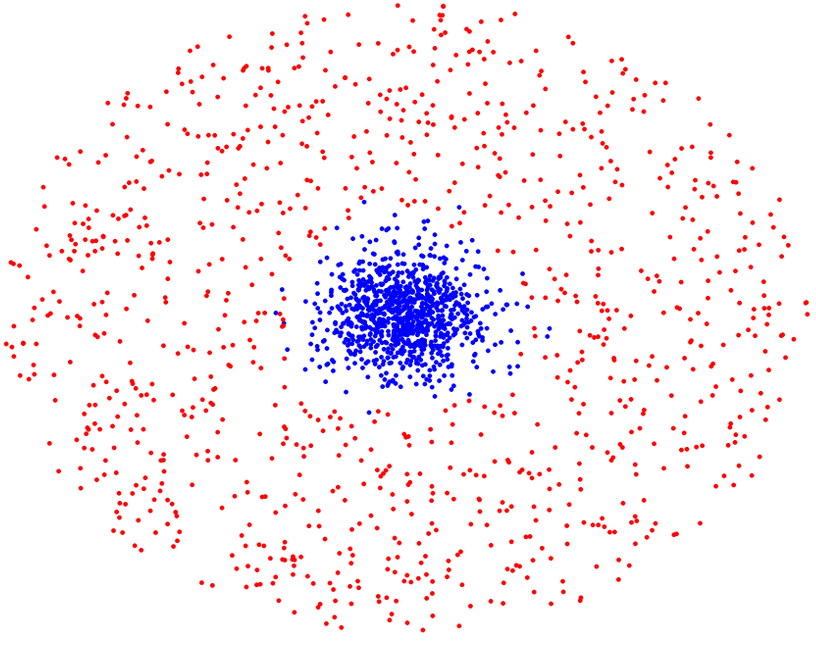


AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$



AdaBoost: Summary of the Algorithm

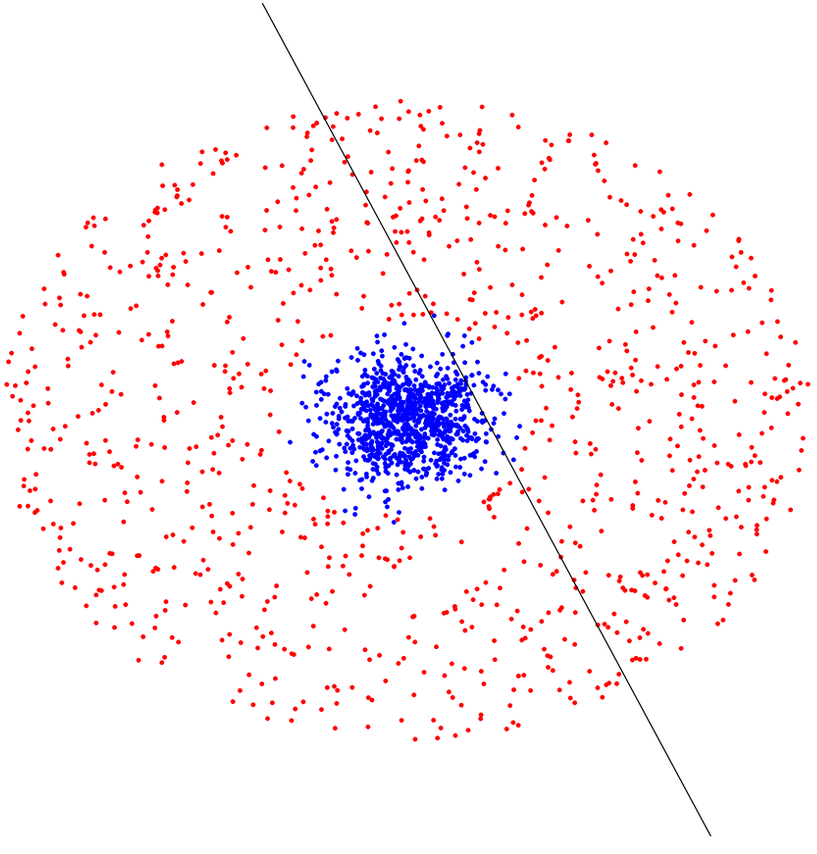
Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop

$t = 1$



AdaBoost: Summary of the Algorithm

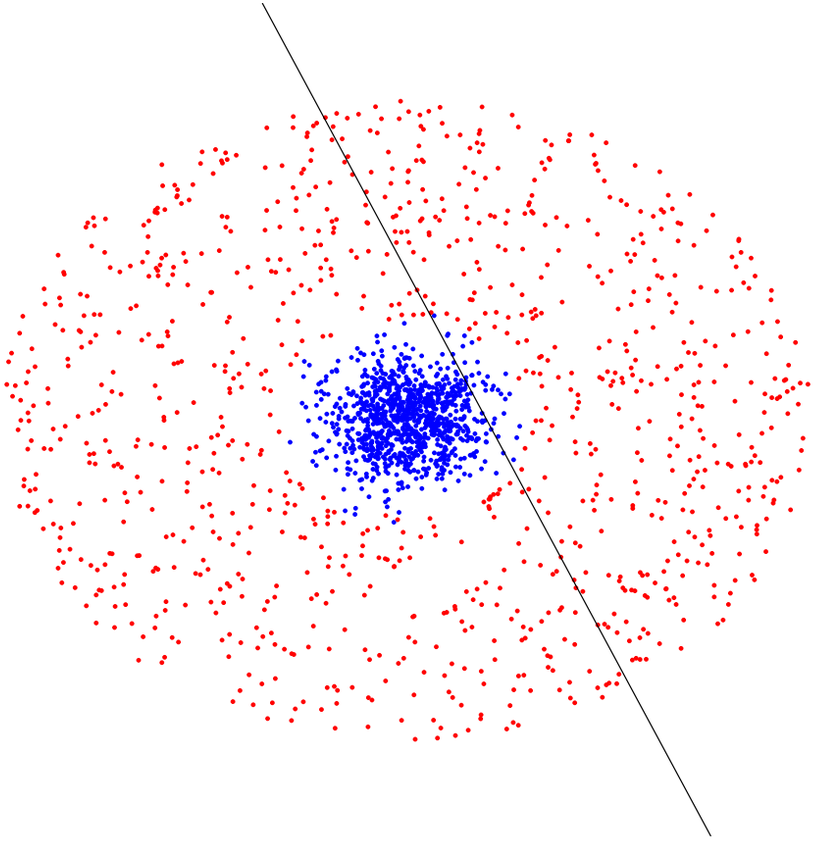
Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$

$t = 1$



AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

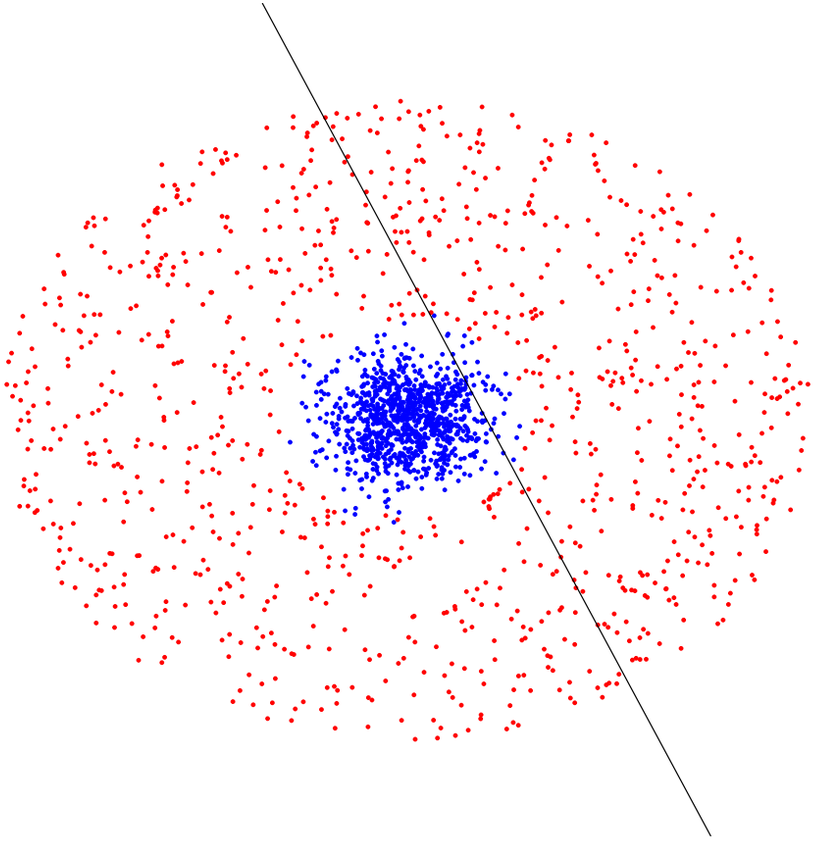
$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

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

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

$t = 1$



AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

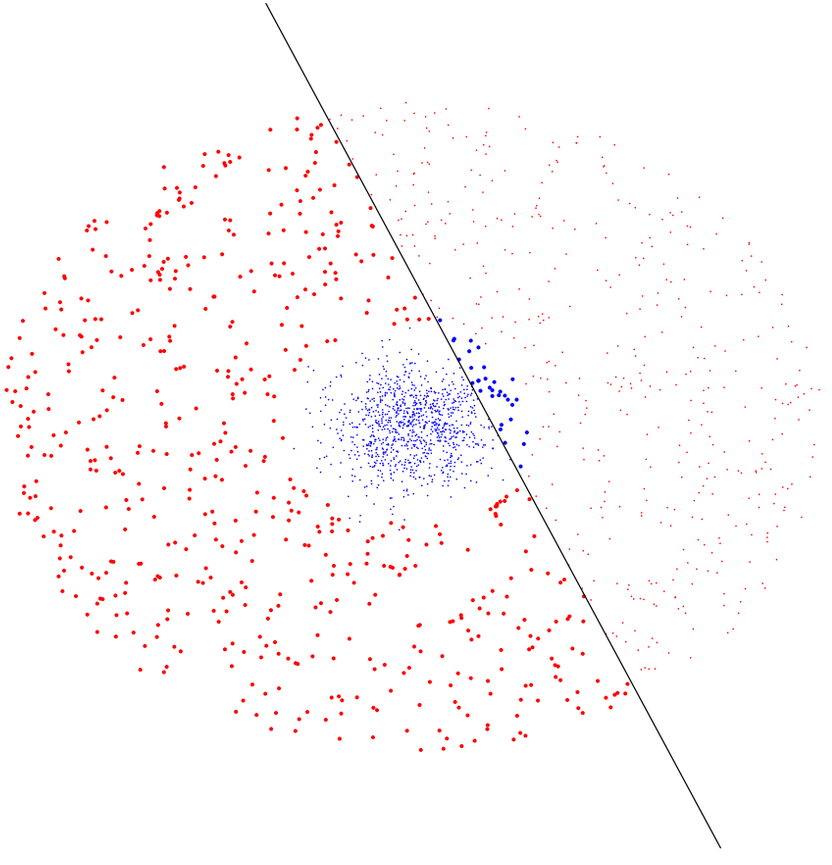
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

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

Output the final classifier:

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

$t = 1$



AdaBoost: Summary of the Algorithm

Initialization... $t = 1$

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

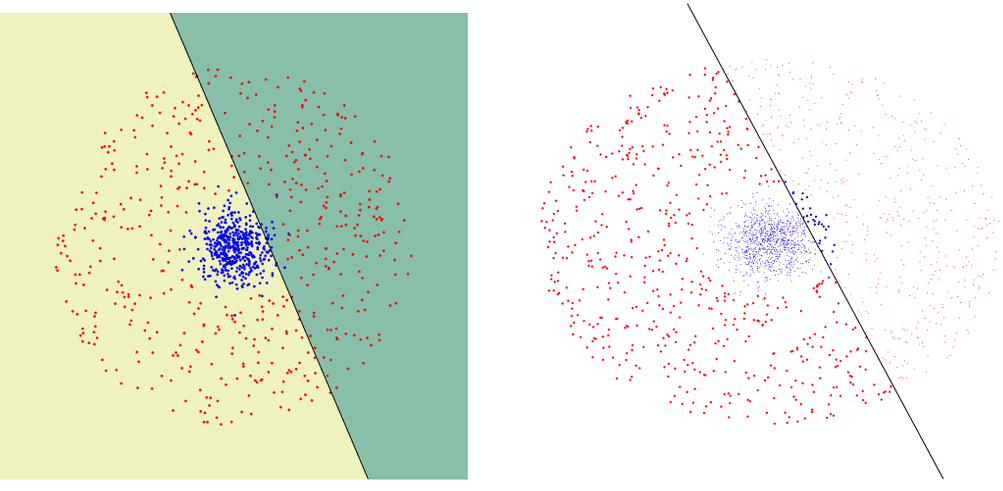
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

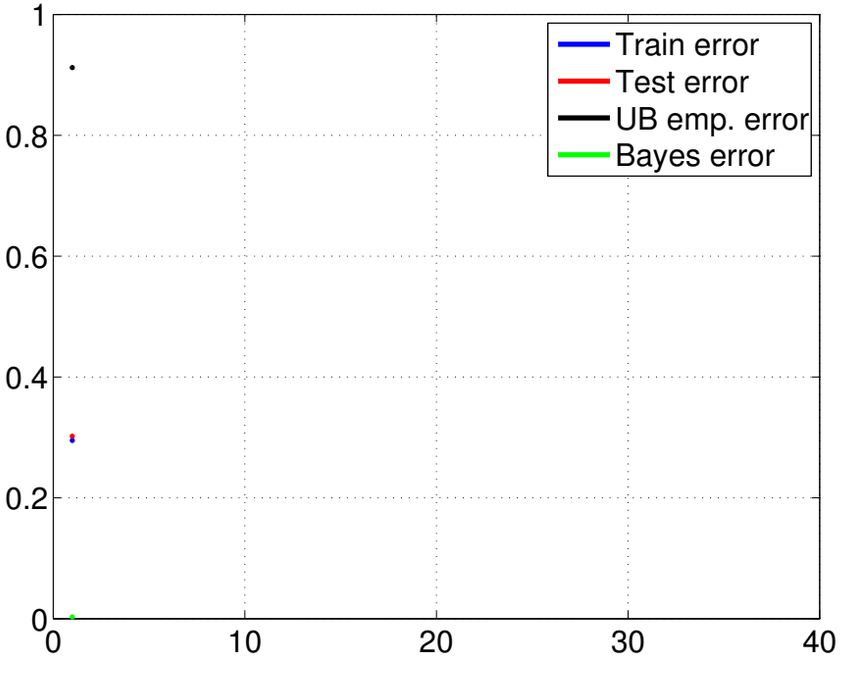
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

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

Output the final classifier:

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





AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

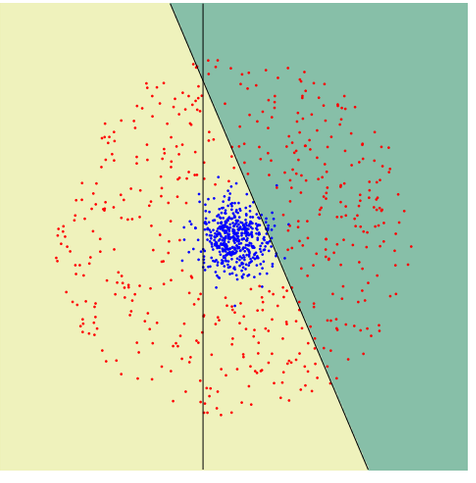
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

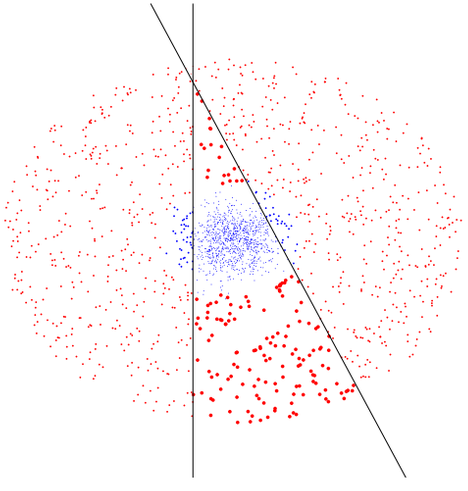
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

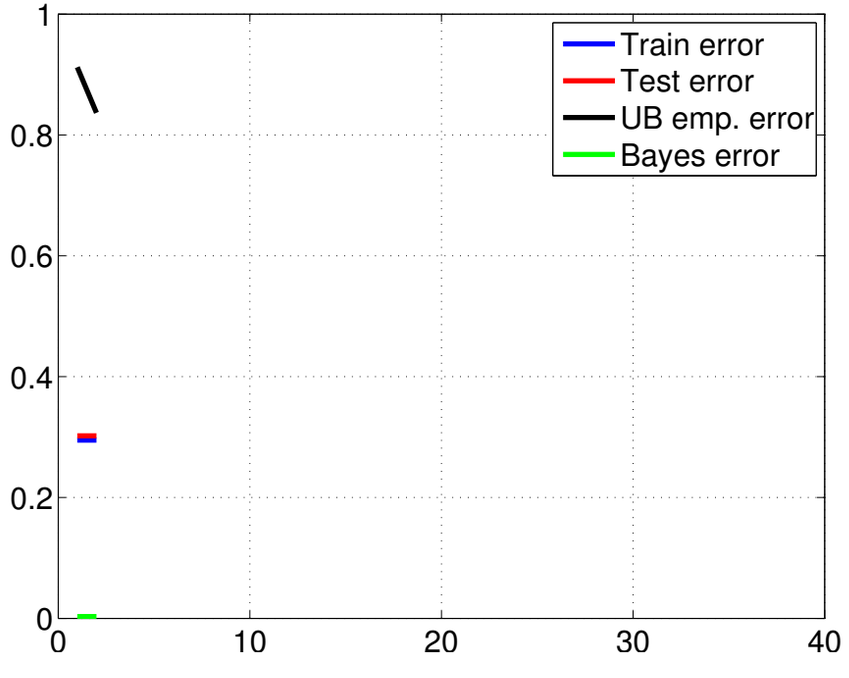
Output the final classifier:

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

$t = 2$







AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

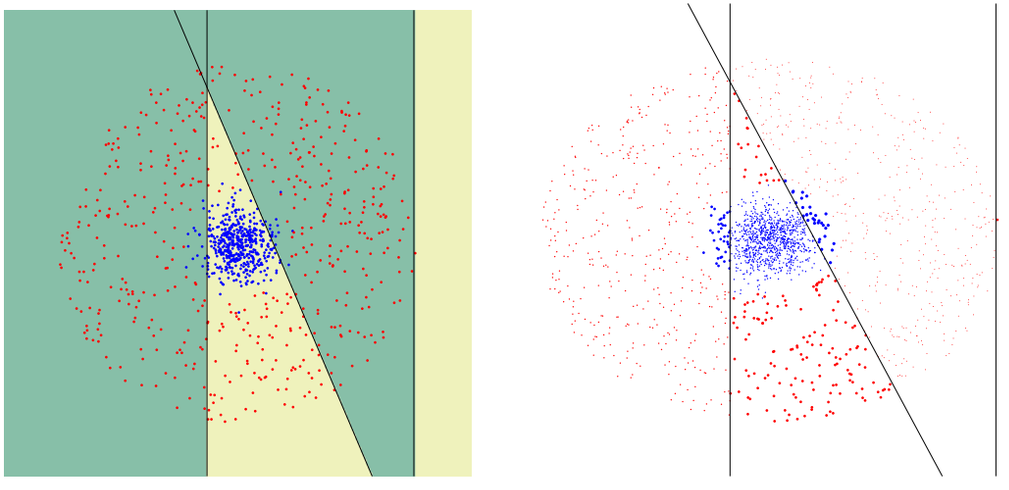
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

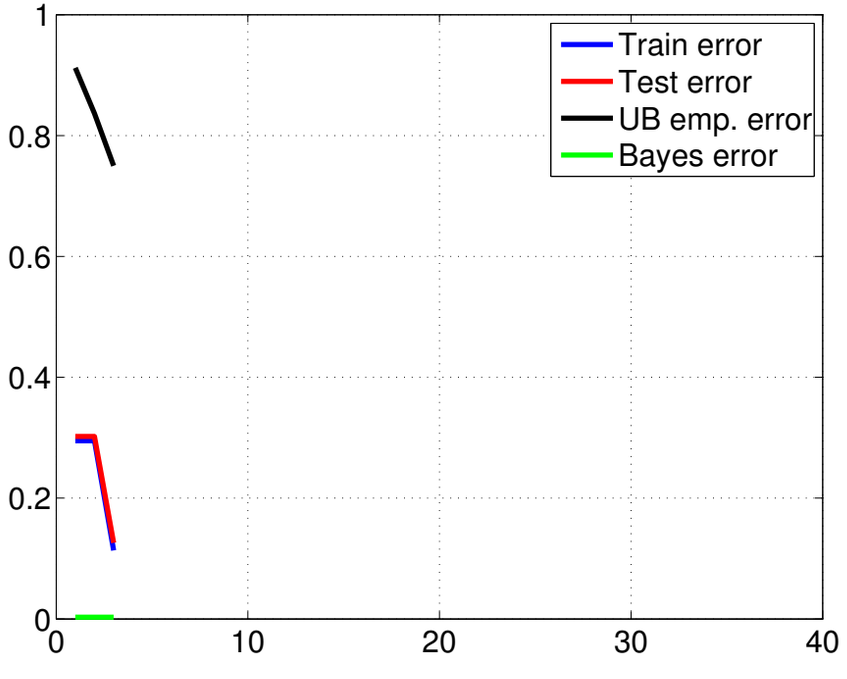
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

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

$t = 3$





AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

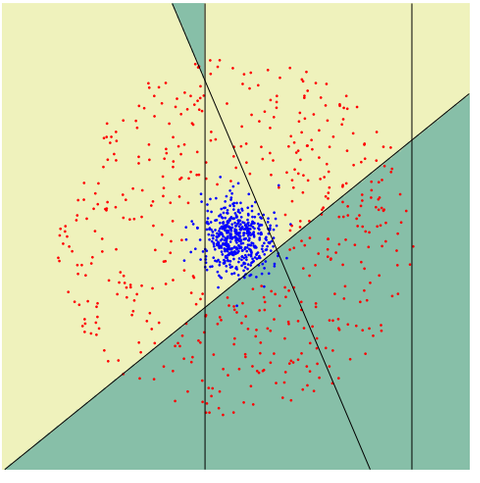
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

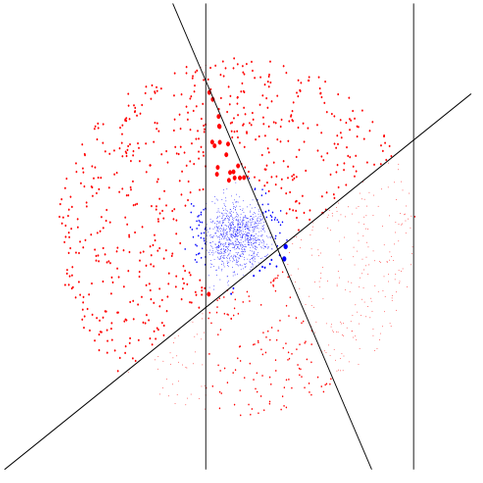
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

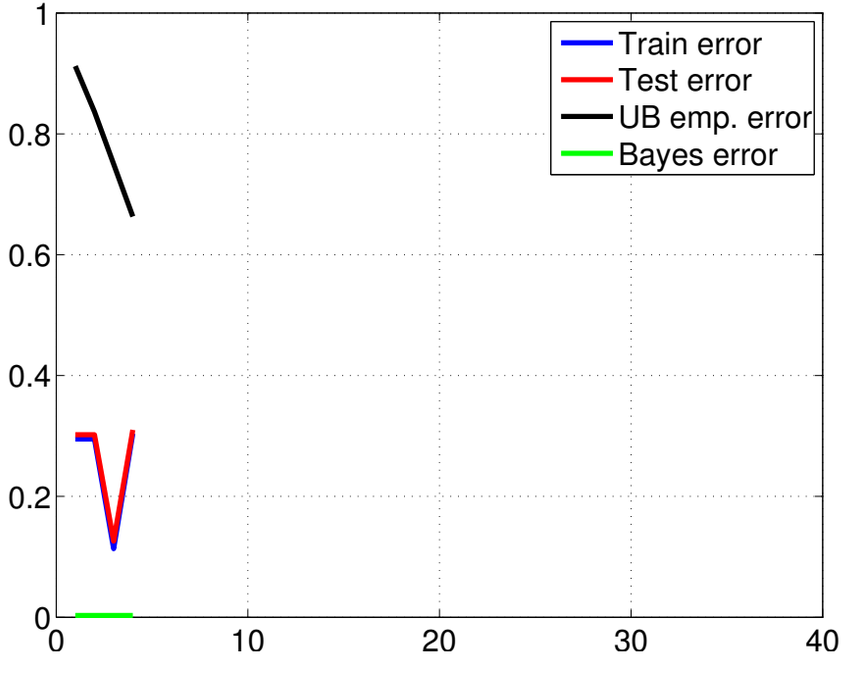
Output the final classifier:

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

$t = 4$







AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

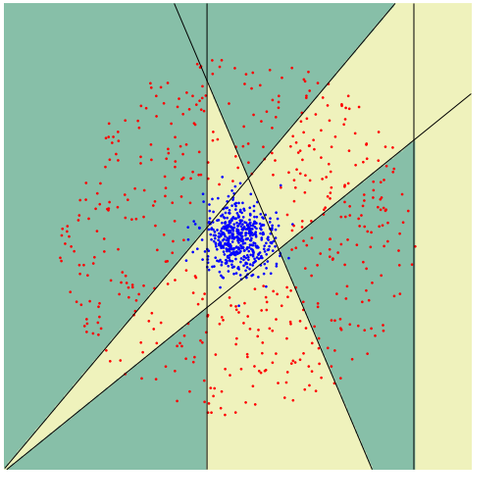
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

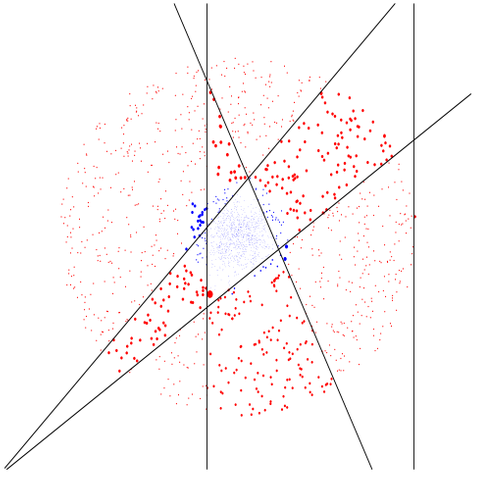
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

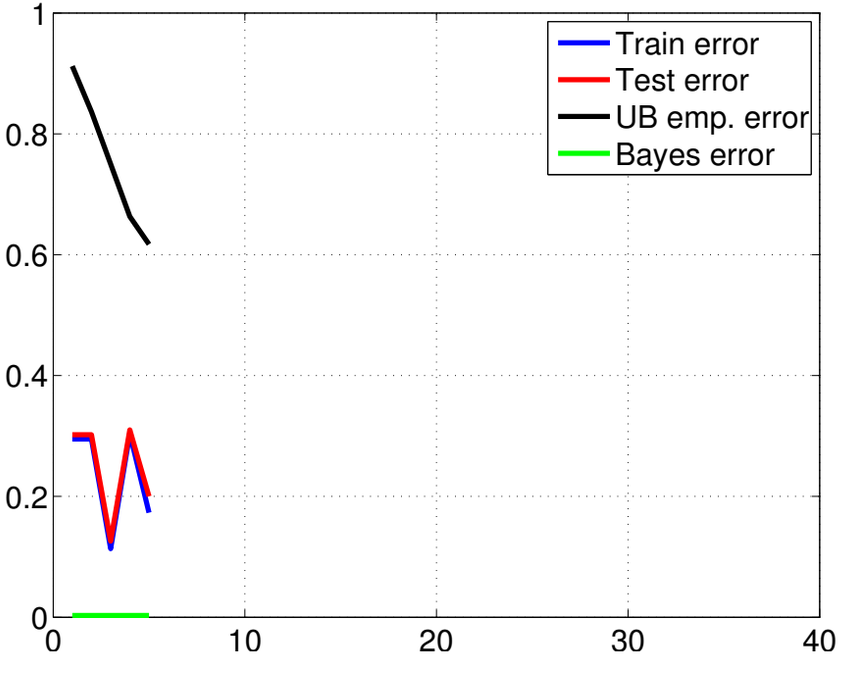
Output the final classifier:

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

$t = 5$







AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

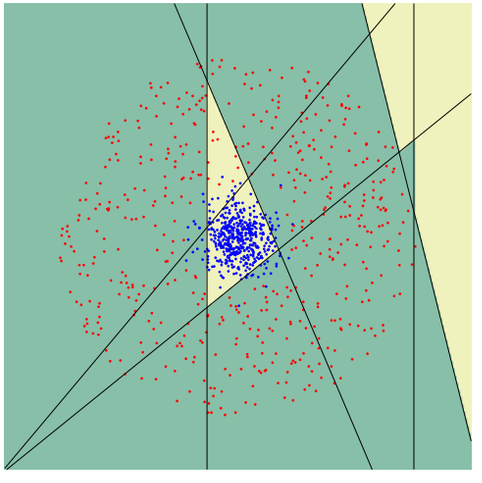
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

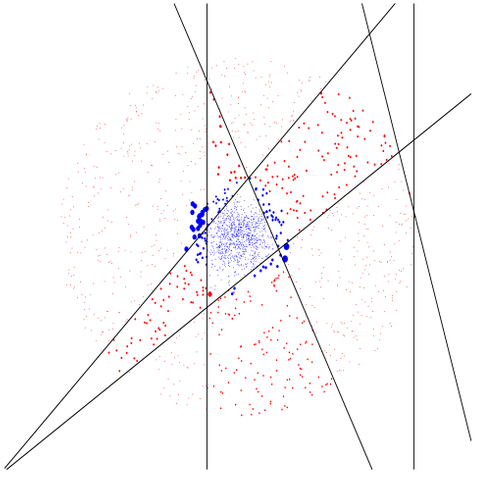
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

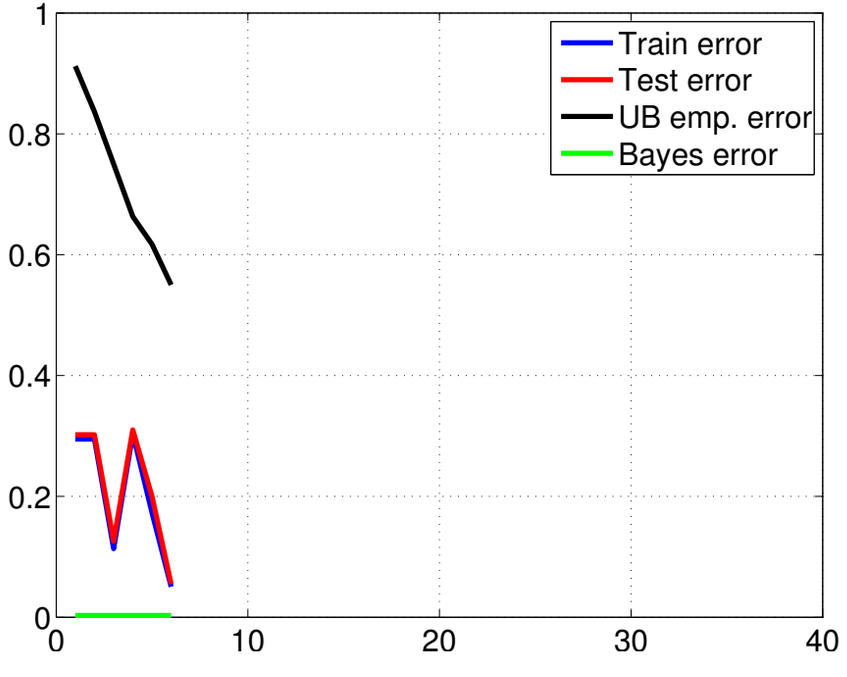
Output the final classifier:

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

$t = 6$







AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i)[\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

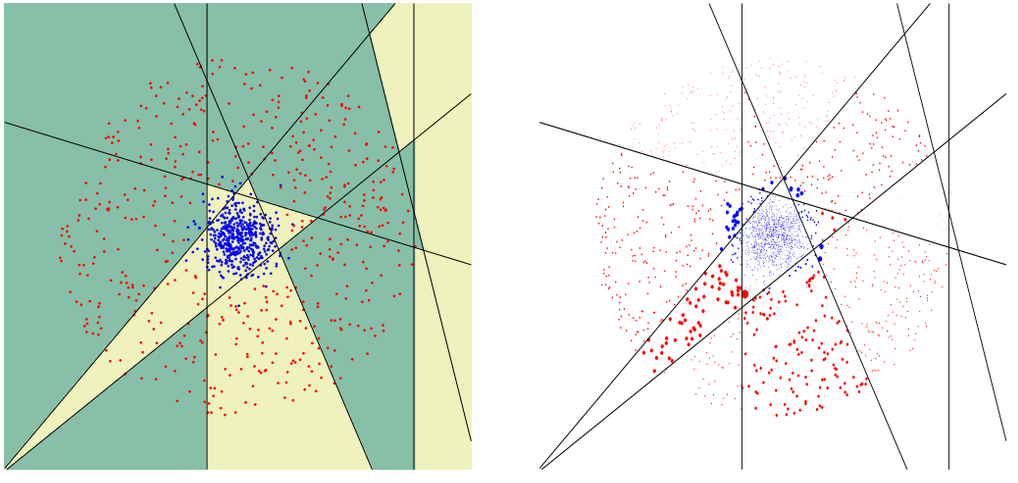
$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

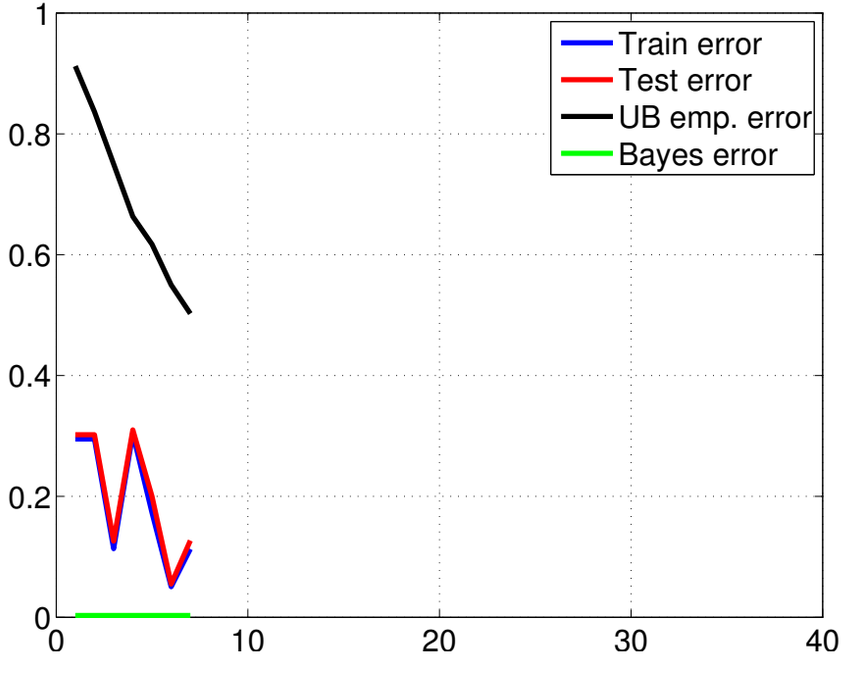
$$Z_t = \sum_{i=1}^{L} D_t(i)e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

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

$t = 7$





AdaBoost: Summary of the Algorithm

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$;

$$\epsilon_j = \sum_{i=1}^{L} D_t(i) [\![y_i \neq h_j(x_i)]\!]$$

- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

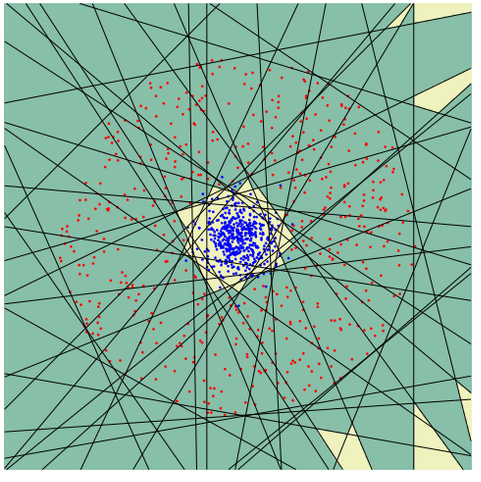
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

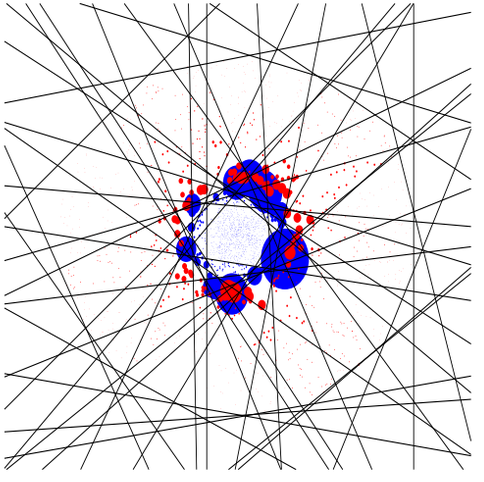
$$Z_t = \sum_{i=1}^{L} D_t(i) e^{y_i \alpha_i h_t(x_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

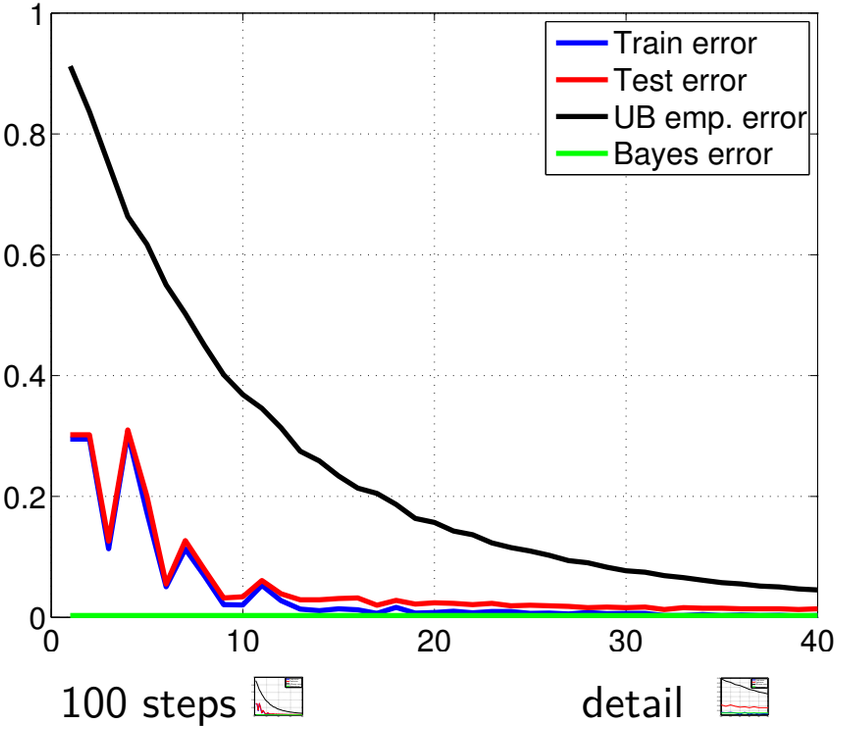
Output the final classifier:

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

$t = 40$







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

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

AdaBoost

- Selects measurements h_t and combines them into classifier
- The output of the AdaBoost algorithm is a classification function

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

- Classification is given by $sign(H_T(x))$

Measurement selection and ordering

$\Rightarrow$ The outputs of selected weak classifiers are taken as measurements used in SPRT.

$h_1, h_2, \ldots$ ordered sequence of measurements

Likelihood Ratio Estimation

Wald's SPRT requires the knowledge (an estimate) of the likelihood ratio

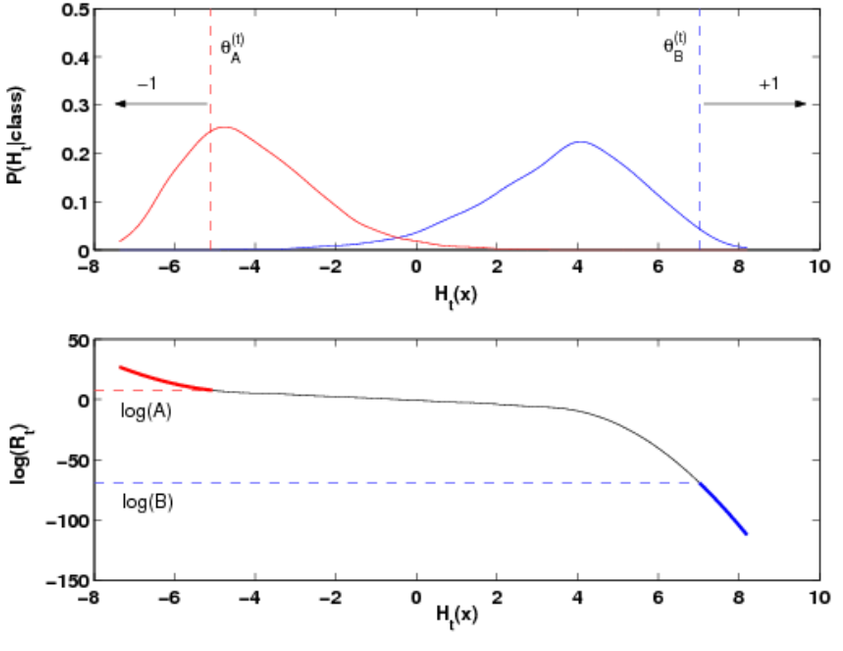
$$\hat{R}_t(x) = \frac{p(h^{(1)}(x), ..., h^{(t)}(x)|y = -1)}{p(h^{(1)}(x), ..., h^{(t)}(x)|y = +1)}$$

The weak classifiers cannot be treated as statistically independent

$\Rightarrow$ difficult multi-dimensional density estimation.

WaldBoost Approach.
The global minimiser of AdaBoost learning is

$$\tilde{H}(x) = -\frac{1}{2}\log R(x) + \frac{1}{2}\log\frac{P(+1)}{P(-1)}.$$



A one-to-one mapping exists between $\tilde{H}(x)$ and $R(x)$
$\Rightarrow$ the ratio is estimated by

$$\hat{R}_t(x) = \frac{p(H_t(x)|y = -1)}{p(H_t(x)|y = +1)}.$$

WaldBoost – Learning

Input: $(x_1, y_1), ..., (x_l, y_l)$; $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$,
desired final false negative rate α and false positive rate β.
Set $A = (1 - \beta)/\alpha$ and $B = \beta/(1 - \alpha)$

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

1. Choose best h_t as in AdaBoost
2. Estimate the likelihood ratio R_t
3. Find thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$
4. Throw away samples from training set for which $H_t \geq \theta_B^{(t)}$ or $H_t \leq \theta_A^{(t)}$
5. (Optional.) Bootstrap: Sample new data into the training set

end

Output: strong classifier H_T and thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$.

WaldBoost – Classification

Given: H_T, $\theta_A^{(t)}$, $\theta_B^{(t)}$, γ
Input: a classified object x

For $t = 1, \ldots, T$ *(SPRT execution)*
  If $H_t(x) \geq \theta_B^{(t)}$, classify x to the class $+1$ and terminate
  If $H_t(x) \leq \theta_A^{(t)}$, classify x to the class -1 and terminate
end

If $H_T(x) > \gamma$, classify x as $+1$. Classify x as -1 otherwise.

Optionally, if the final stage is reached very rarely, make the final decision by a classifier of arbitrary complexity, using $h_t(x)$ or in general x.

Applying WaldBoost 1: Face Detection

Face detection specific properties

- Highly unbalanced face and background class sizes and complexities.
- Missed detection more serious than false positive.

WaldBoost learning for face detection

We therefore set $\beta = 0$ in WaldBoost. Thus

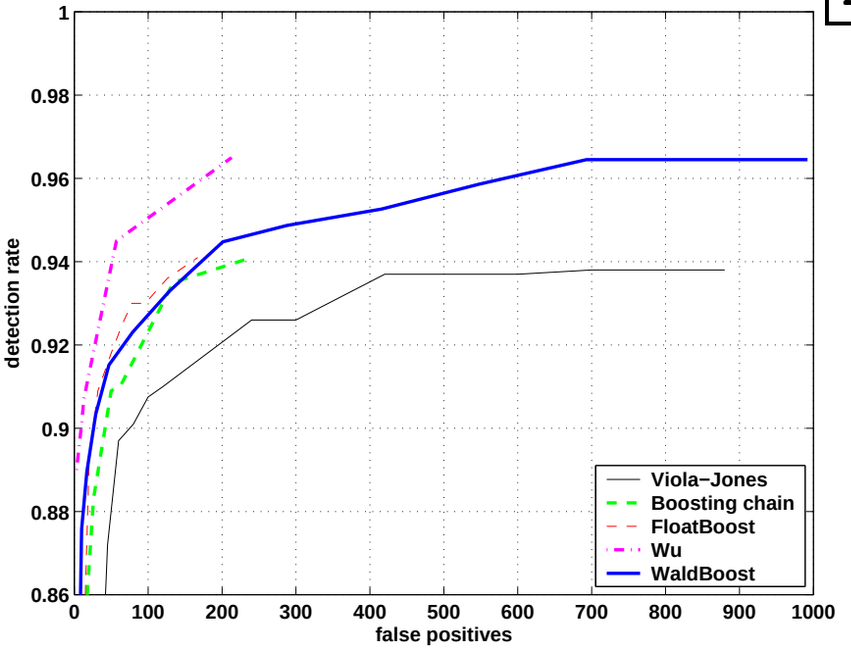
$$A = \frac{1-0}{\alpha} = \frac{1}{\alpha}, \qquad B = \frac{0}{1-\alpha} = 0$$

and the SPRT strategy becomes

$$S_t^* = \begin{cases} +1, & R_t \leq 0 \\ -1, & R_t \geq 1/\alpha \\ \sharp, & 0 < R_t < 1/\alpha \end{cases}$$

$\Rightarrow$ Since $R_t > 0$, only allowed decision is the classification to the background class
$\Rightarrow$ Only background class pruned during training

Face Detection Results

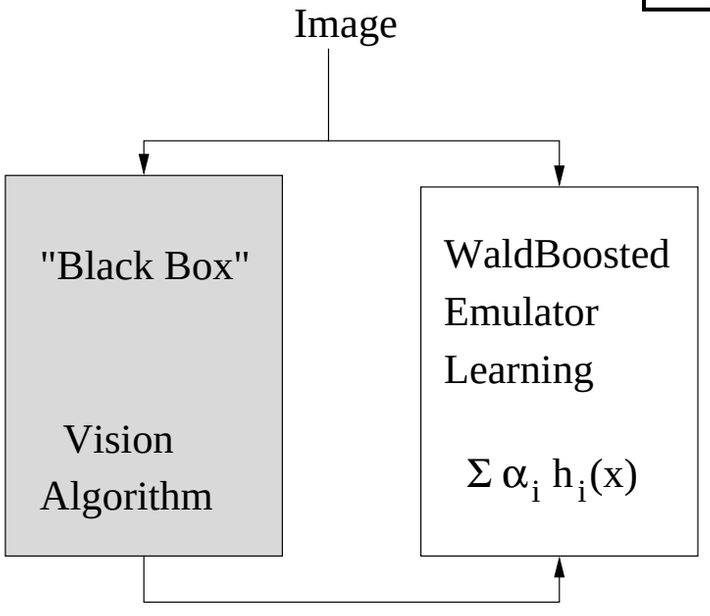


Method	**WB**	VJ	FloatBoost	Boosting chain	Wu
$\#wc$	**400**	4297	2546	700	756
$\bar{T}_S$	**10.84**	8	18.9	18.1	N/A

- WaldBoost results in the fastest classifier (see $\bar{T}_S$) among those with comparable detection rate.
- Least number of weak classifiers needed in WaldBoost.
- Detection rates are comparable to the state-of-the-art methods.

Applying WaldBoost 2: Emulating a Black Box

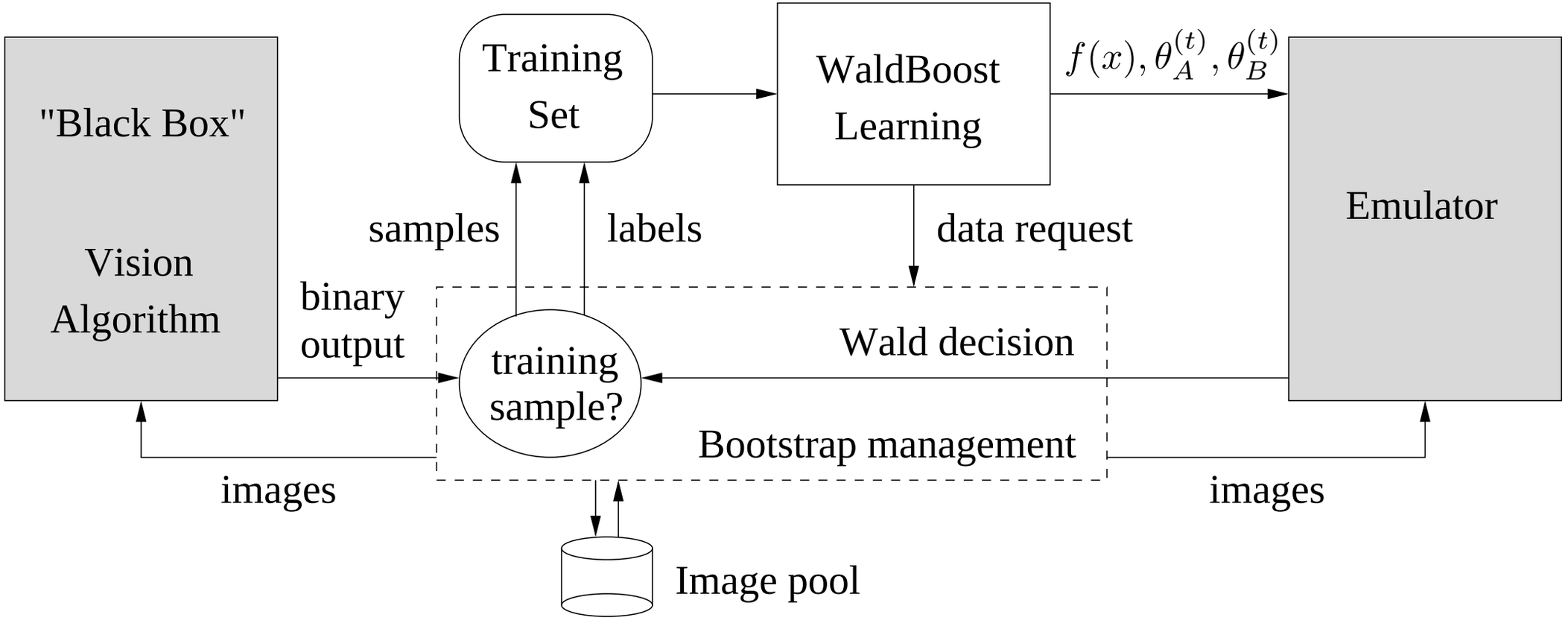
- Let's assume an executable of a binary-valued (and slow) vision algorithm and suitable set of efficient weak classifiers $\mathcal{H}$ "suitable" for the algorithm is given.
- WaldBoost is trained on a set of examples generated by the "black box". Optimizing time to decision given limits on false positive and false negative rates
- Result: fast version of the black box algorithm



Notes:

- If the emulator is not fast enough, search for better $\mathcal{H}$.
  This is very different from spending man-months optimizing and/or finding fast approximation of a process (e.g. SIFT v. SURF)!
- Your own MATLAB implementation can serve as the "black box".
- Through false postives and negatives, the most complicated (from the computational point of view) examples are dropped.

Applying WaldBoost 2: Emulating a Black Box Algorithm



Applying WaldBoost 2: Emulating Kadir and Hessian-Laplace Interest Point Detectors

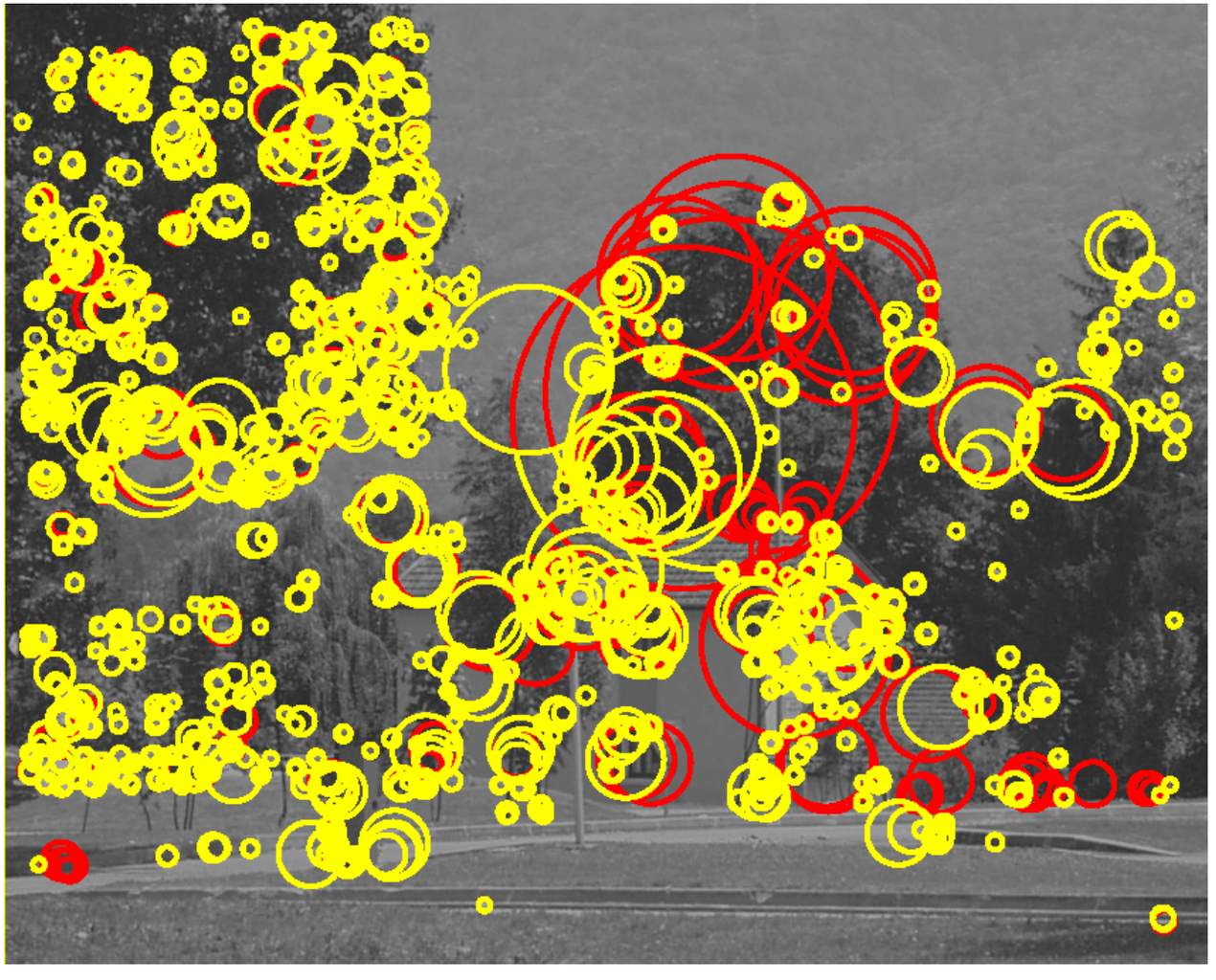
Motivation:

- Executables of both methods available at `robots.ox.ac.uk`
- Both detectors are scale-invariant (not affine), which is easily implemented via a scanning window + a sequential test.
- Standard testing protocol exists [Mikolajcyk et al. IJCV05]
- Kadir's detector very slow (100x slower than Difference of Gaussians).
- Hessian-Laplace is close to state of the art.

Implementation:

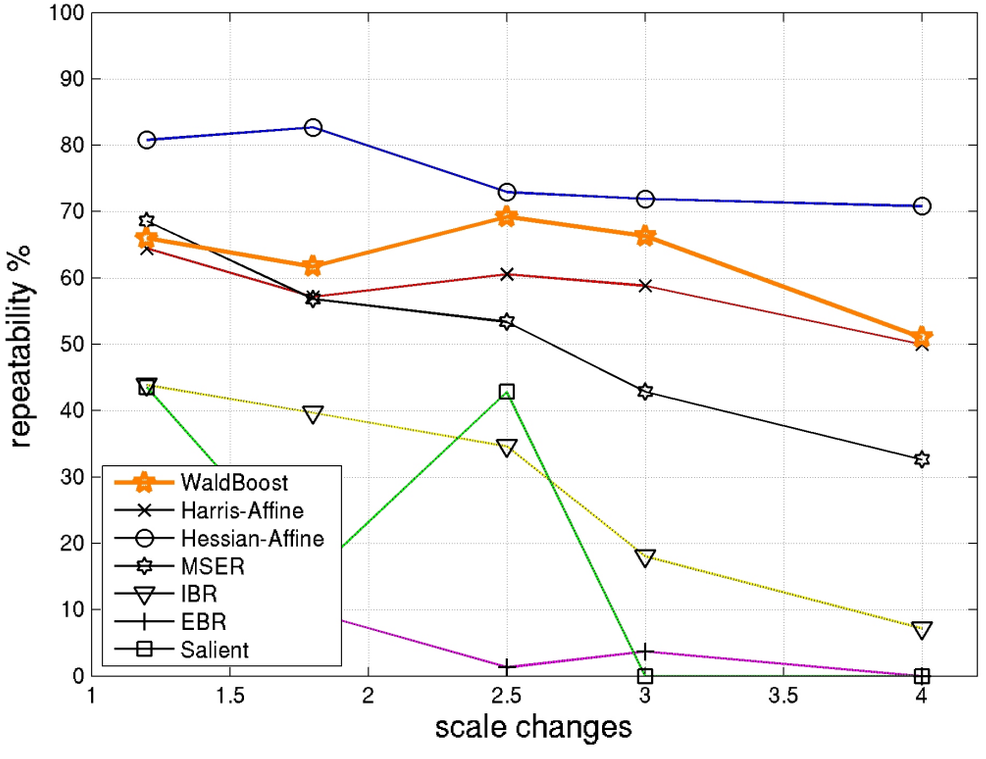
- Choice of $\mathcal{H}$: various filters computable with integral images of intensity and intensity squared: difference of rectangular regions, variance in a window.
- False negative rate $\alpha = 0.2$, false negative rate $\beta = 0.2$.

Emulating Harris-Laplace Detector - Results

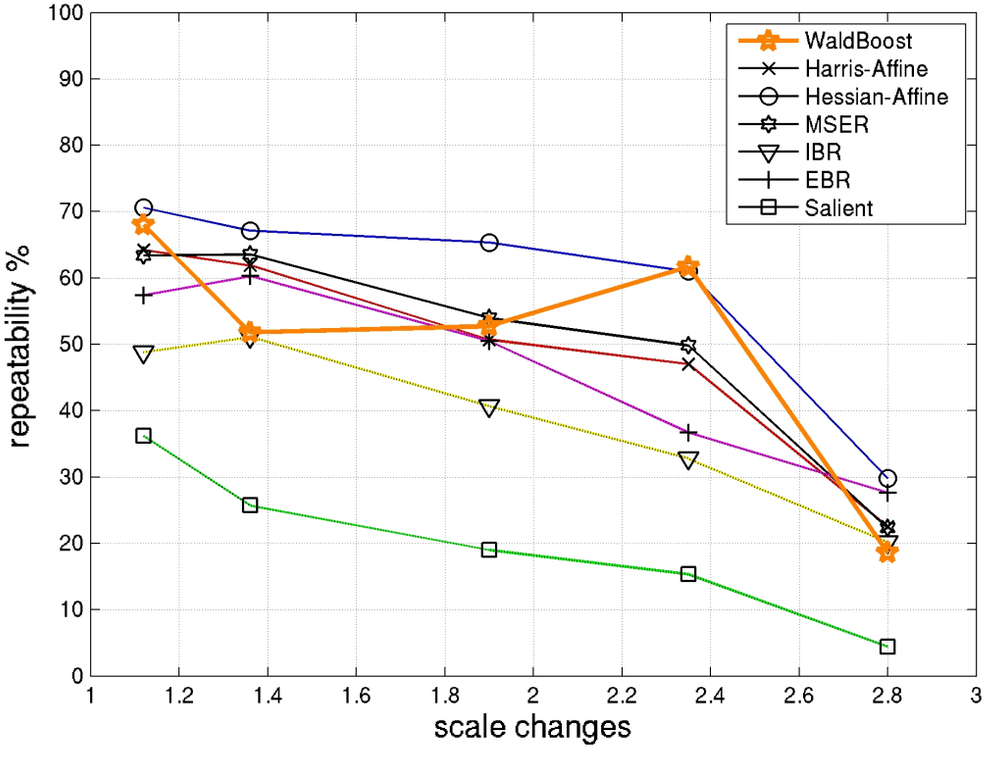


- Yellow: repeated dections - 85%
- Red: original detections not found by the WaldBoost detector

Emulating Harris-Laplace Detector - Results

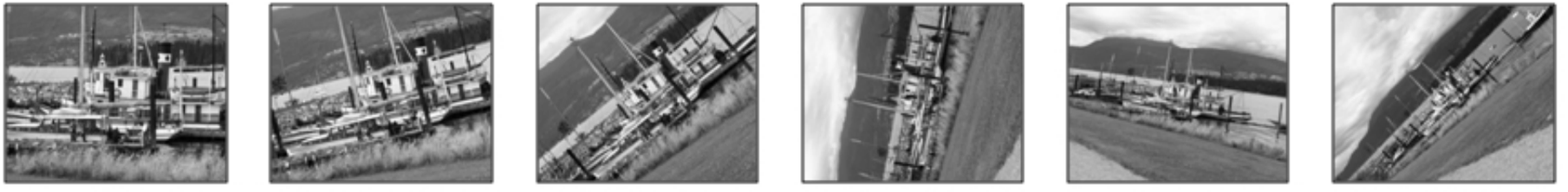


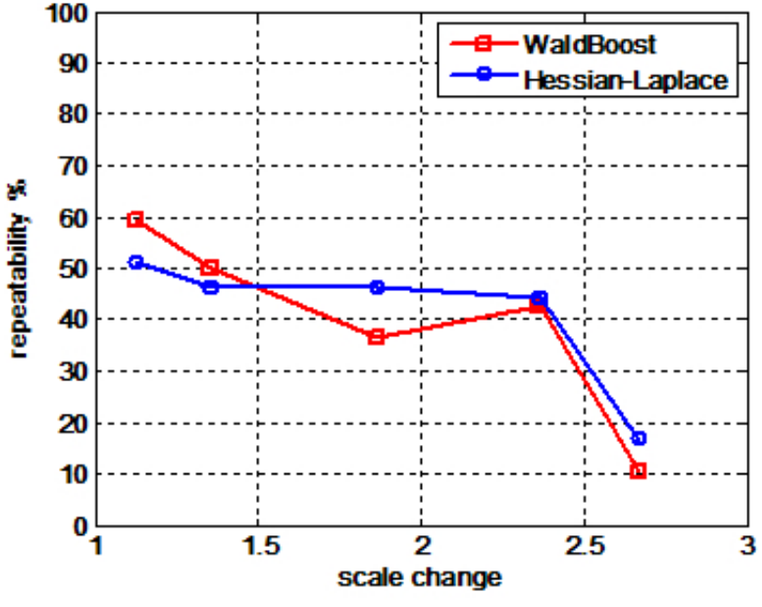
Repeatability score of the WaldBoost detector for Bark sequence (overlap 40%, norm. size = 30 pixels).

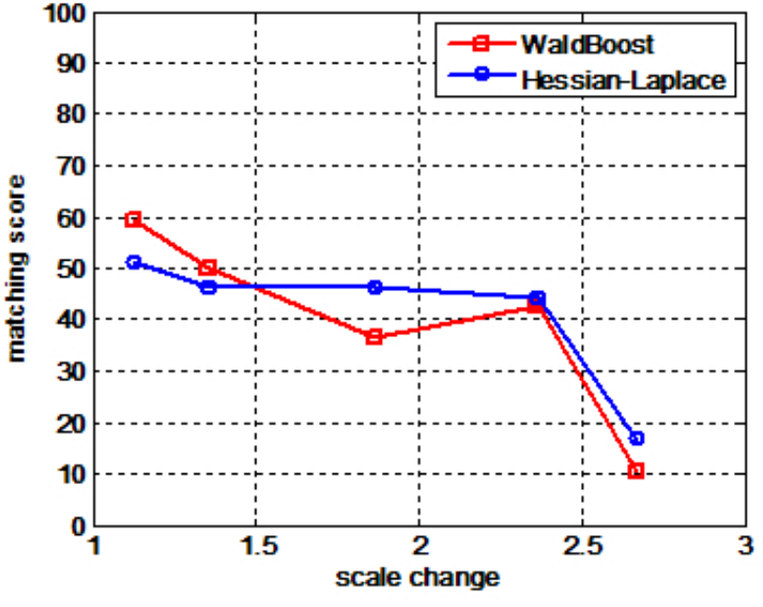


Repeatability score of the WaldBoost detector for Boat sequence (overlap 40%, norm. size = 30 pixels).

Emulating Harris-Laplace Detector - Results



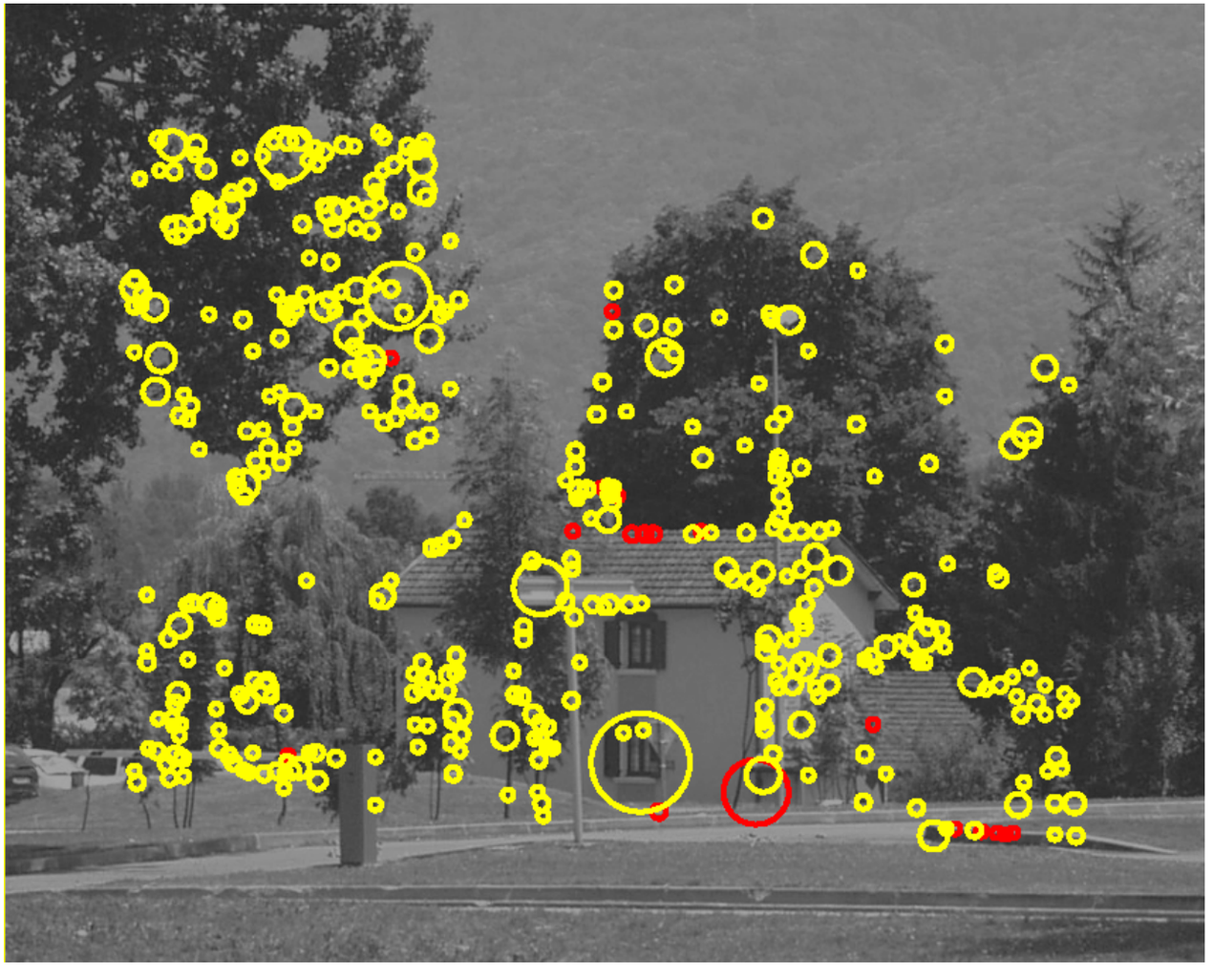




Waldboost preformance:

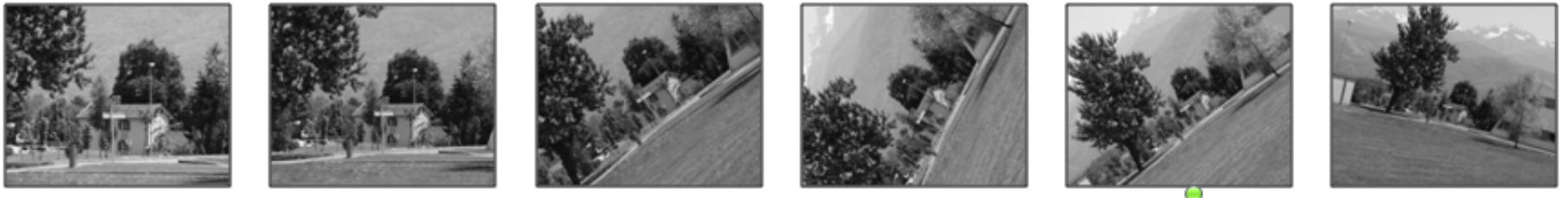
- Higher number of correspondences (not shown).
- Higher or similar percentage of correct matches.
- Similar speed. For a 850x680 image:
  Original: 1.3 secs v. WaldBoost: 0.7 secs

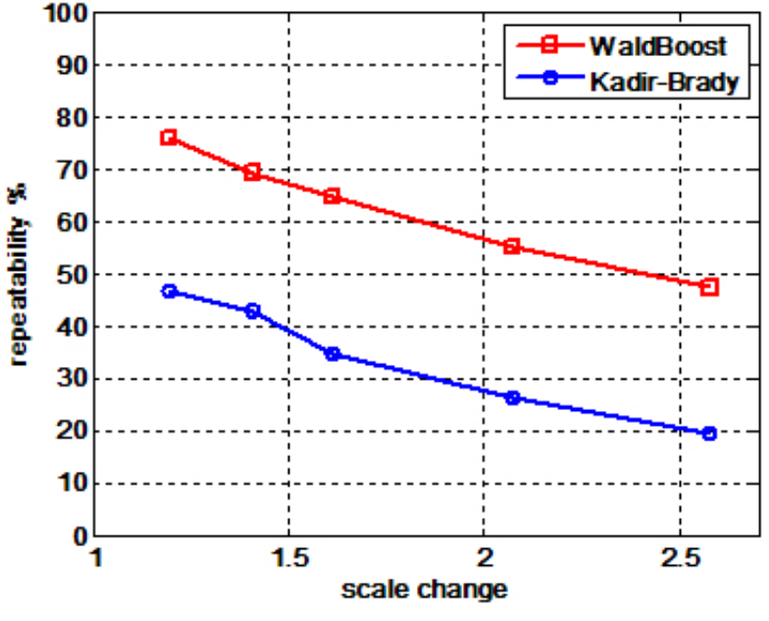
Emulating Kadir Detector - Results

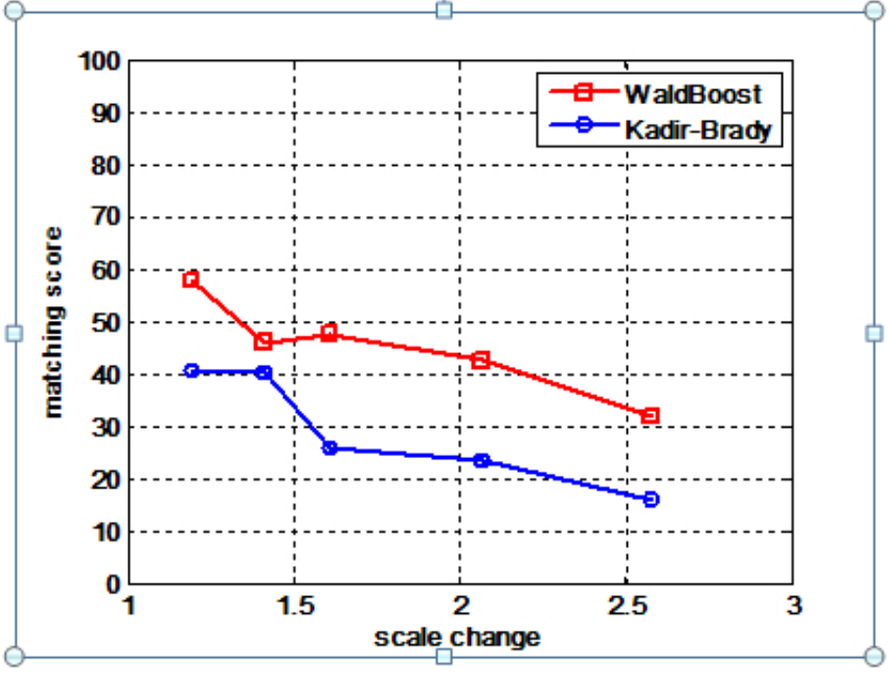


- Yellow: repeated dections - 96%
- Red: original detections not found by the WaldBoost detector

Emulating Kadir Detector - Results







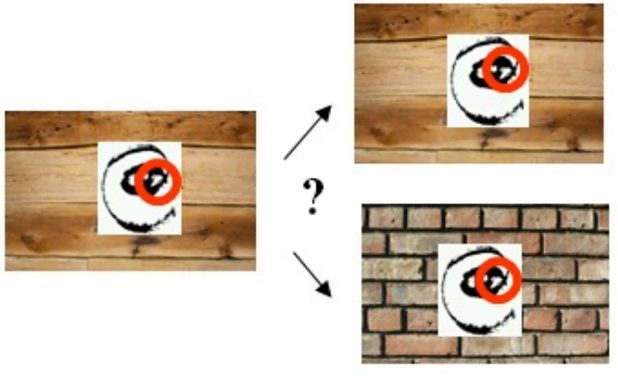
Waldboost performance:

- Higher number of correspondences and correct matches.
- Signfiicant speed-up ($\approx 10^2$). For a 850x680 image:
  Original: 1 min 44 sec v. WaldBoost: 0.76 sec

Applying Wald #3: Sequential Correspondence Selection by Cosegmentation (Cech, Matas, Perdoch, PAMI 2009)

Establishing correspondences, Standard solution (Lowe, IJCV 2004):
(steps 1. and 2 carried out in both images independently)

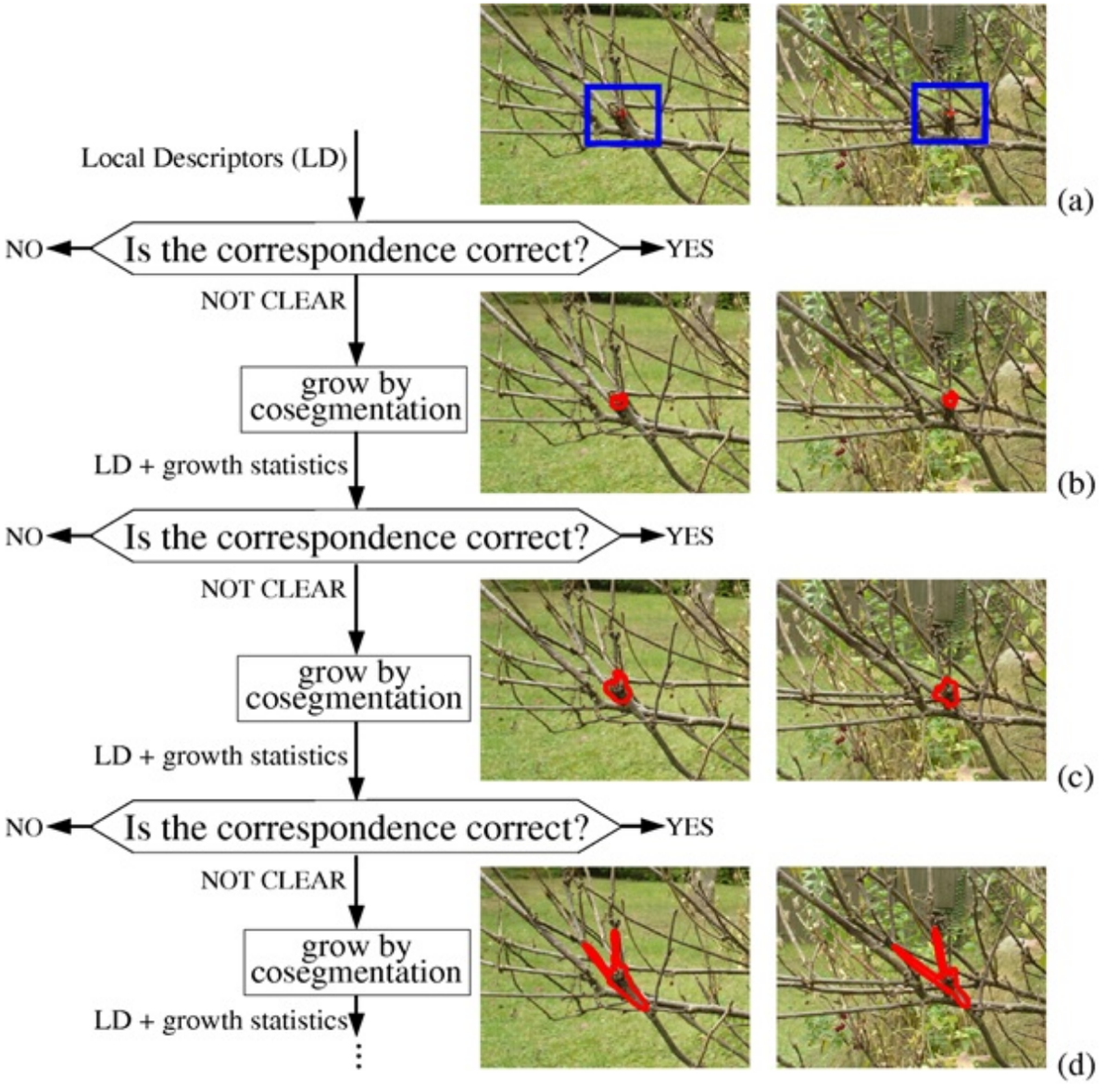
1. Detect covariant regions (Diff. of Gaussians, Hessian, MSER, Harris, ...)
2. Describe regions with a SIFT from an "arbitrary measurement area"
3. Select regions matches with SIFT ratio ≤ 0.8



Drawbacks:

1. fix-shaped measurement regions
2. independent detection, normalization (geometric, photometric)

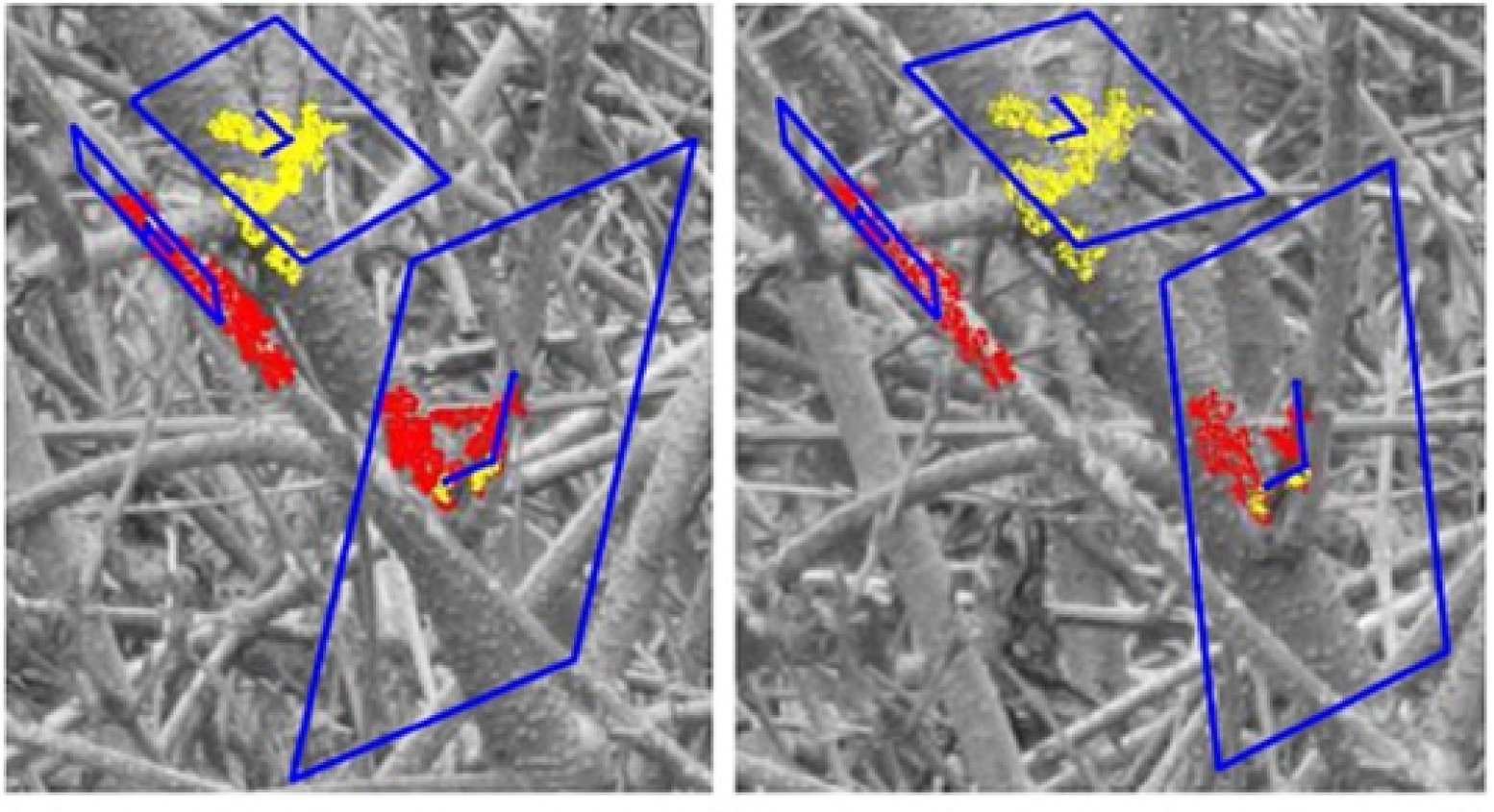
Sequential Correspondence Verification (SCV) by Cosegmentation



Sequential Correspondence Verification (SCV)

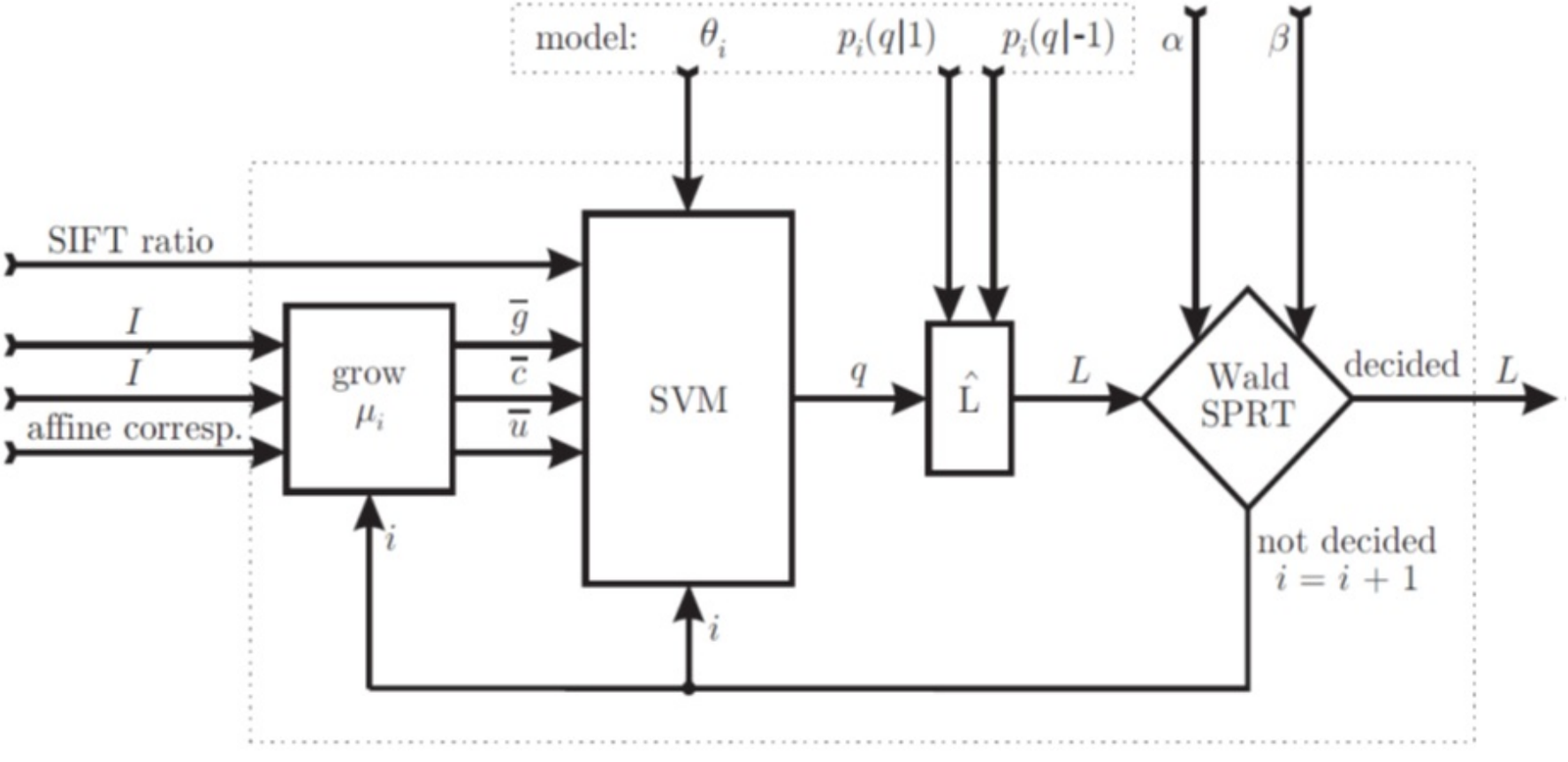
The method (similar to stereo matching or cosegmentation):

1. Intial matching seeds from SIFT correspondence
2. Apply a match growing process with a fixed number of steps
3. Collect statistics from the growth process:
   growth rate, *average correlation*, *non-bijectivity*
4. Use Wald's SPRT test to accept or reject the correspondence



Sequential Correspondence Verification (SCV)

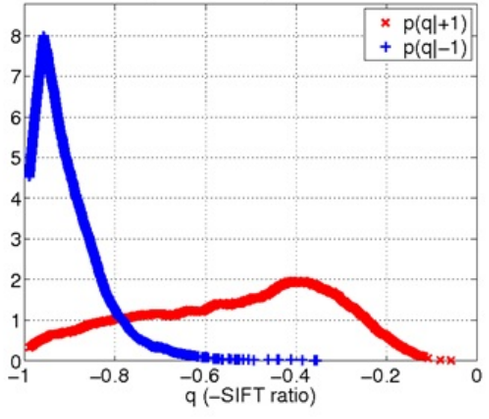
- Growing the correspondence collects more data, increases the discriminability.
- It costs time: a suitable task for sequential decision.



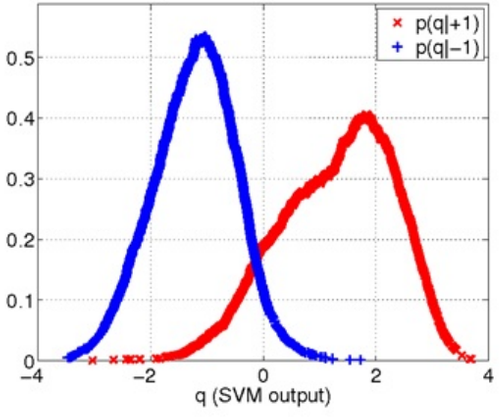
SCV - training

Distribution of the statistics for correct and incorrect correspondences learned from a training set of .

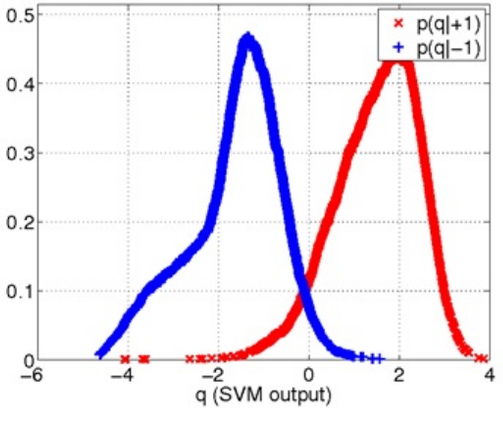
$i = 1 : \mu = 0$ (SIFT ratio only)



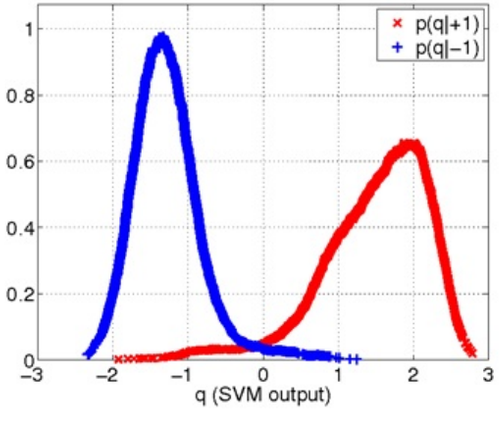
$i = 2 : \mu = 10$



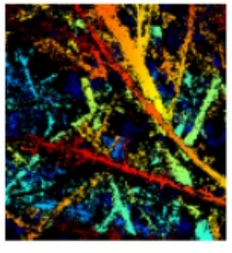
$i = 3 : \mu = 100$

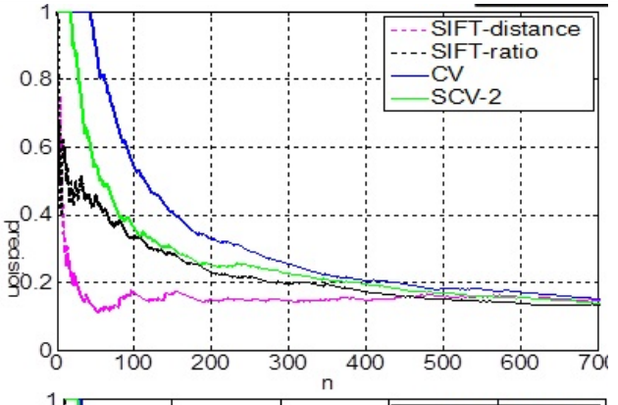


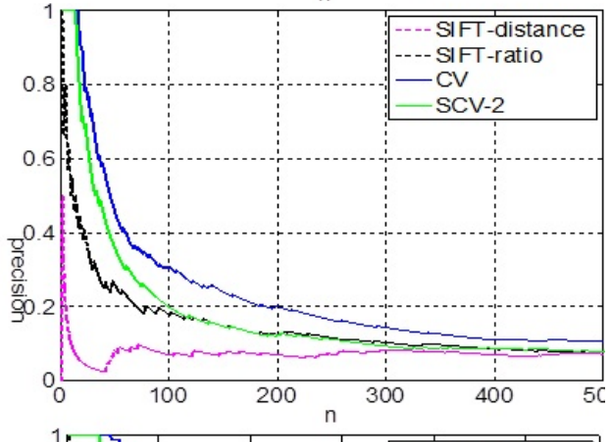
$i = 4 : \mu = 1000$

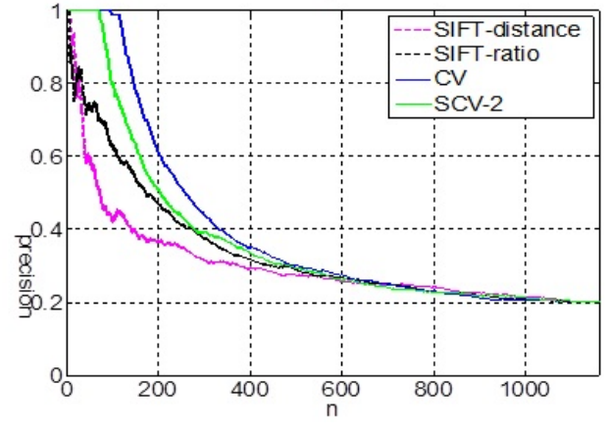


SCV - Results 1

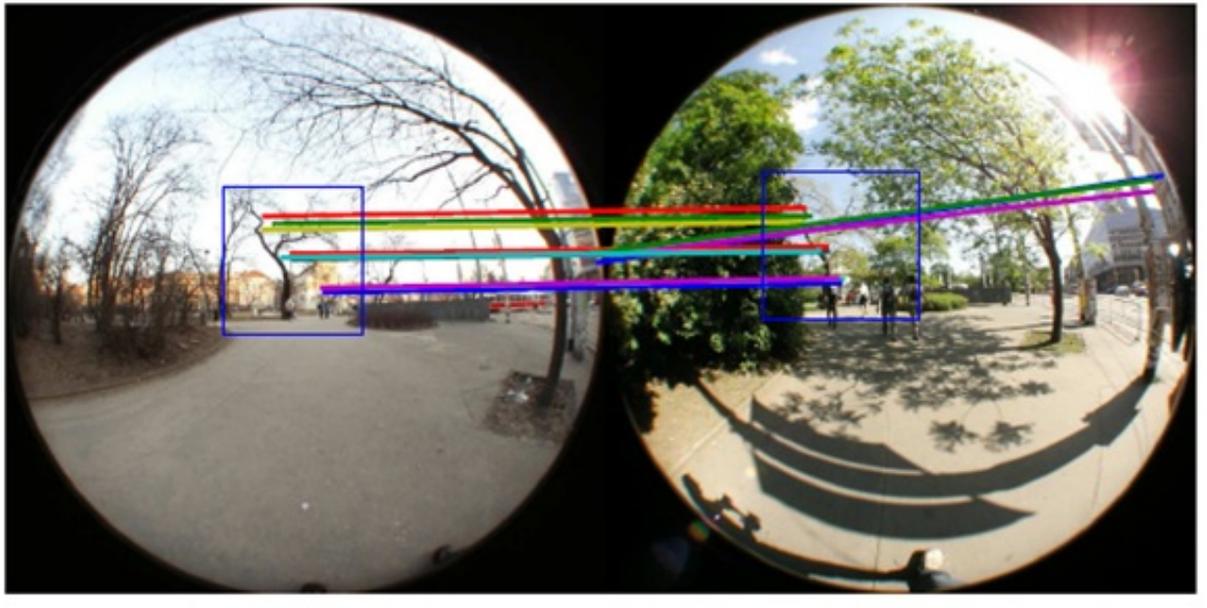




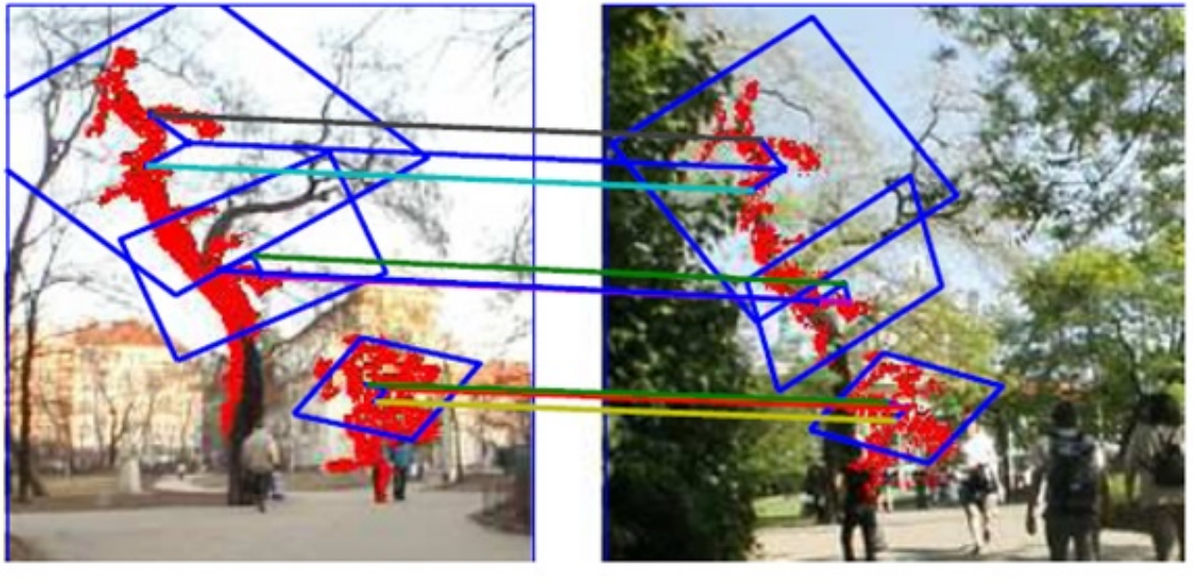




SCV - Results 2



zoom (grown regions)



SCV - Conclusions

- high precision and recall, significantly outperforms standard SIFT-ratio test
- very fast (0.5 sec / 1000 correspondences), controllable by accuracy requirements
- SCV time is small in comparison with the time spent by matching of tentative correspondences
- applicable before RANSAC
- the process generating tentative correspondences can be much more permissive: even 99% of outliers not a problem, correct correspondences recovered

Implementation available for download at `http://cmp.felk.cvut.cz/software/SCV`

Randomised RANSAC [Matas, Chum 02]

Repeat $k/(1-\alpha)$ times

1. Hypothesis generation
2. Model pre-verification $T_{d,d}$ test
   - Verify $d << N$ data points, reject
   - the model if not all d data points
   - are consistent with the model
3. Model verification

   Verify the rest of the data points

Time t_M

$\overline{m}_s \cdot V$

V – average number of data points verified

α – probability that a good model is rejected by $T_{d,d}$ test

$$t = \frac{k}{1-\alpha}(t_M + \overline{m}_s V)$$

WaldSAC

Repeat $k/(1\text{-}1/A)$ times
1. Hypothesis generation
2. Model verification
   use SPRT

Time
t_M

$\overline{m}_S \cdot C \log A$

$C \approx ((1-\delta)\log\frac{1-\delta}{1-\varepsilon} + \delta \log\frac{\delta}{\varepsilon})^{-1}$

$$t(A) = \frac{k}{(1-1/A)}(t_M + \overline{m}_S C \log A)$$

In sequential statistical decision problem decision errors are traded off for time. These are two incomparable quantities, hence the constrained optimization.

In WaldSAC, decision errors cost time (more samples) and there is a single minimised quantity, time t(A), a function of a single parameter A.

WaldBoost - Conclusions

- Time to decision vs. precision trade-off problem was formalised as a constrained optimisation problem.
- Wald's SPRT can be used directly in some computer vision problems
- WaldBoost overcomes limitations of SPRT to a priori ordered measurements and known multidimensional pdf's by using the AdaBoost algorithm.
- Adaboost learning converges (on an infinite training set) to the optimal decision function, but the ordering of weak classifiers is greedy.
- WaldBoost has been successfully applied in a number of applications
  - faces, vehicles, licence plates, ... ,
  - tracking (Grabner et al. ICPR 2008)
- The order of weak classifiers is fixed, selection of h_t depending on the *values* $x_1, x_2, \ldots, x_{k-1}$ not (yet) considered (leads to a decision tree with Wald's test)


m p

Segmentation as Selective Search for Object Recognition

Koen E. A. van de Sande* Jasper R. R. Uijlings† Theo Gevers* Arnold W. M. Smeulders*
*University of Amsterdam
Amsterdam, The Netherlands
†University of Trento
Trento, Italy
ksande@uva.nl, jrr@disi.unitn.it, th.gevers@uva.nl, a.w.m.smeulders@uva.nl

Abstract

For object recognition, the current state-of-the-art is based on exhaustive search. However, to enable the use of more expensive features and classifiers and thereby progress beyond the state-of-the-art, a selective search strategy is needed. Therefore, we adapt segmentation as a selective search by reconsidering segmentation: We propose to generate many approximate locations over few and precise object delineations because (1) an object whose location is never generated can not be recognised and (2) appearance and immediate nearby context are most effective for object recognition. Our method is class-independent and is shown to cover 96.7% of all objects in the Pascal VOC 2007 test set using only 1,536 locations per image. Our selective search enables the use of the more expensive bag-of-words method which we use to substantially improve the state-of-the-art by up to 8.5% for 8 out of 20 classes on the Pascal VOC 2010 detection challenge.

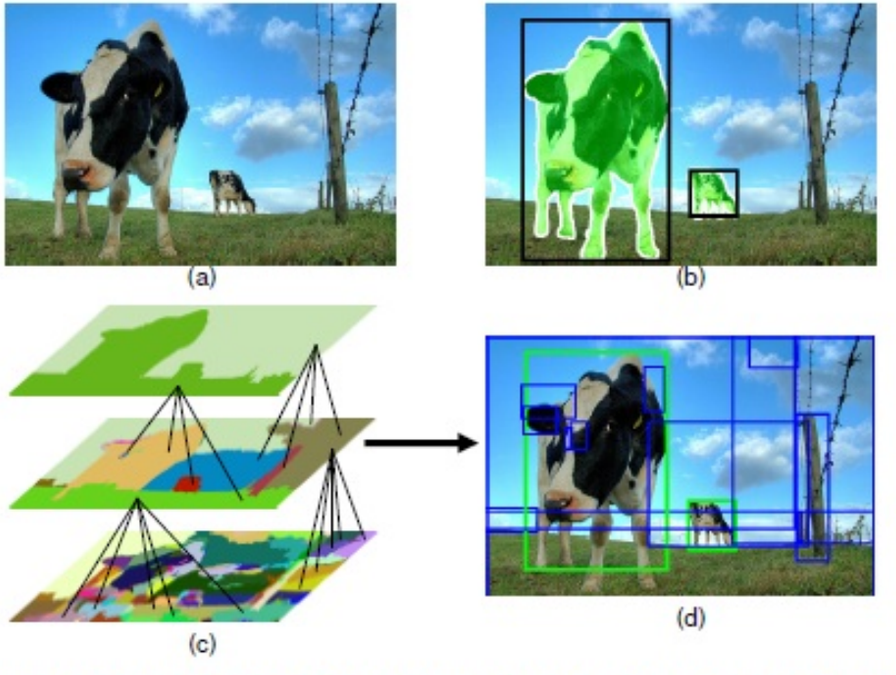
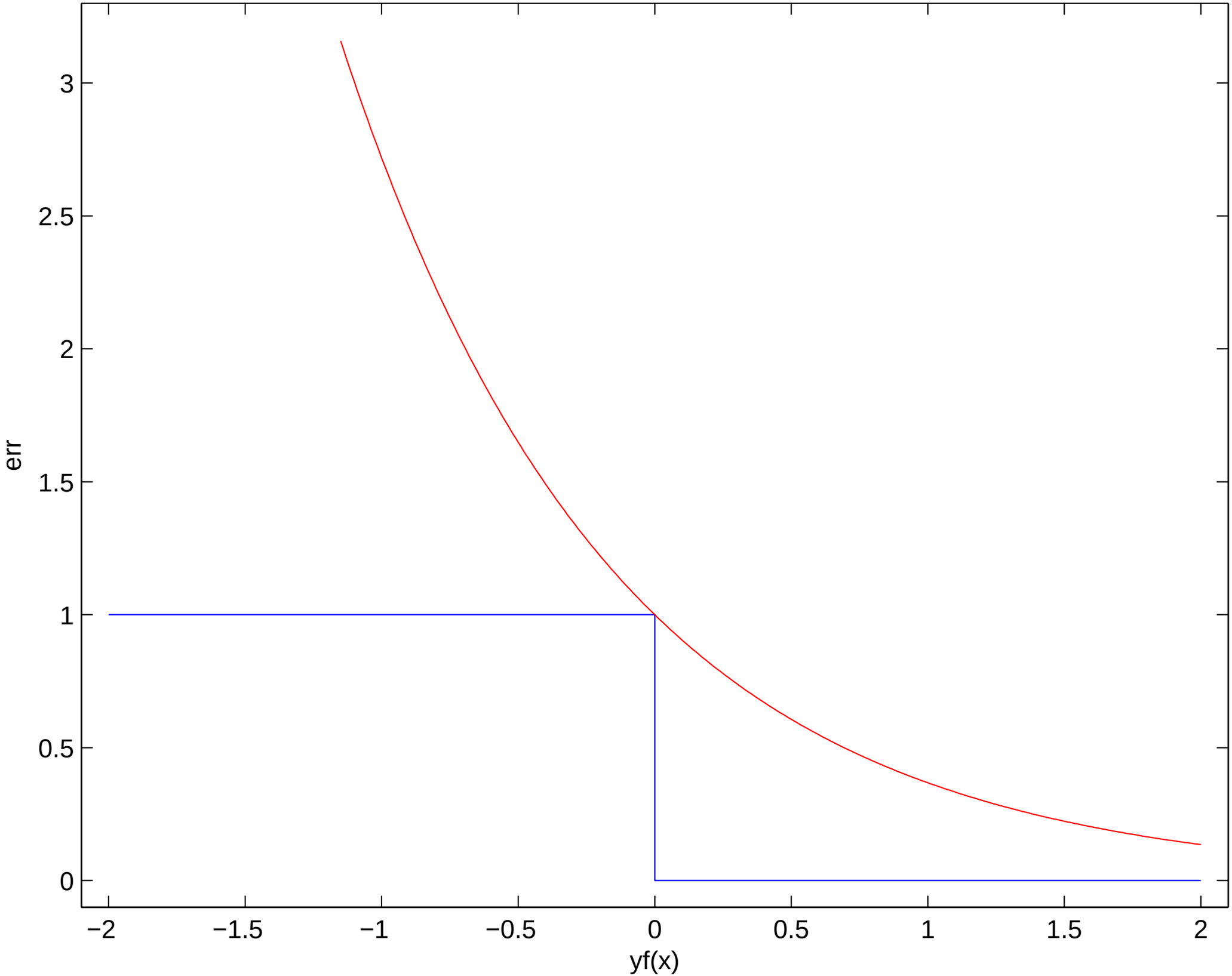
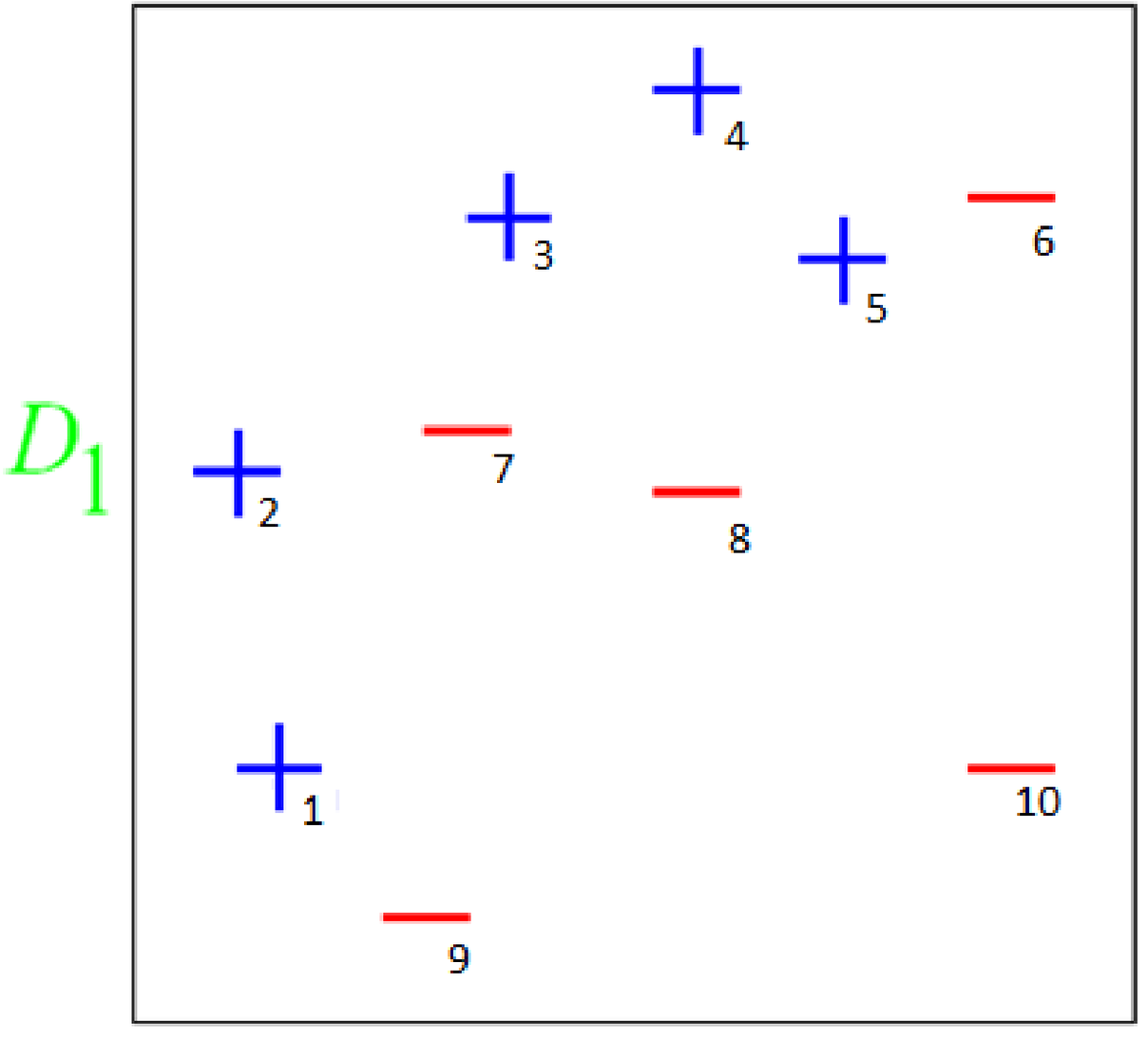
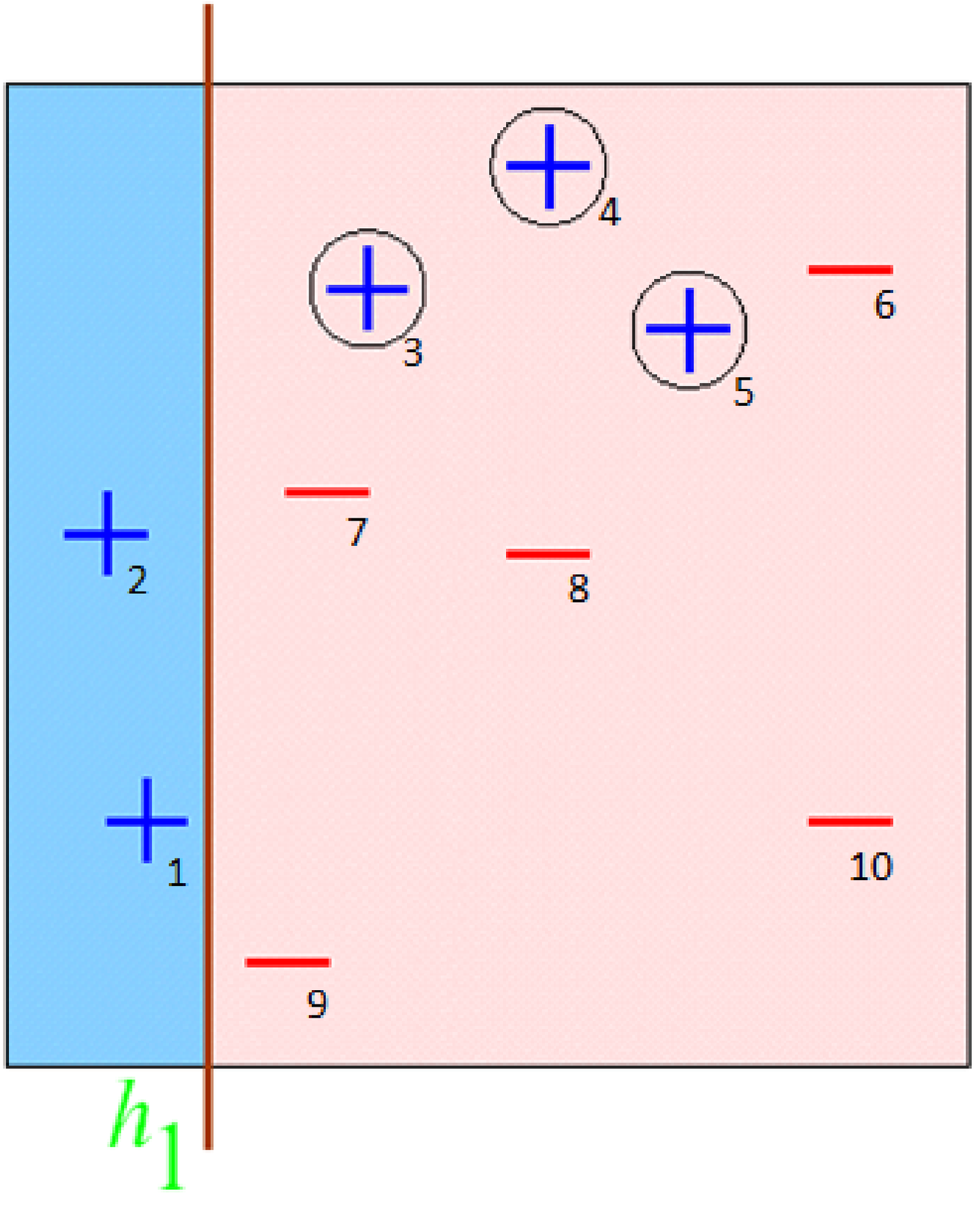


Figure 1. Given an image (a) our aim is to find its objects for which the ground truth is shown in (b). To achieve this, we adapt segmentation as a selective search strategy: We aim for high recall by generating locations at all scales and account for many different scene conditions by employing multiple invariant colour spaces. Example object hypotheses are visualised in (d).

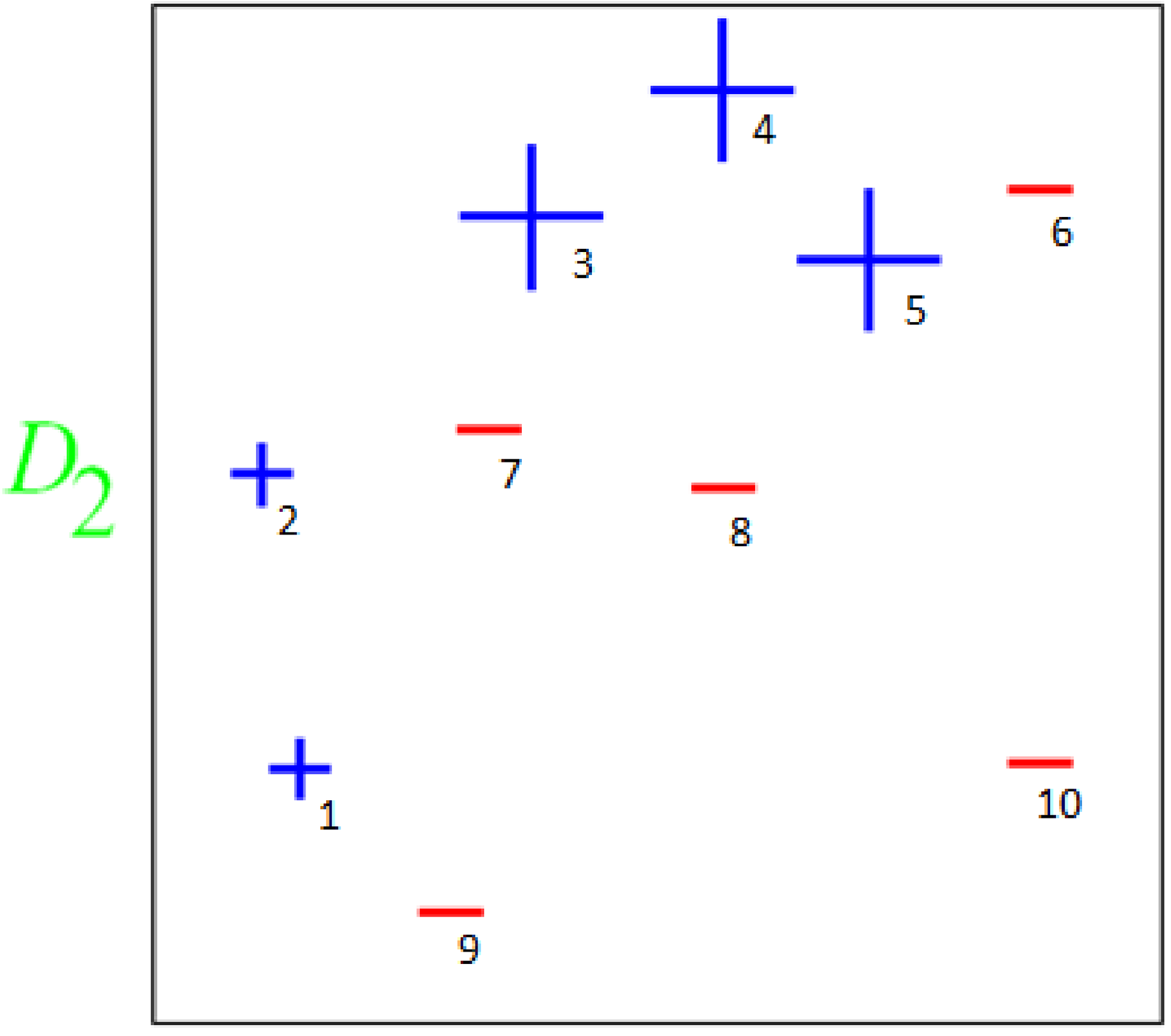

err
yf(x)
3
2.5
2
1.5
1
0.5
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2

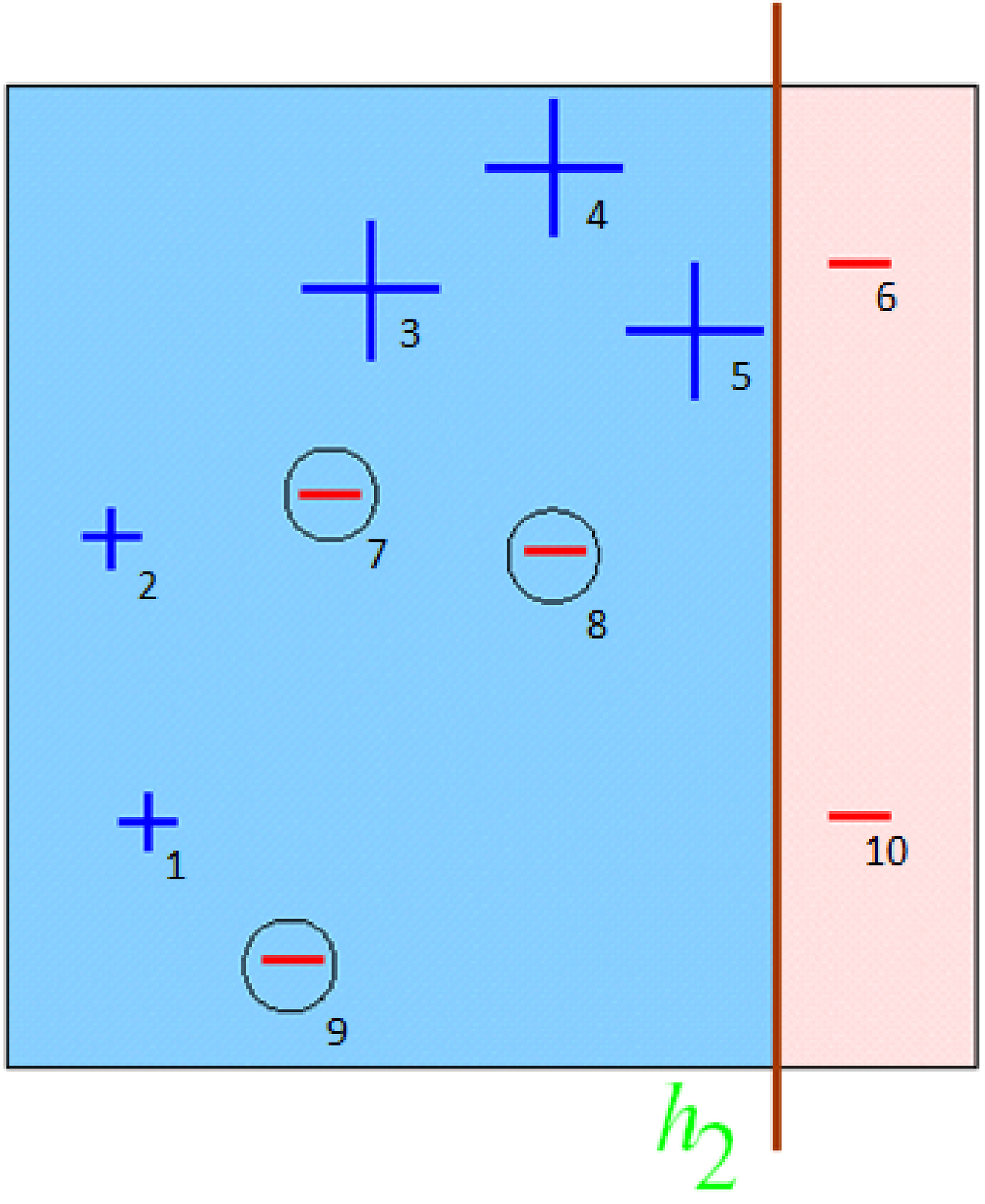

D_1
4
3
5
6
2
7
8
1
10
9



$\varepsilon_1 = 0.30$

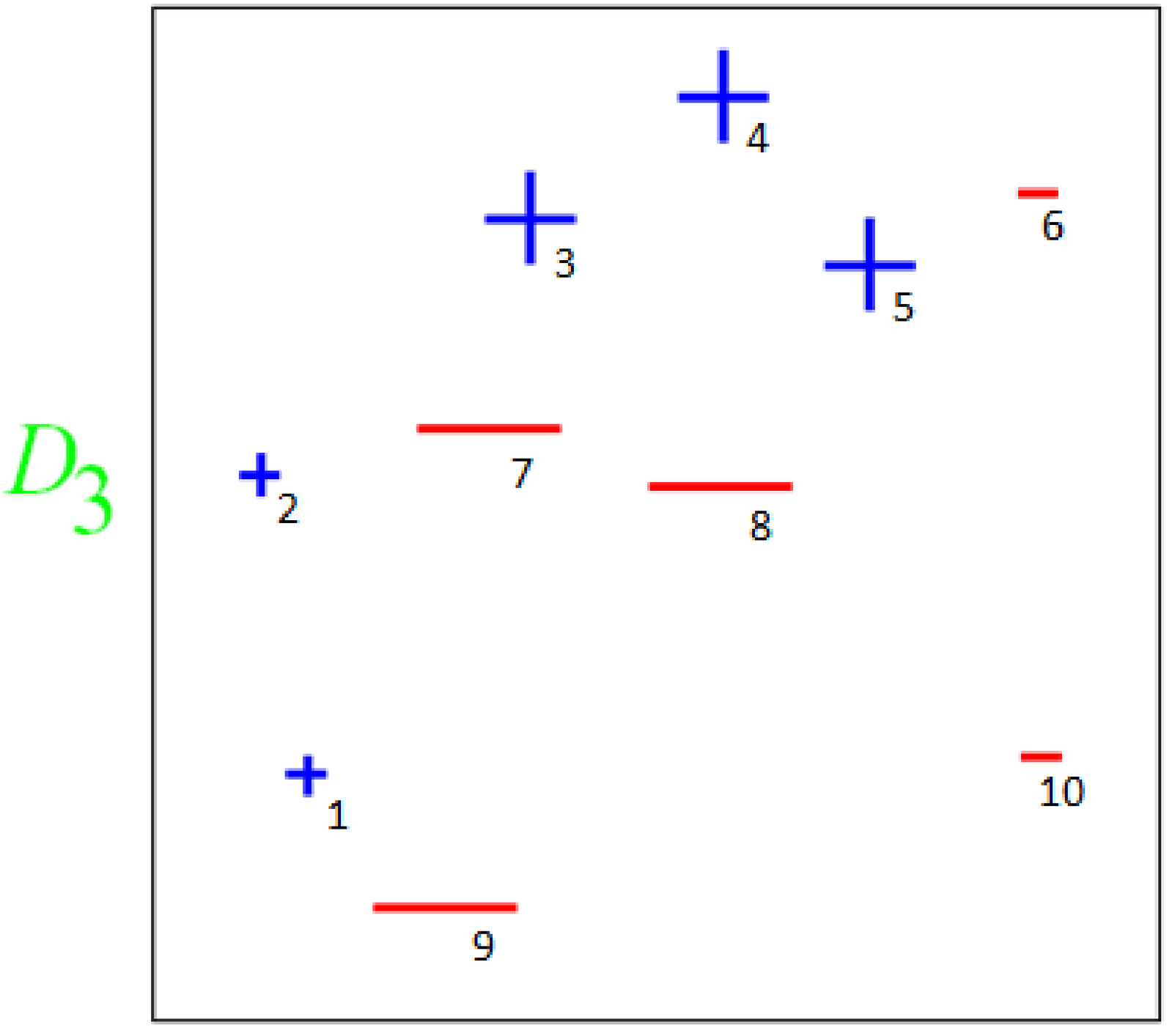
$\alpha_1 = 0.42$

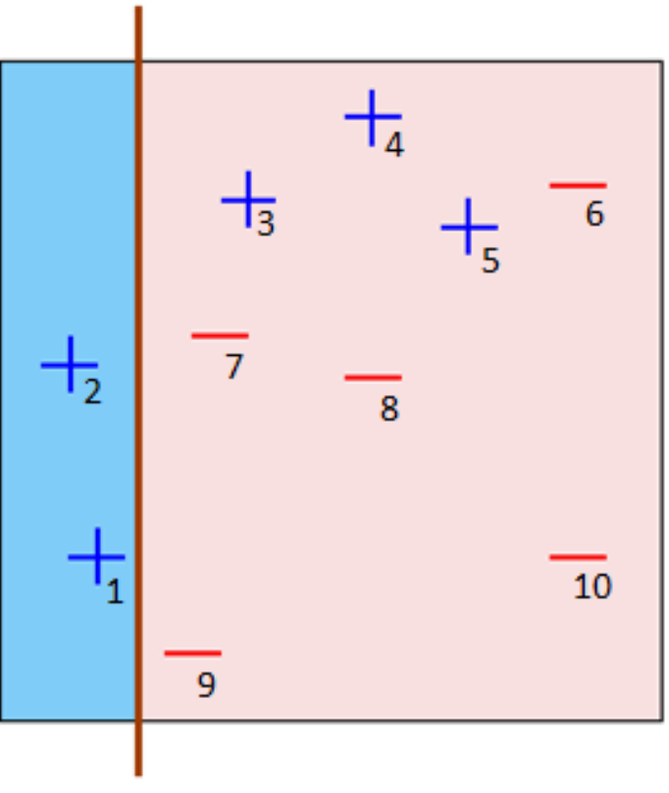

D_2
4
3
5
6
7
2
8
1
10
9

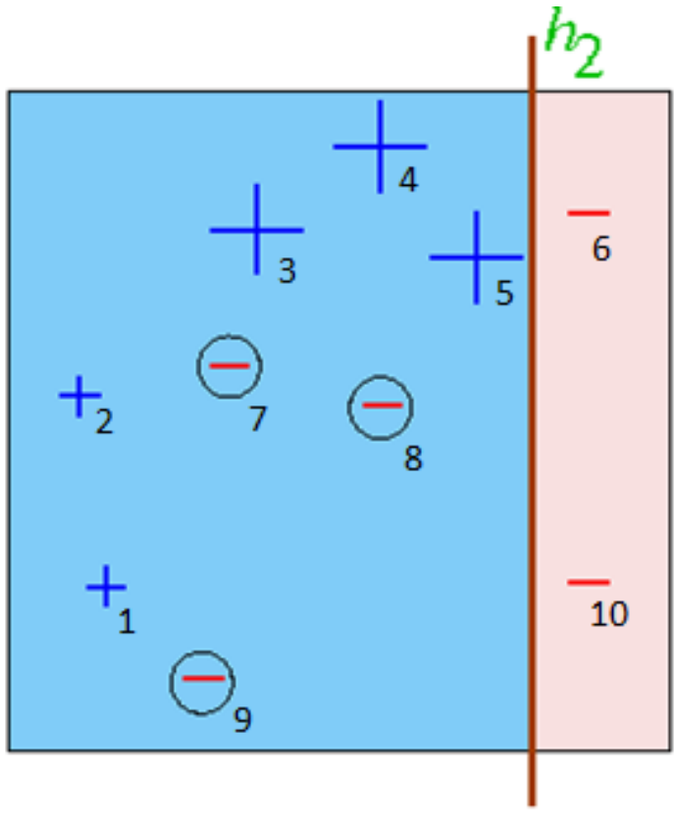


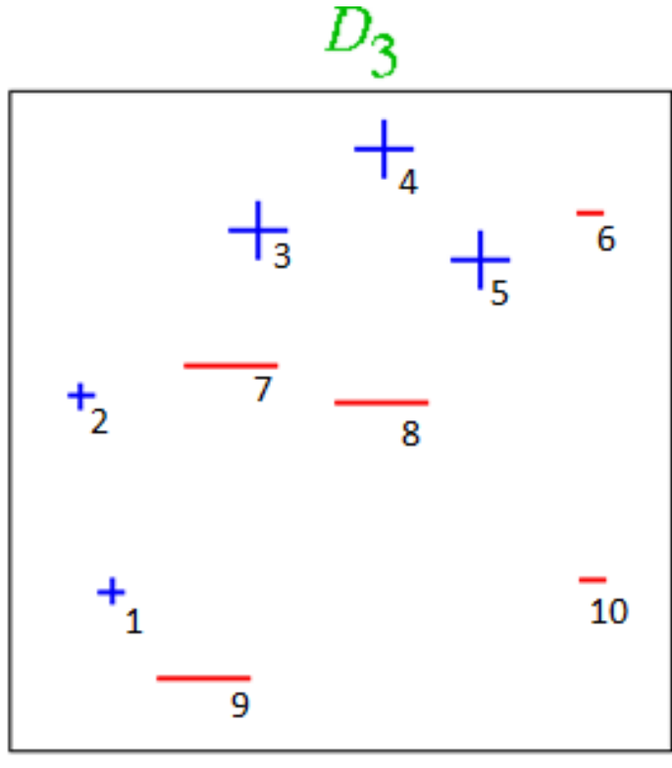
$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$


D_3
4
3
5
6
2
7
8
1
10
9

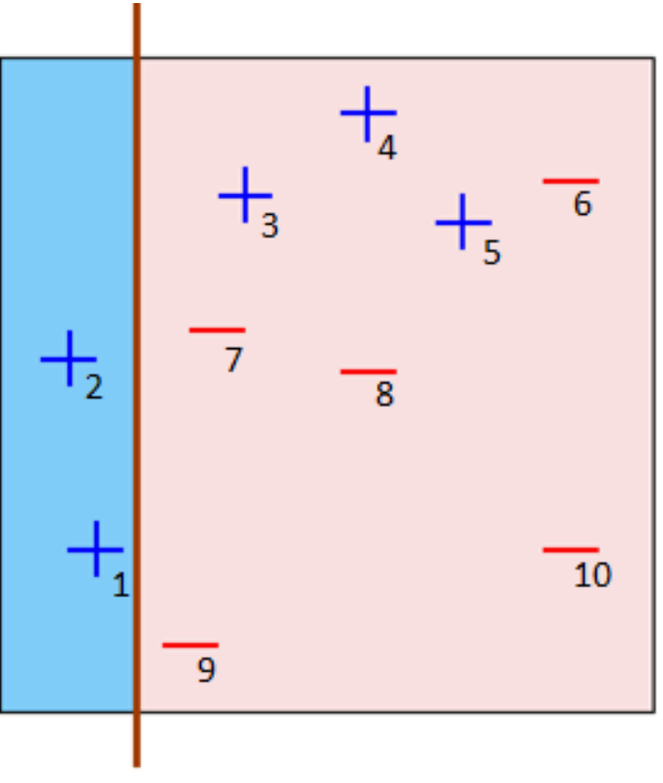


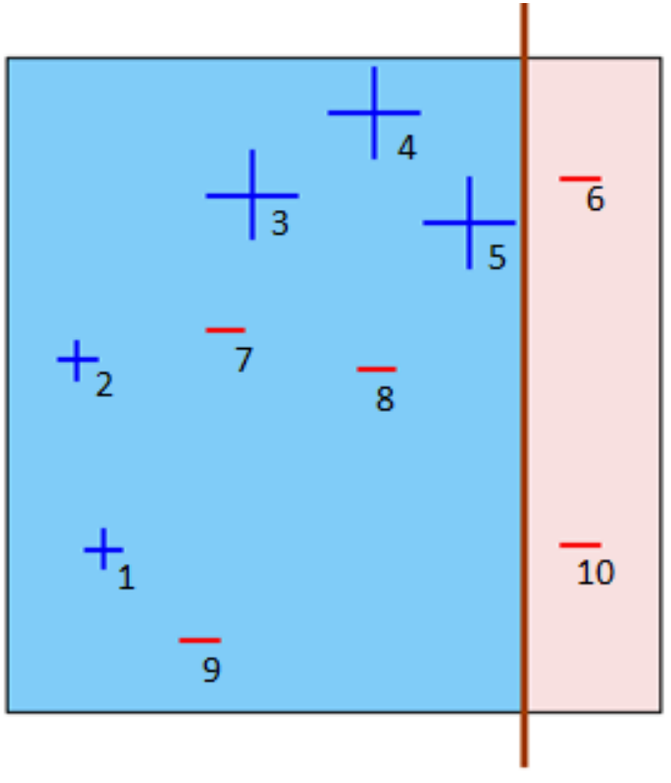


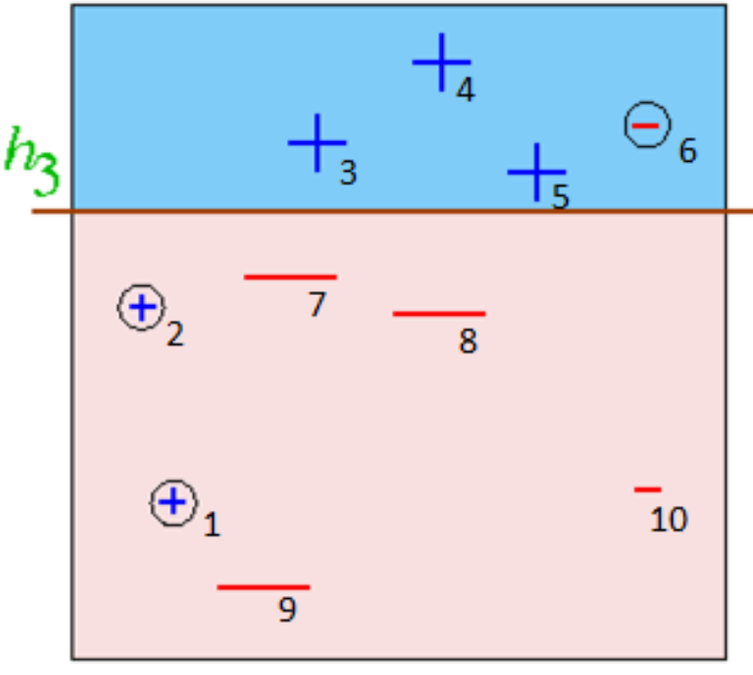


$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$



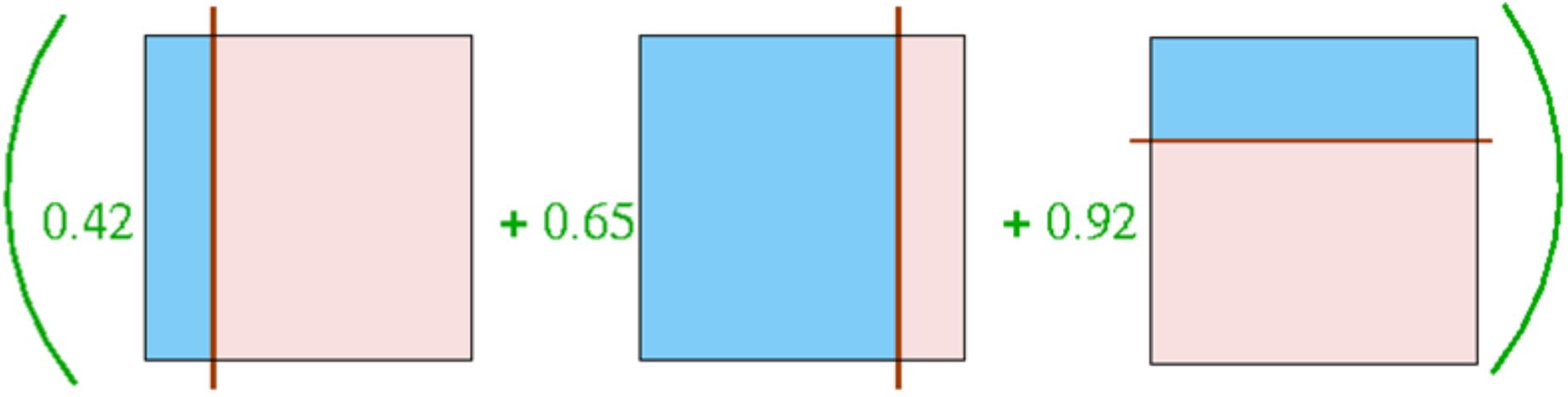




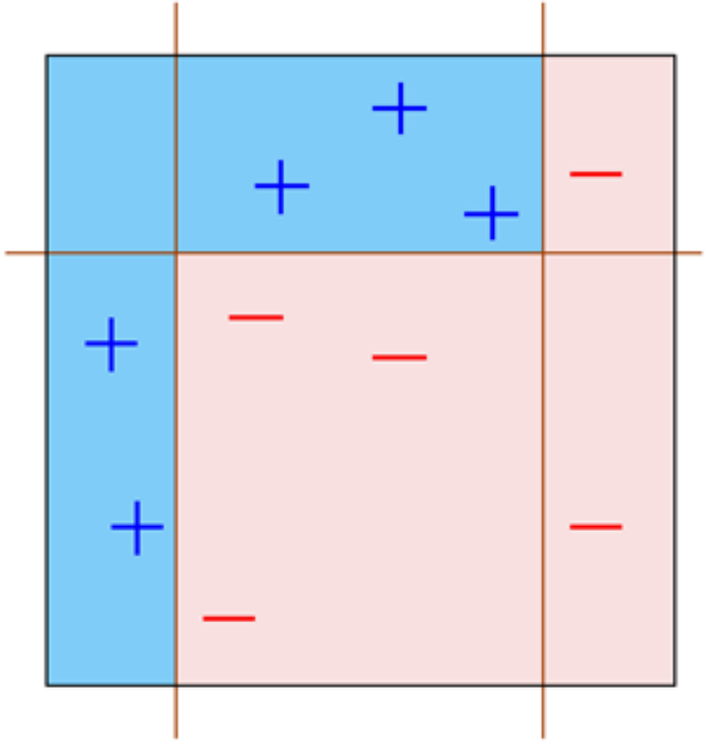
$\varepsilon_3 = 0.14$

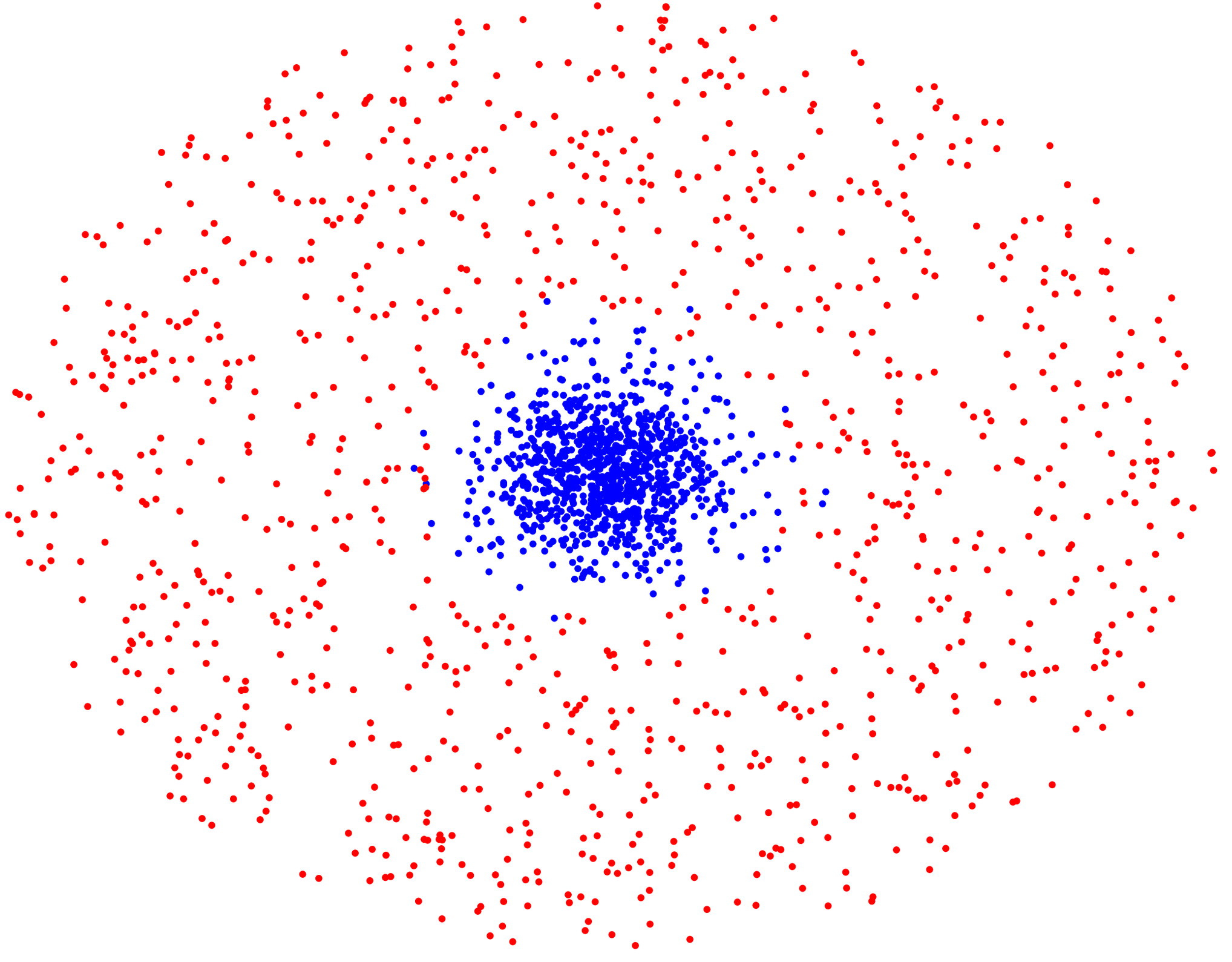
$\alpha_3 = 0.92$

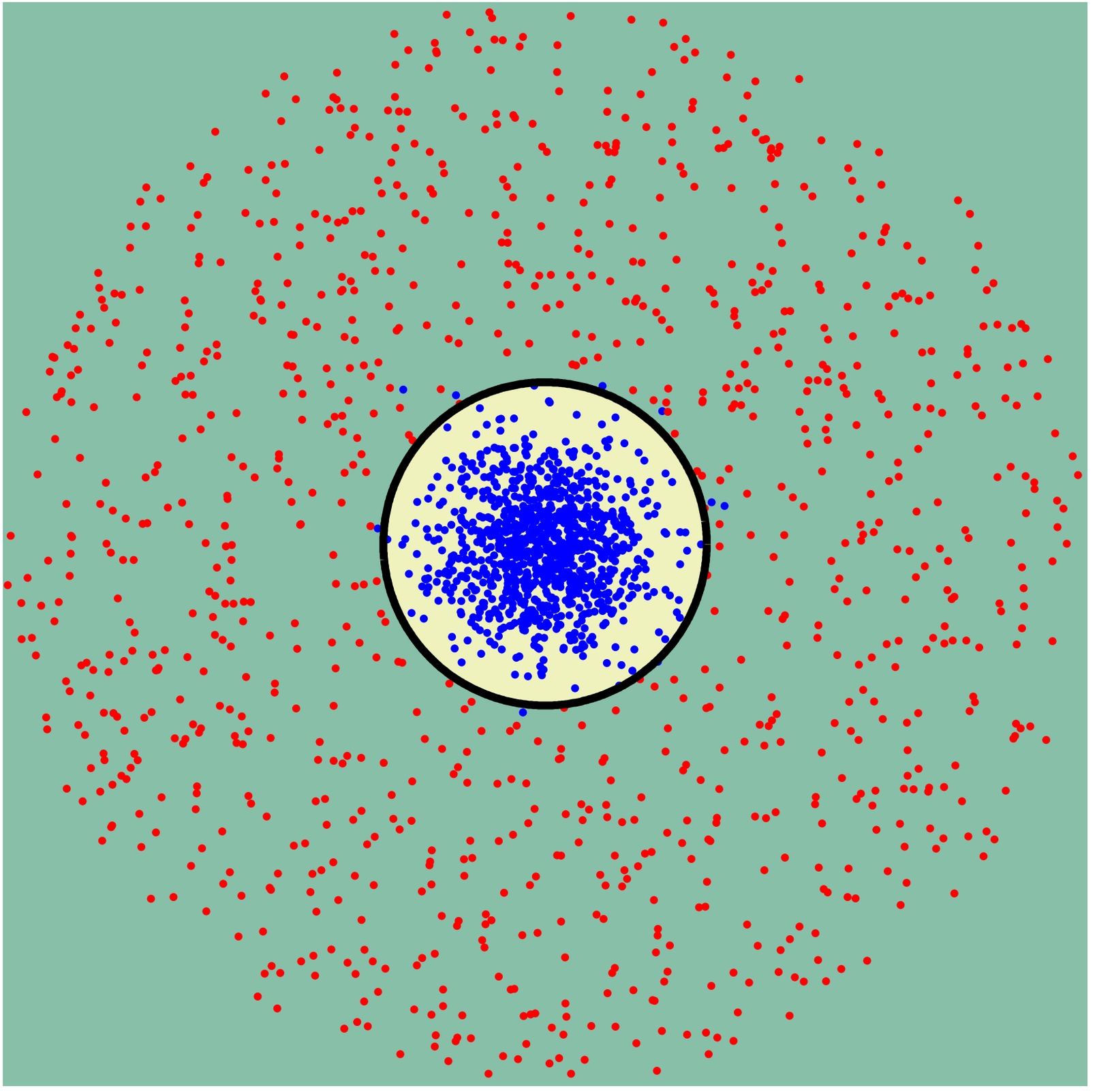
$H_{\text{final}} = \text{sign}\left(0.42 \; + 0.65 \; + 0.92 \right)$

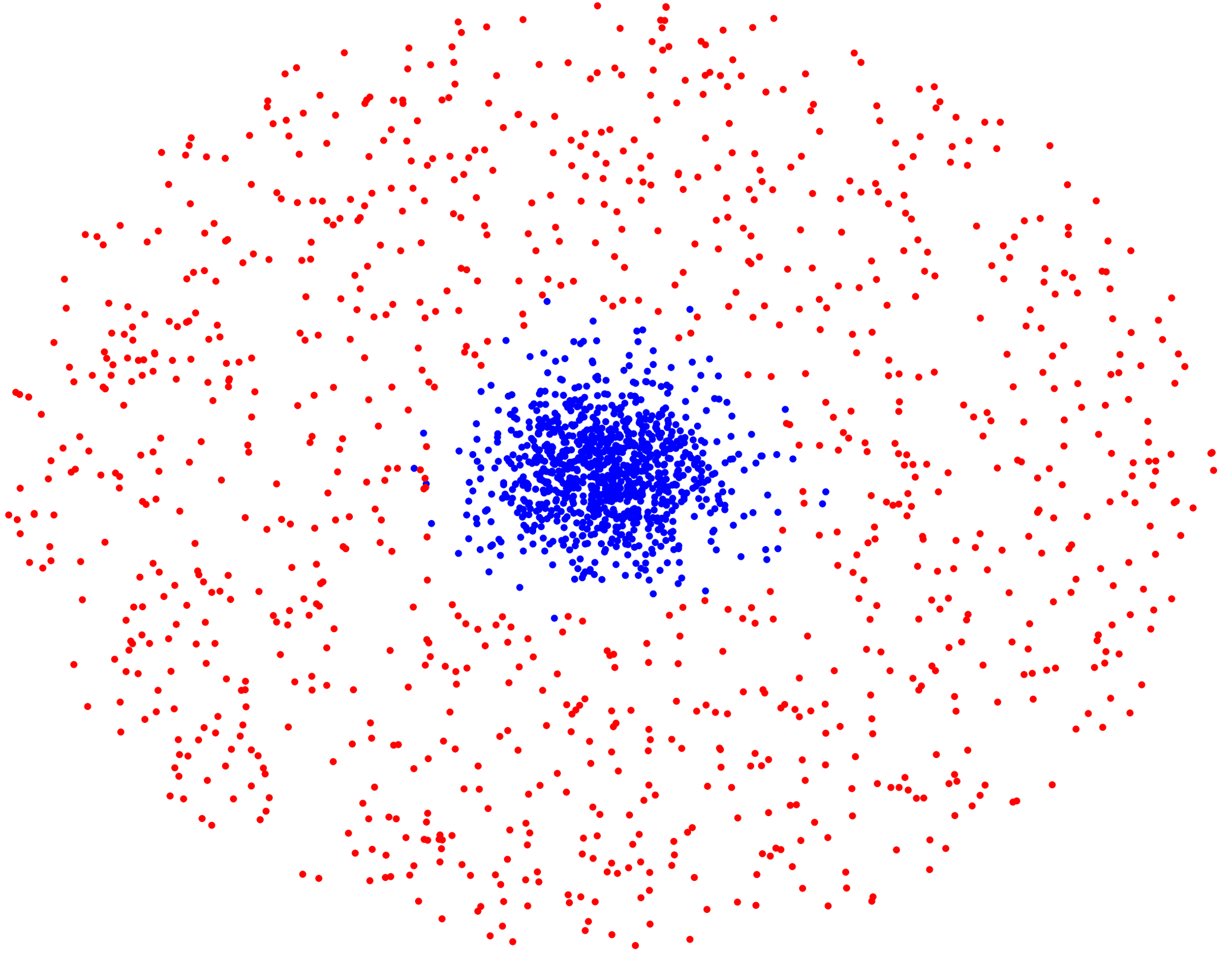


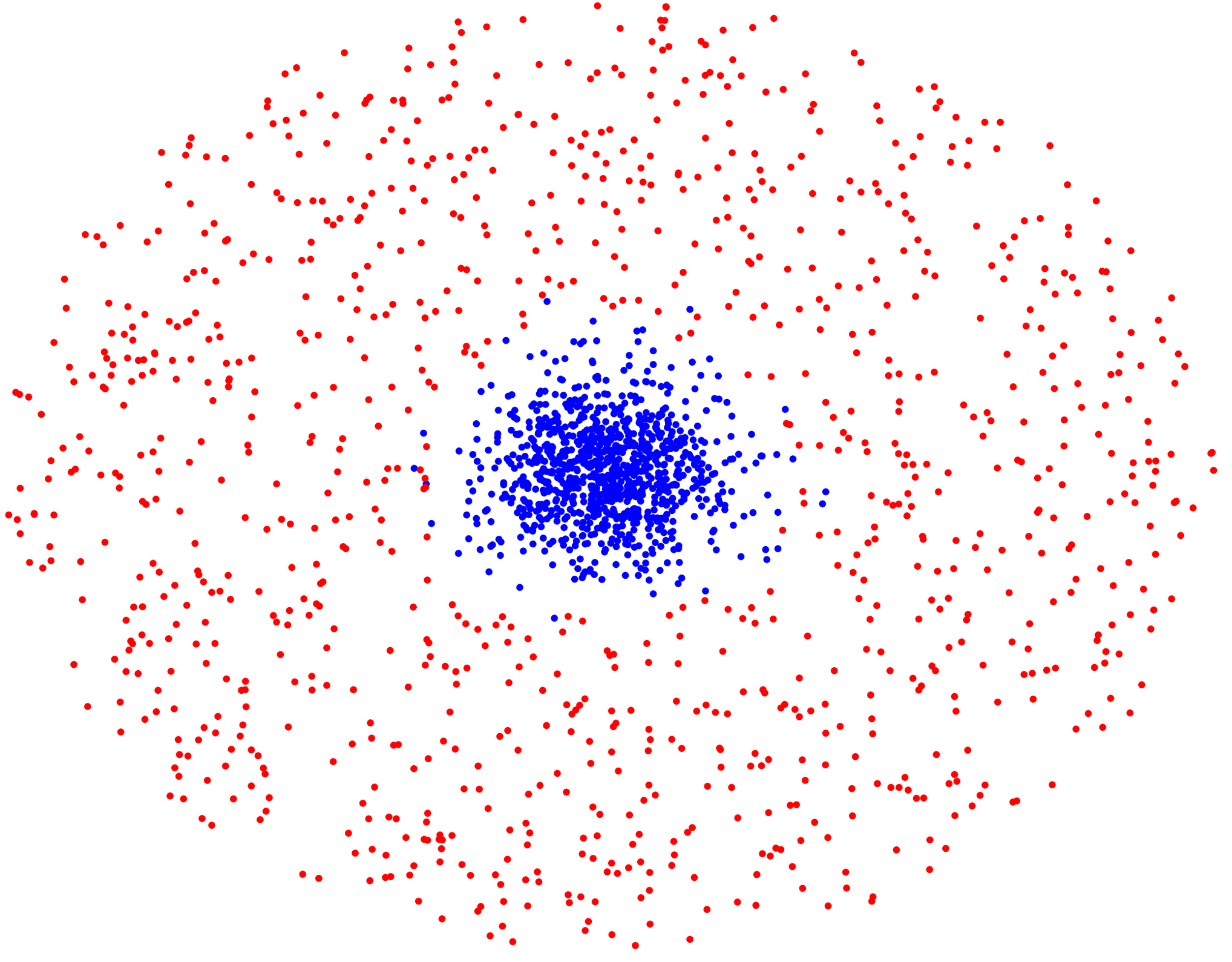
$=$

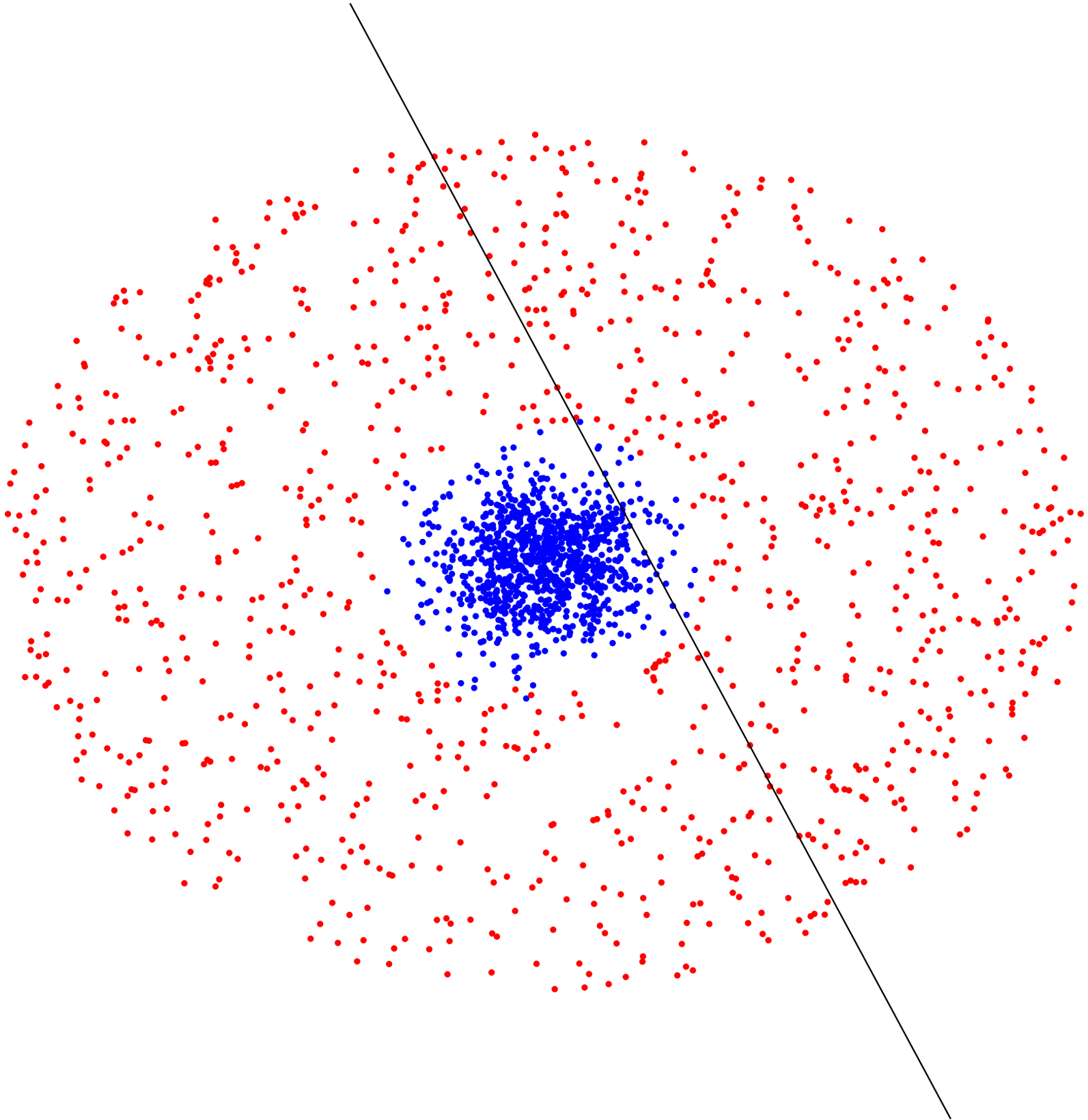


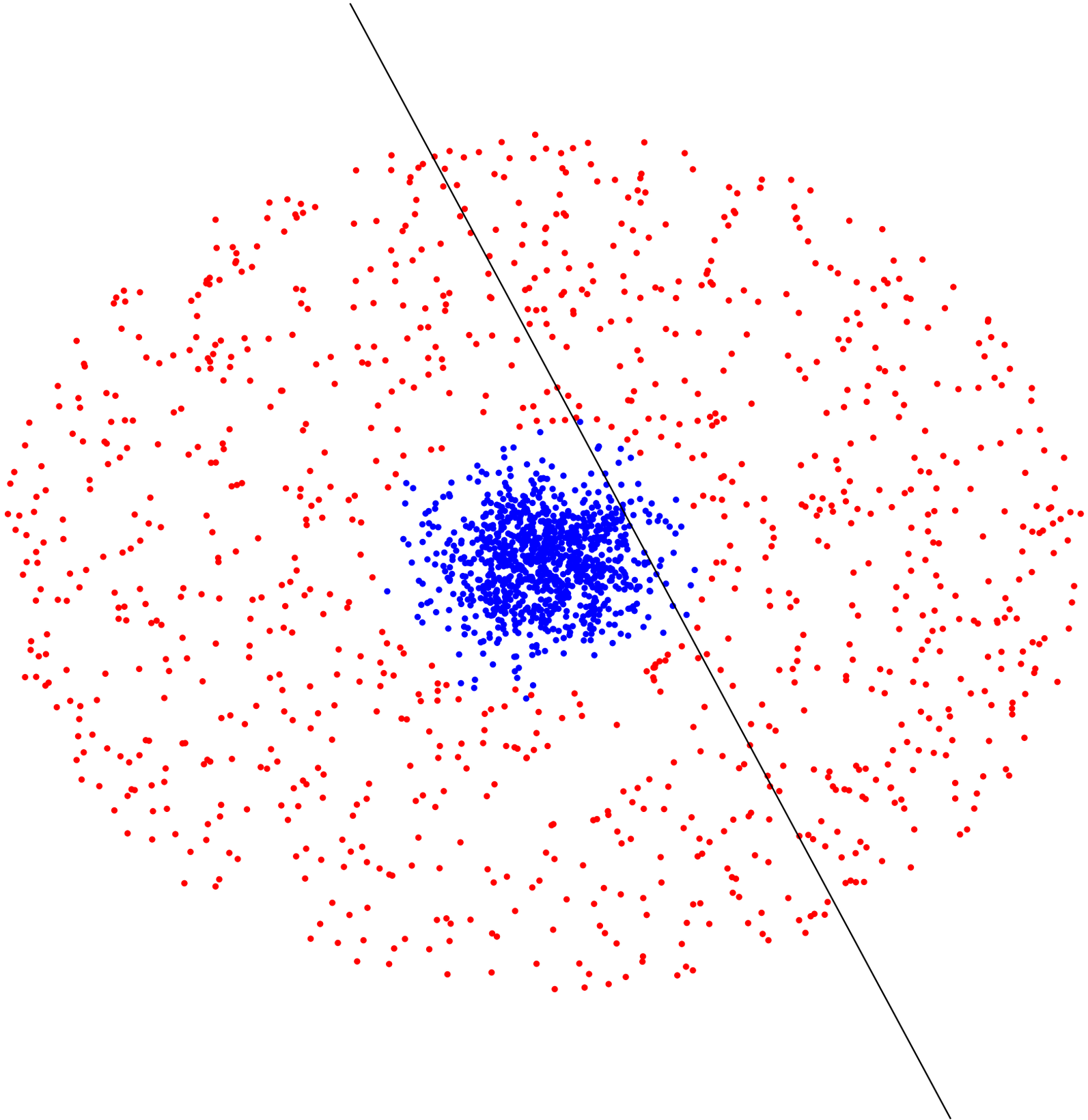


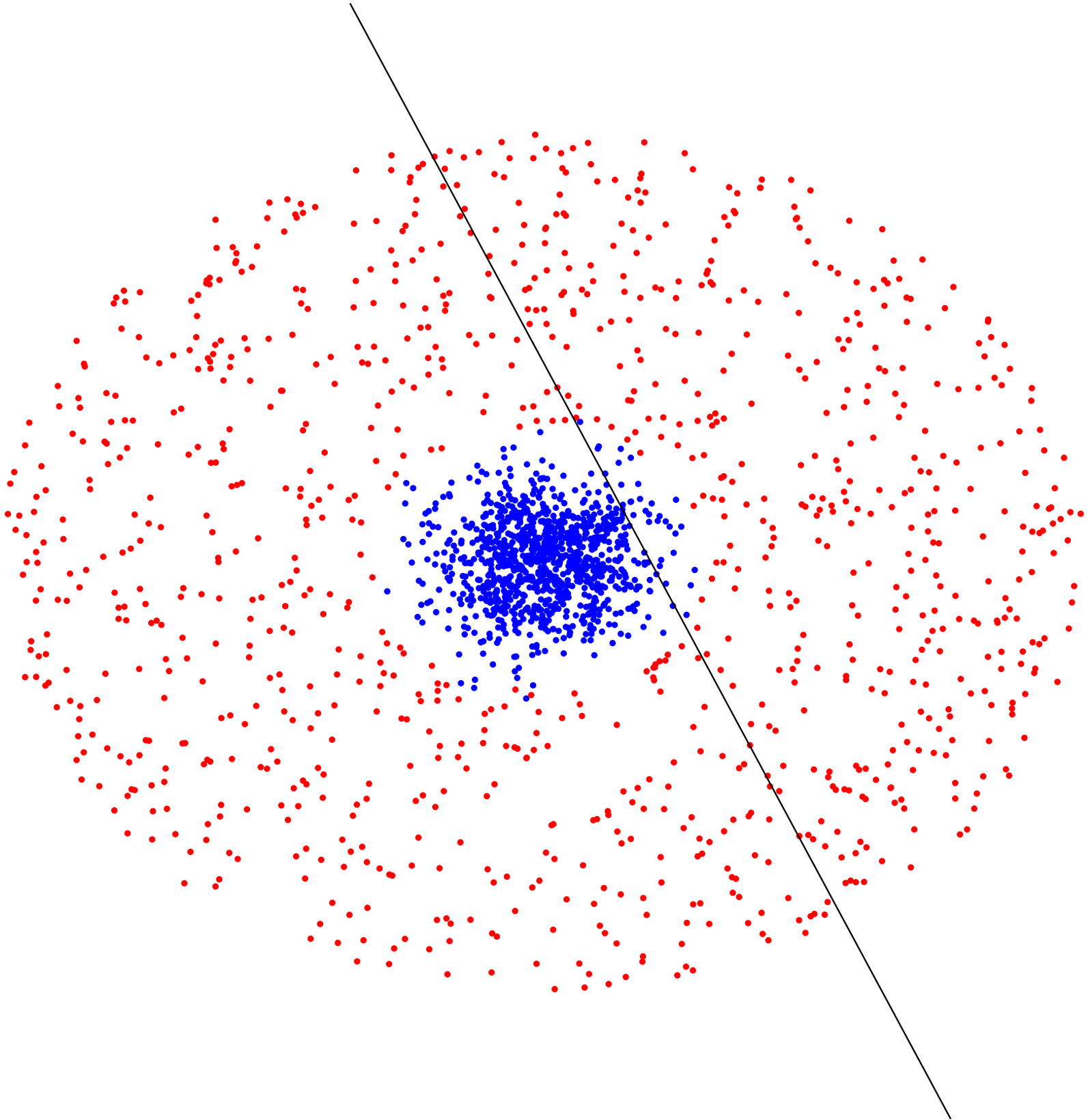


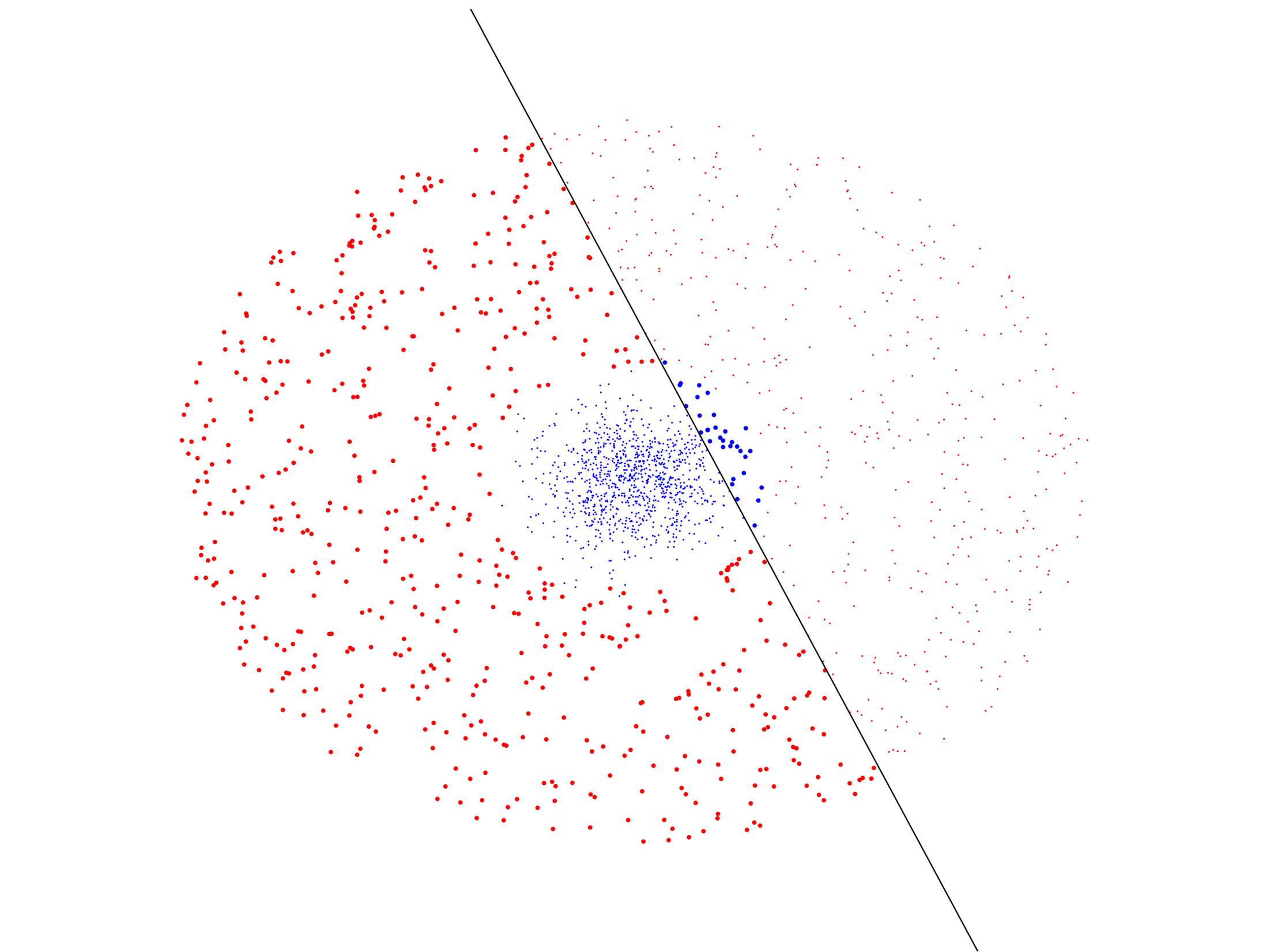


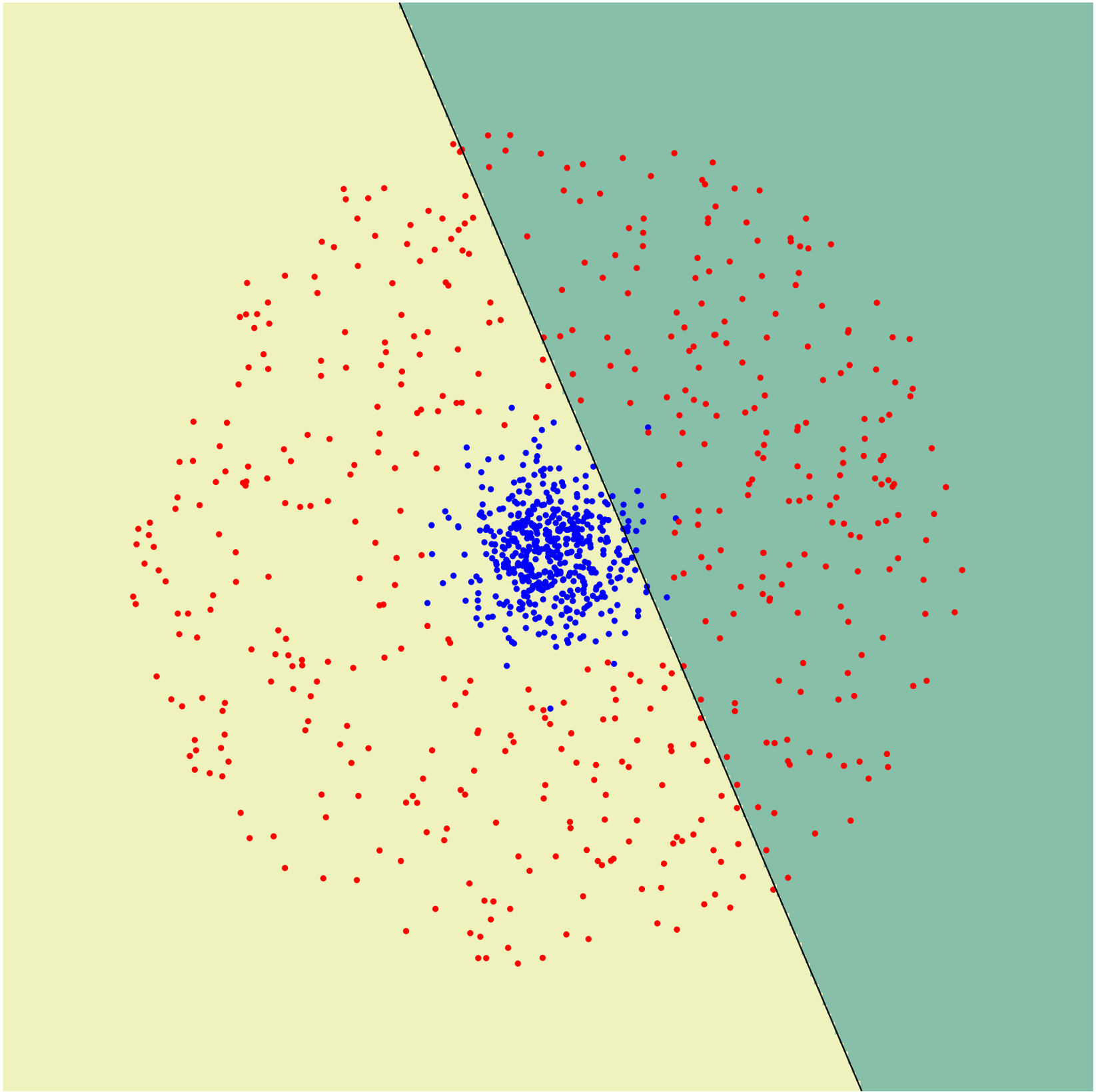


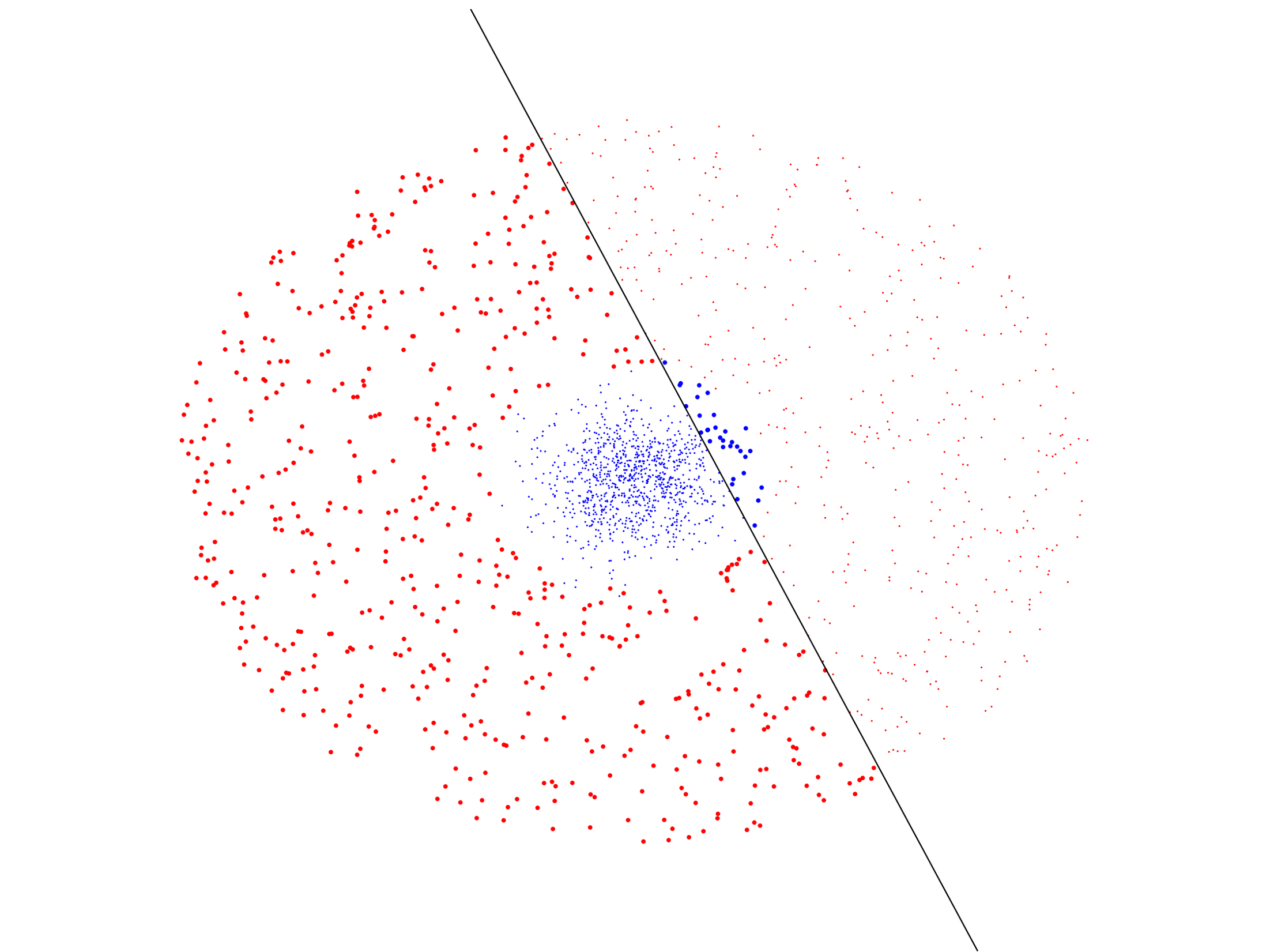


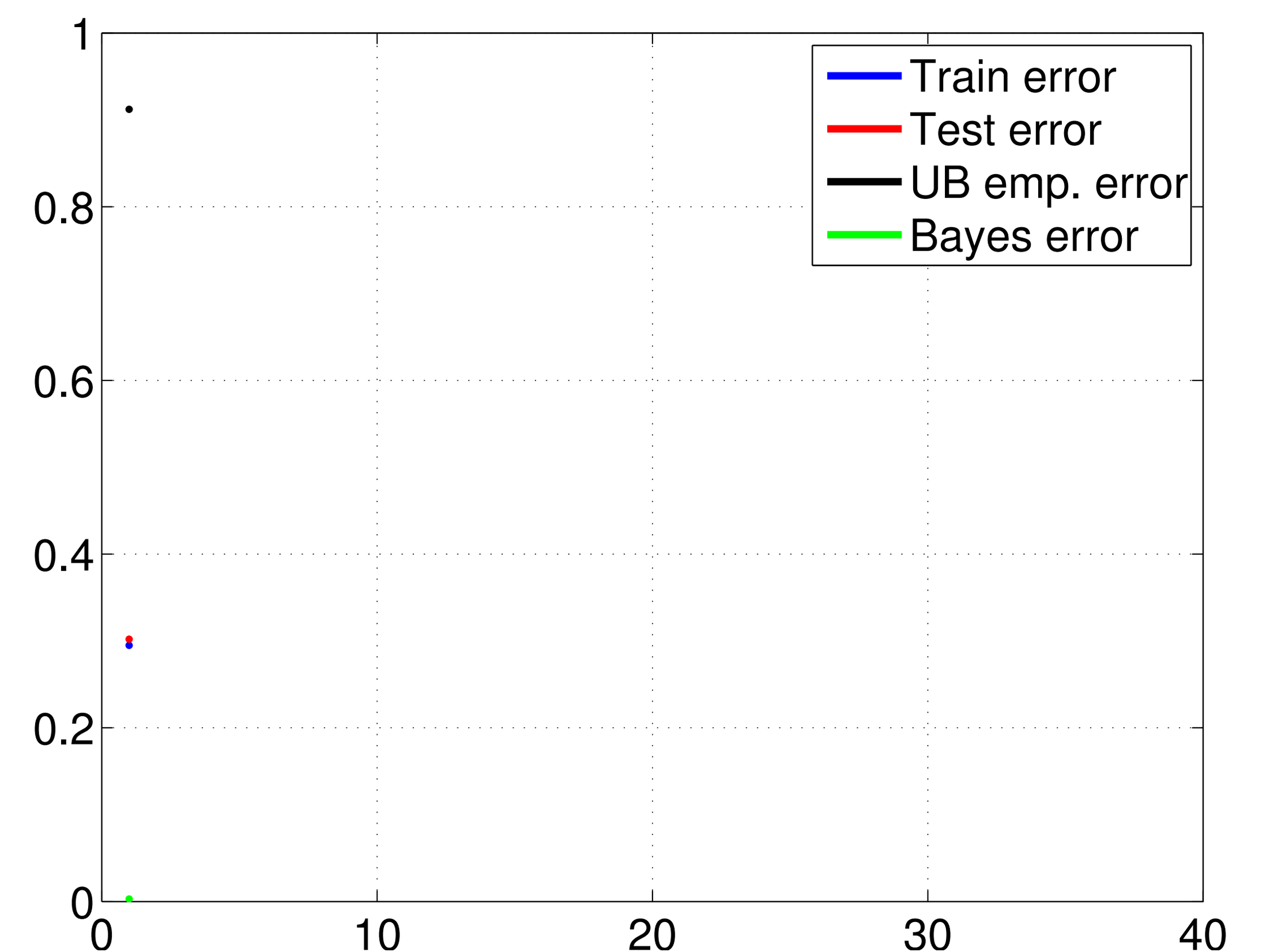


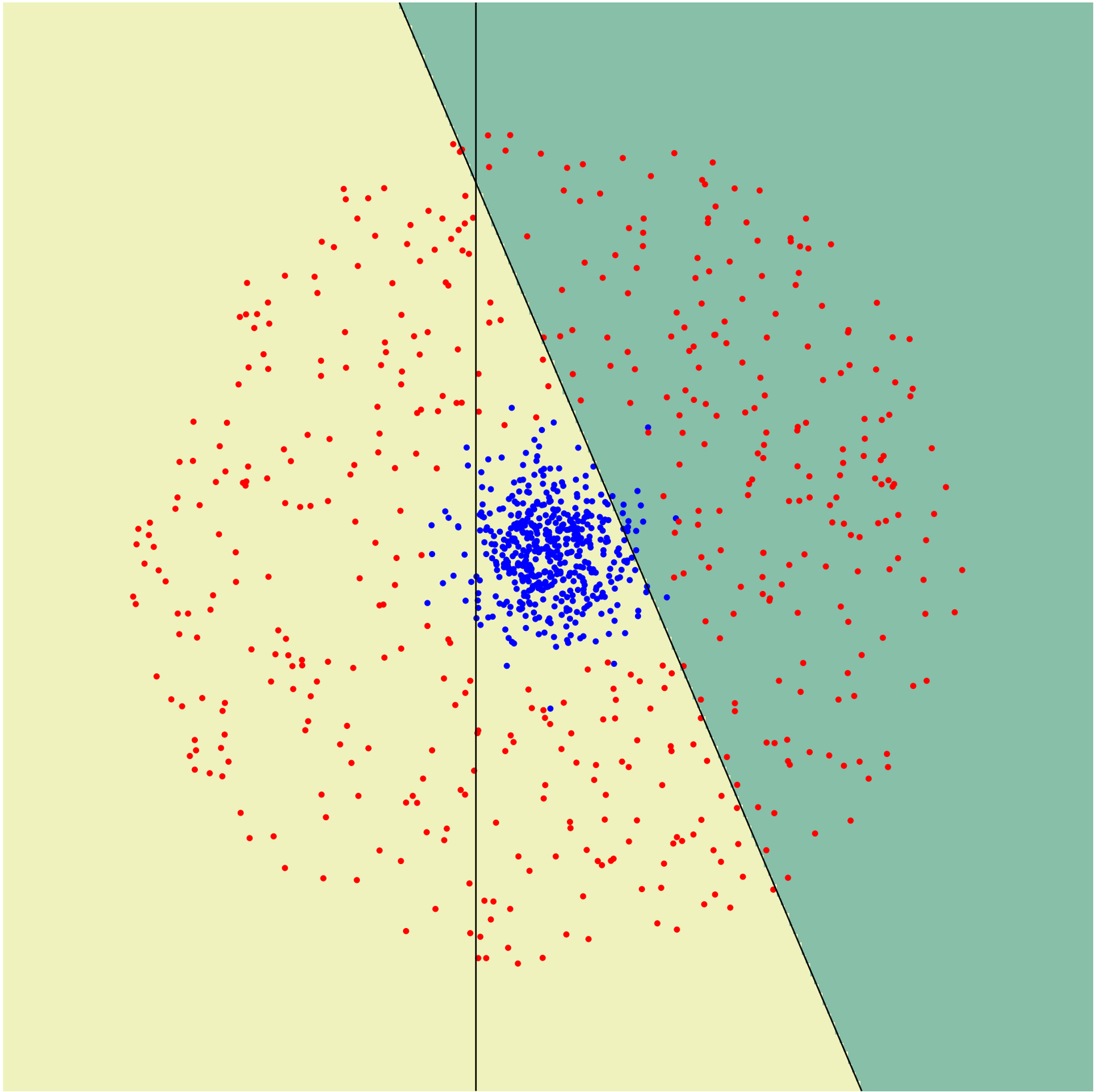


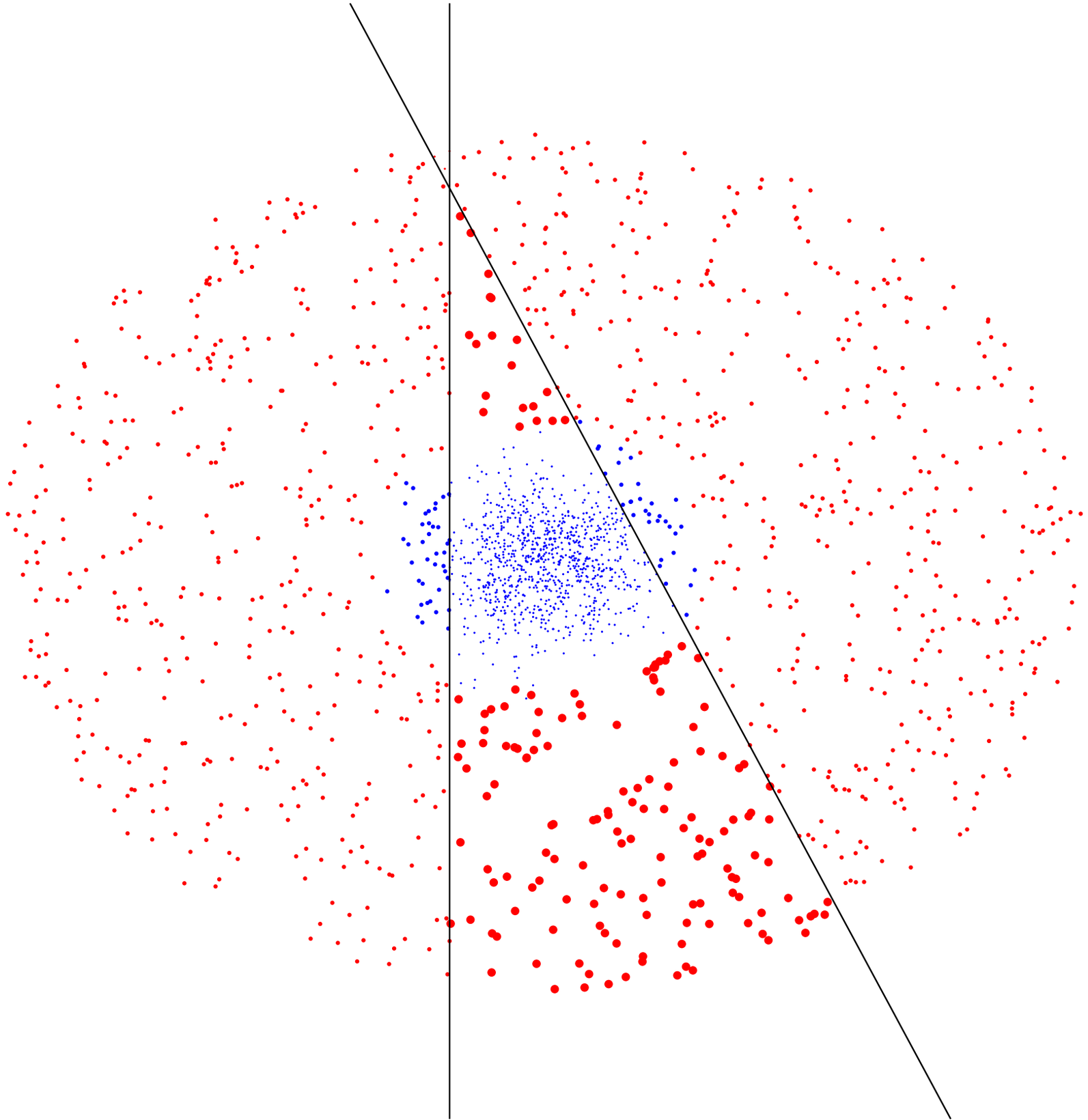


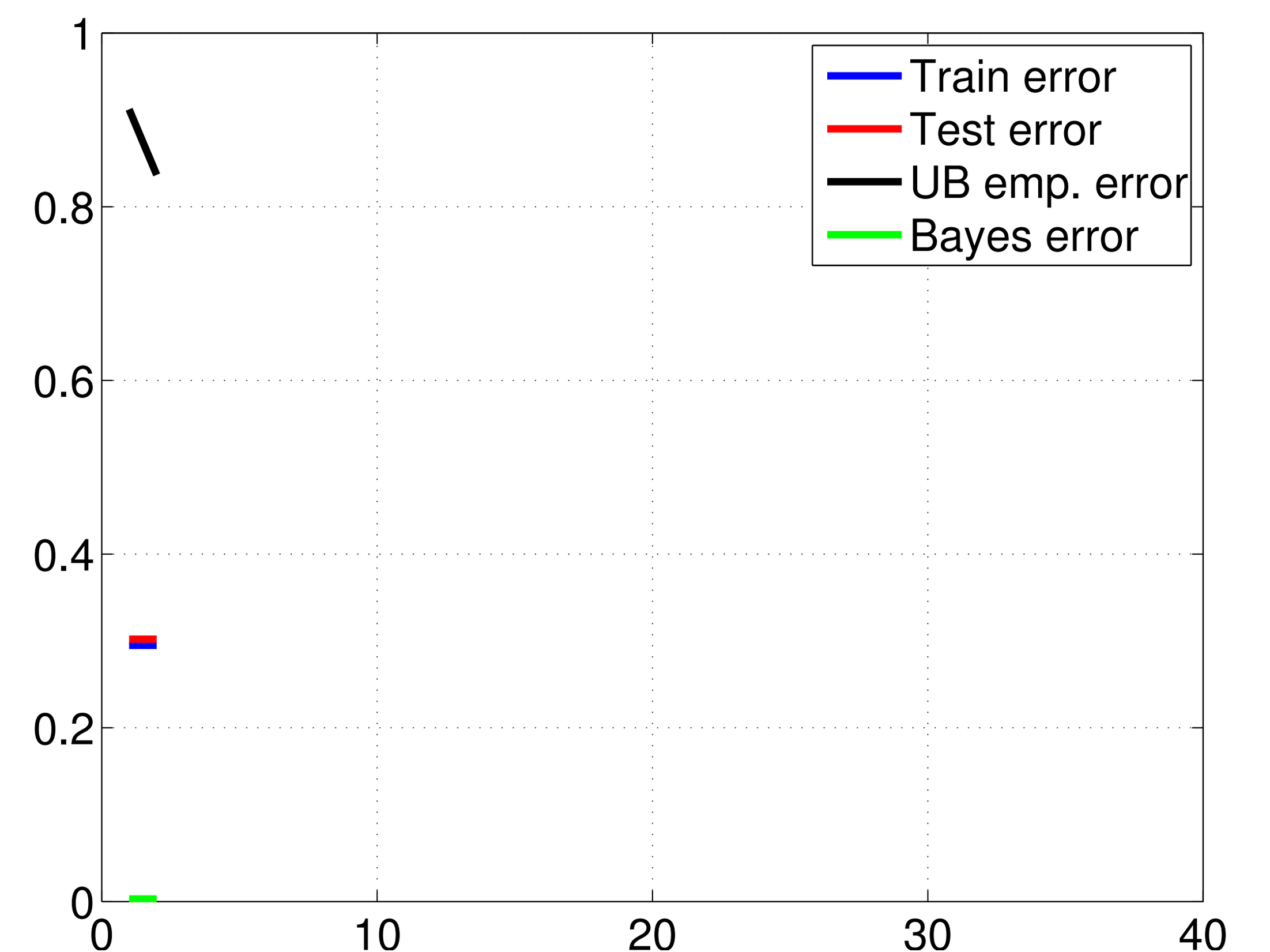


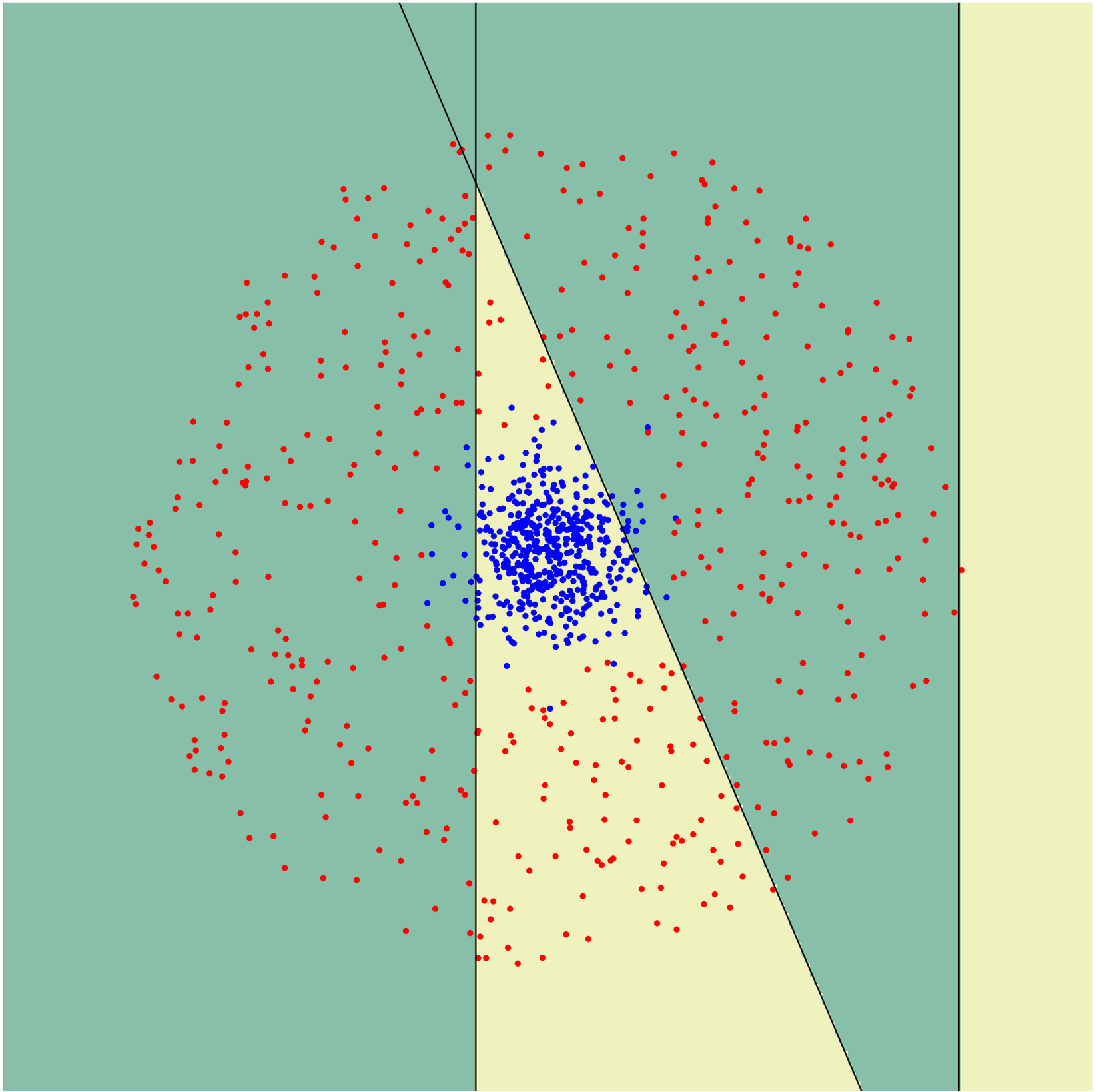


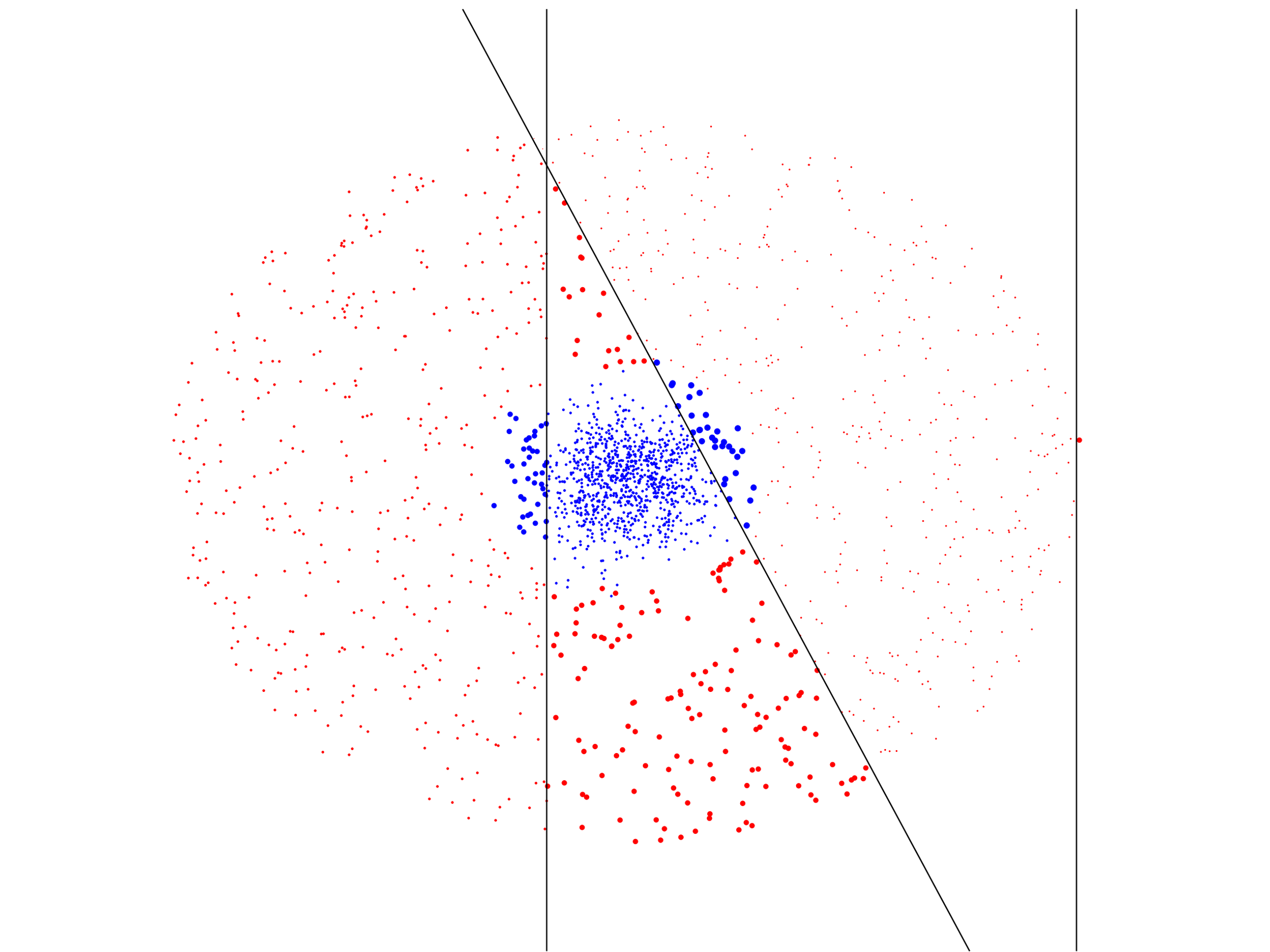

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

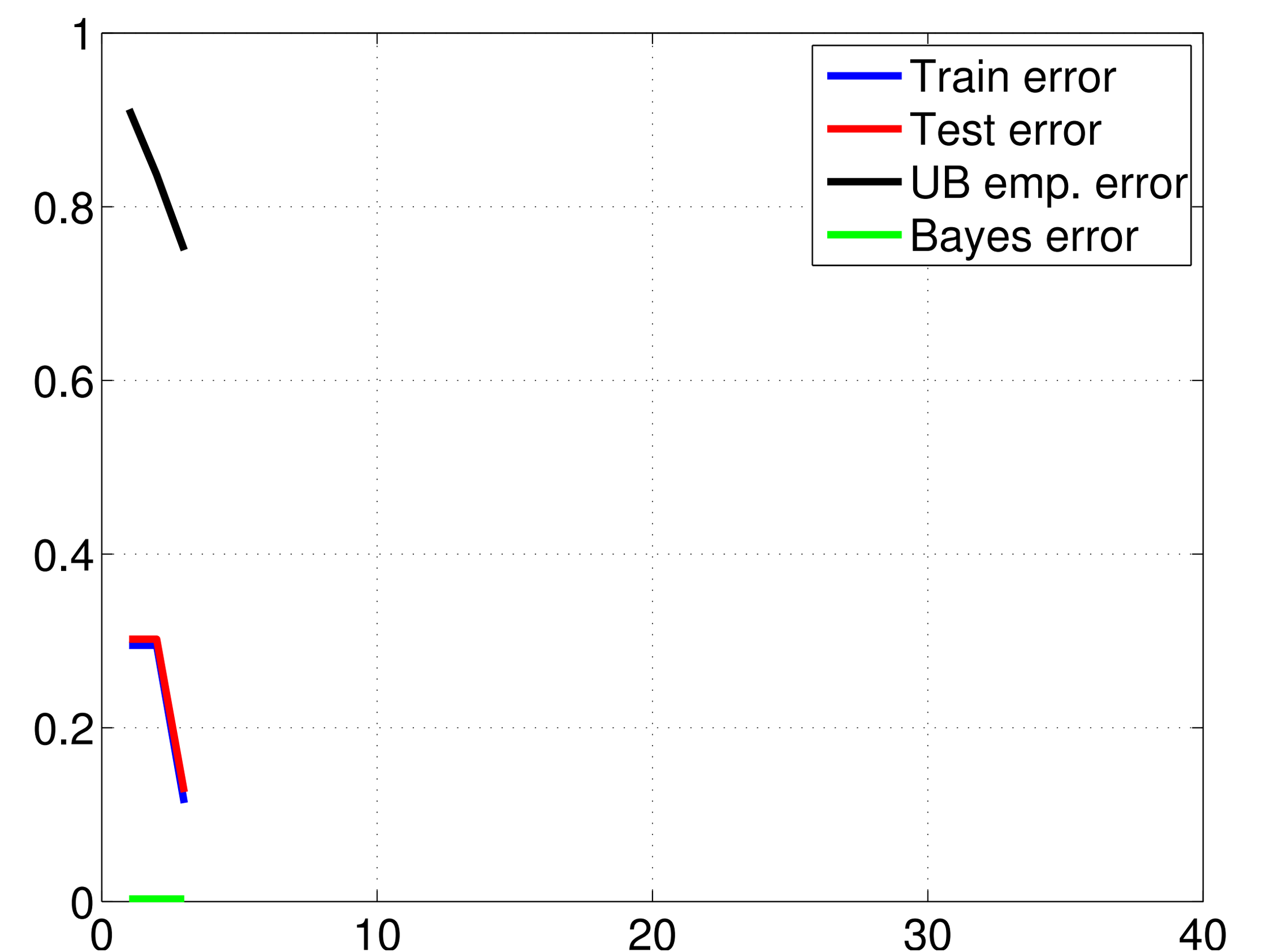


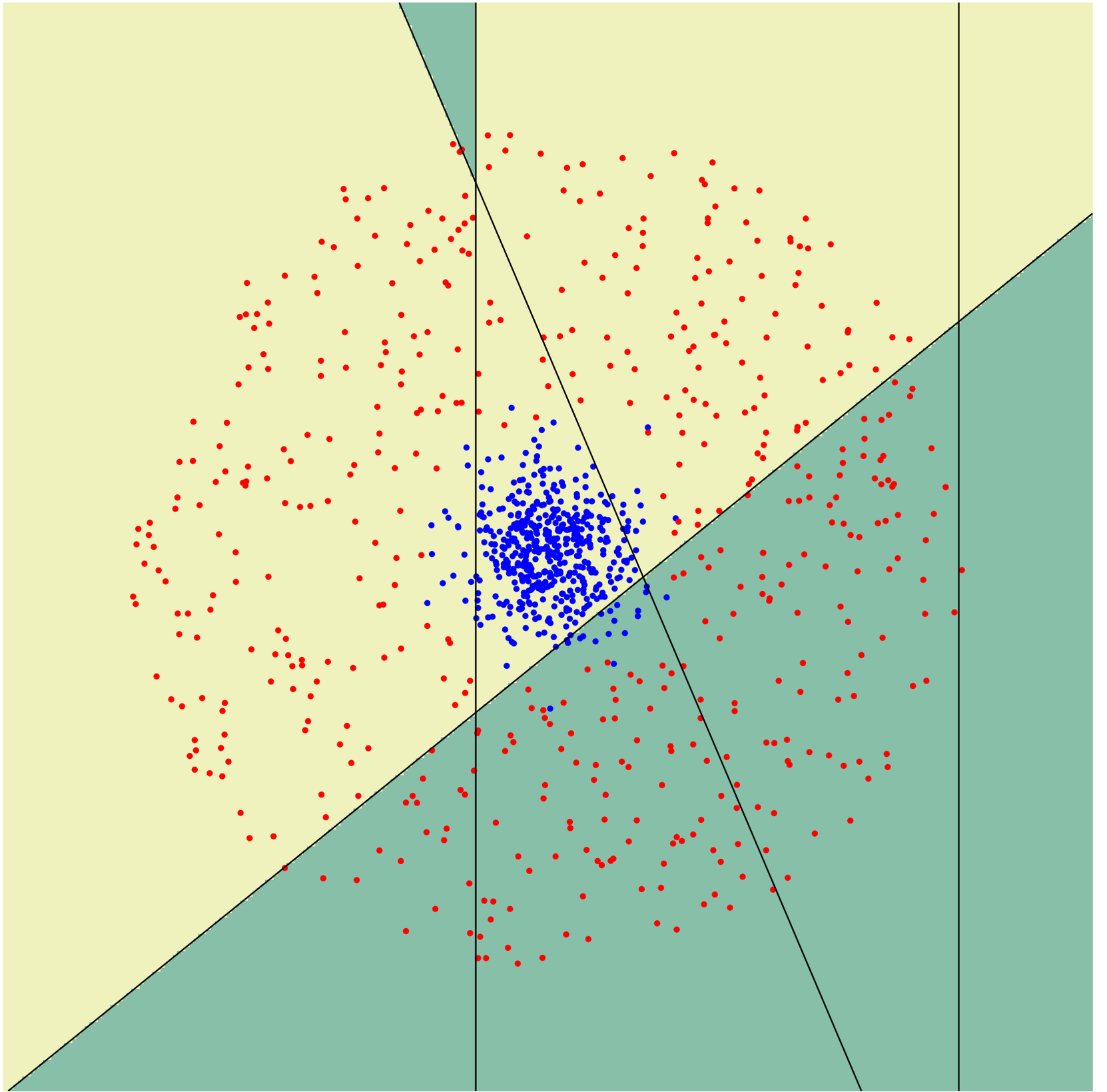


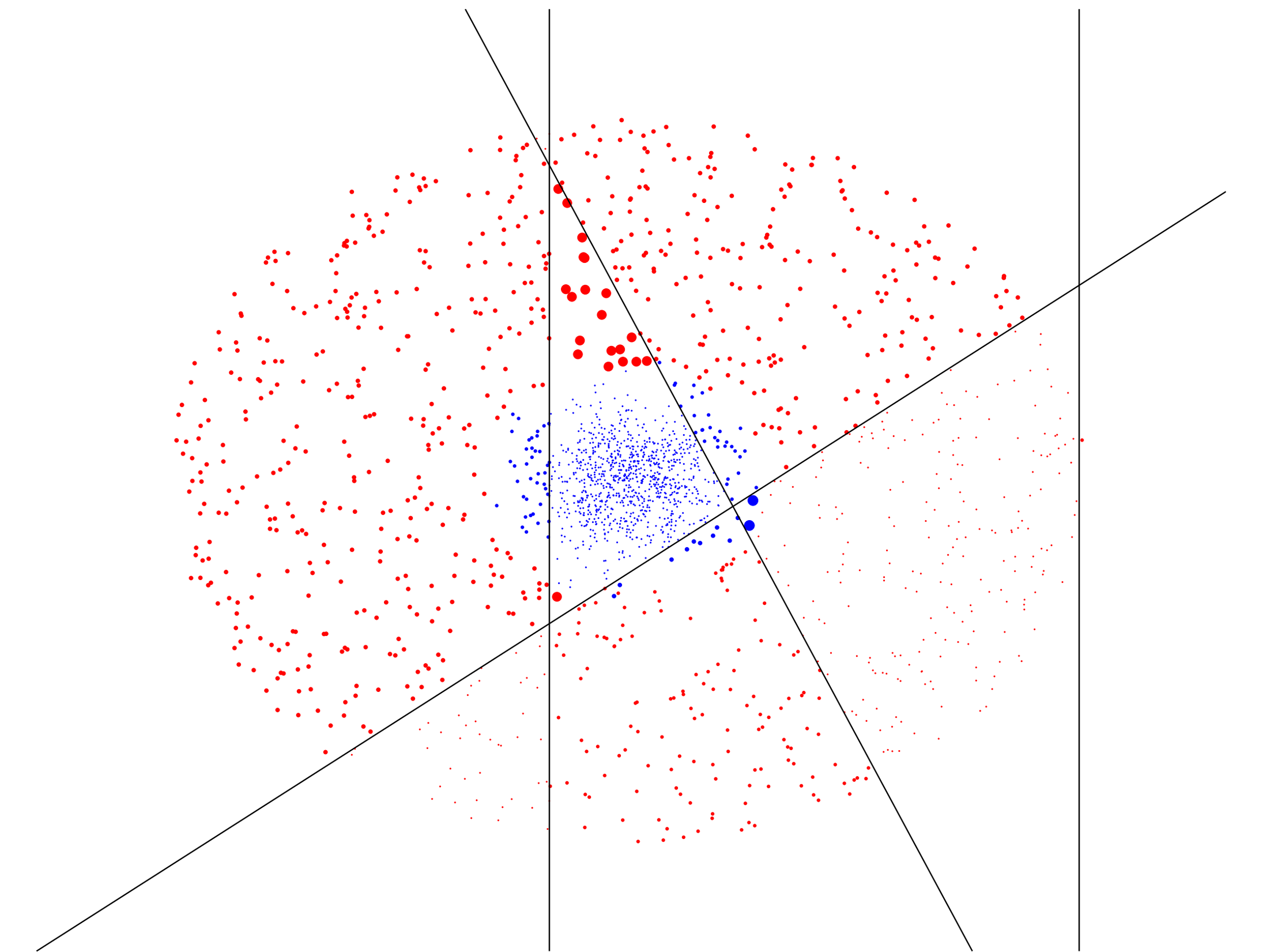

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

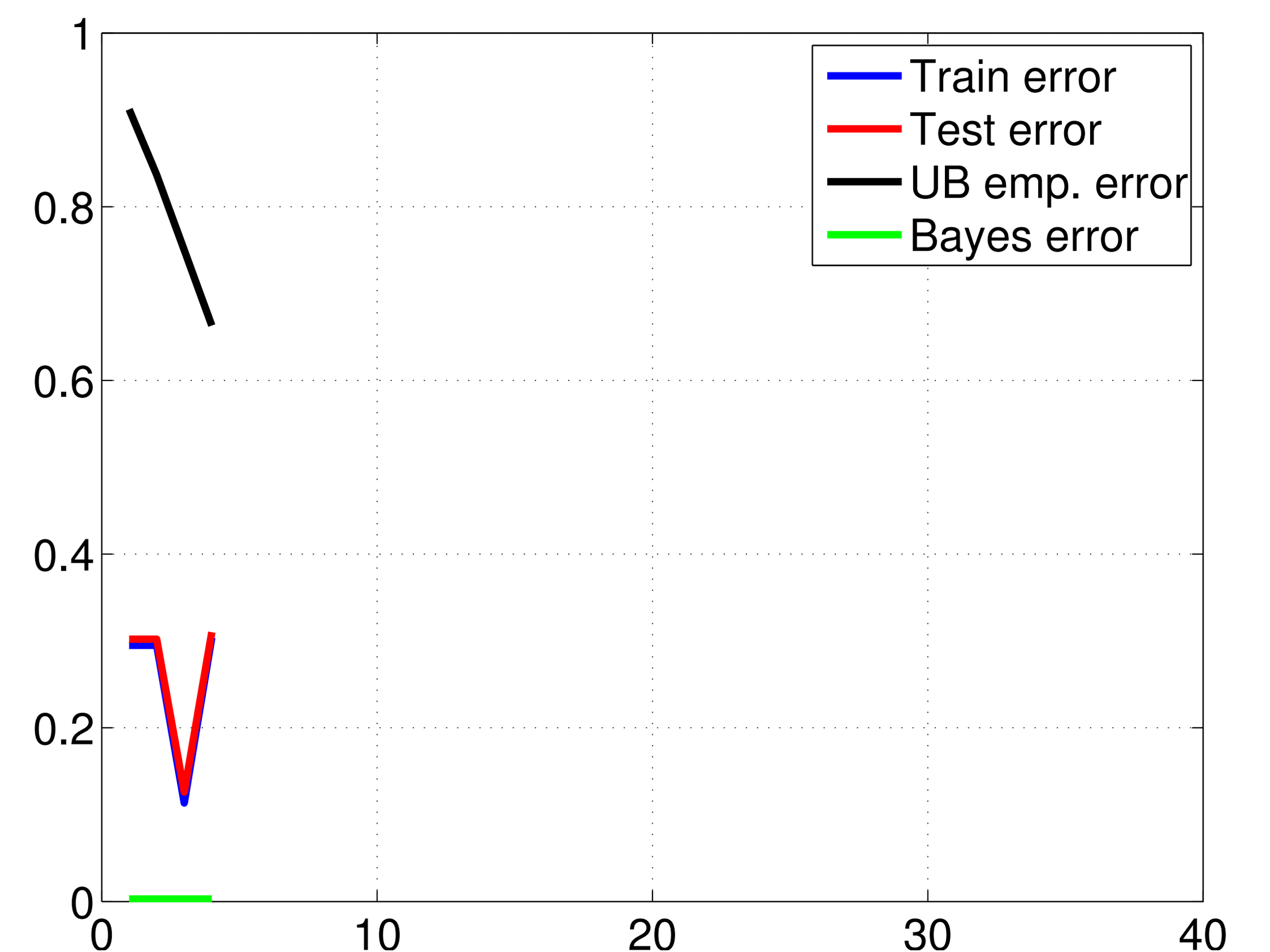


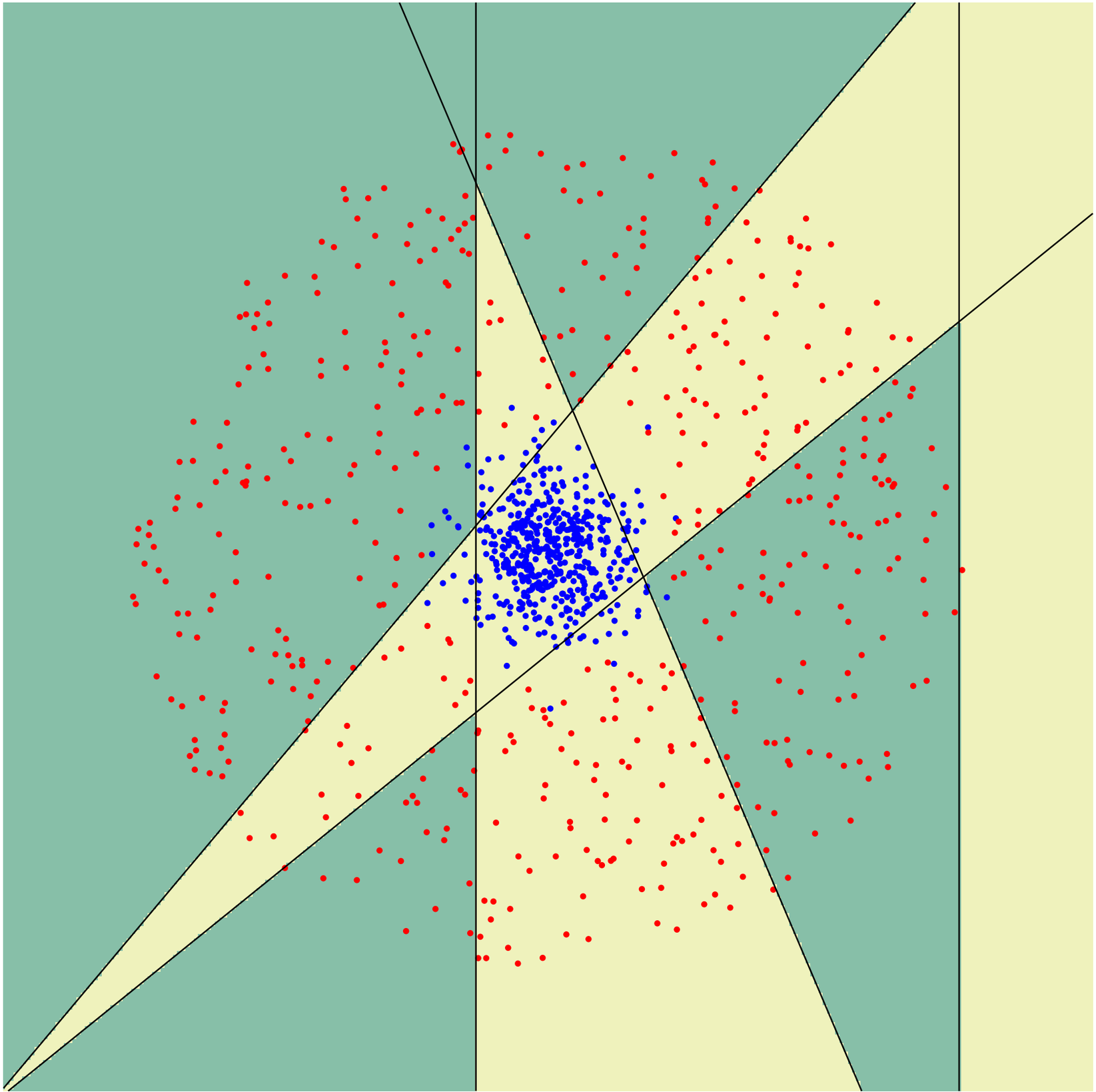


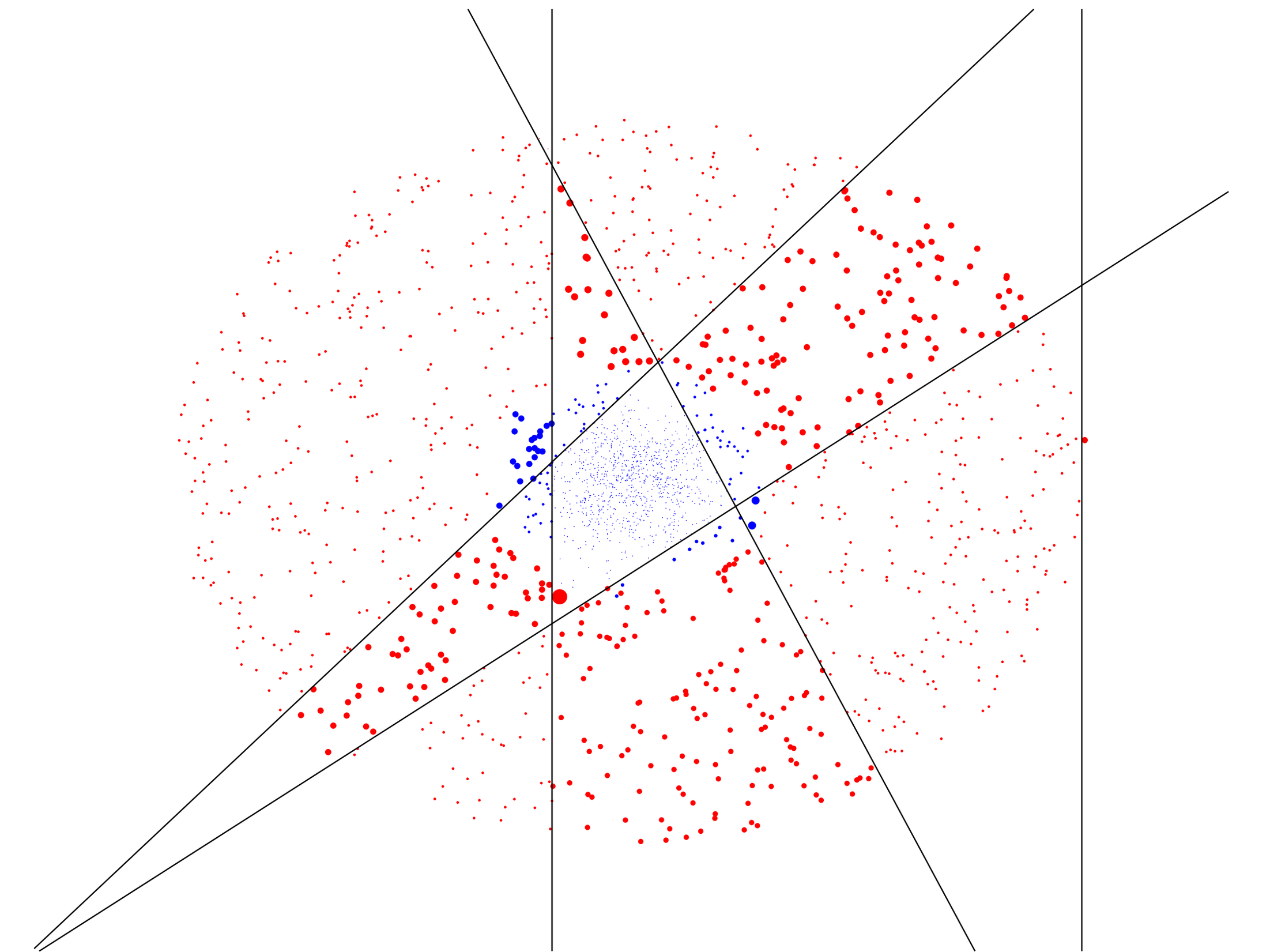

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

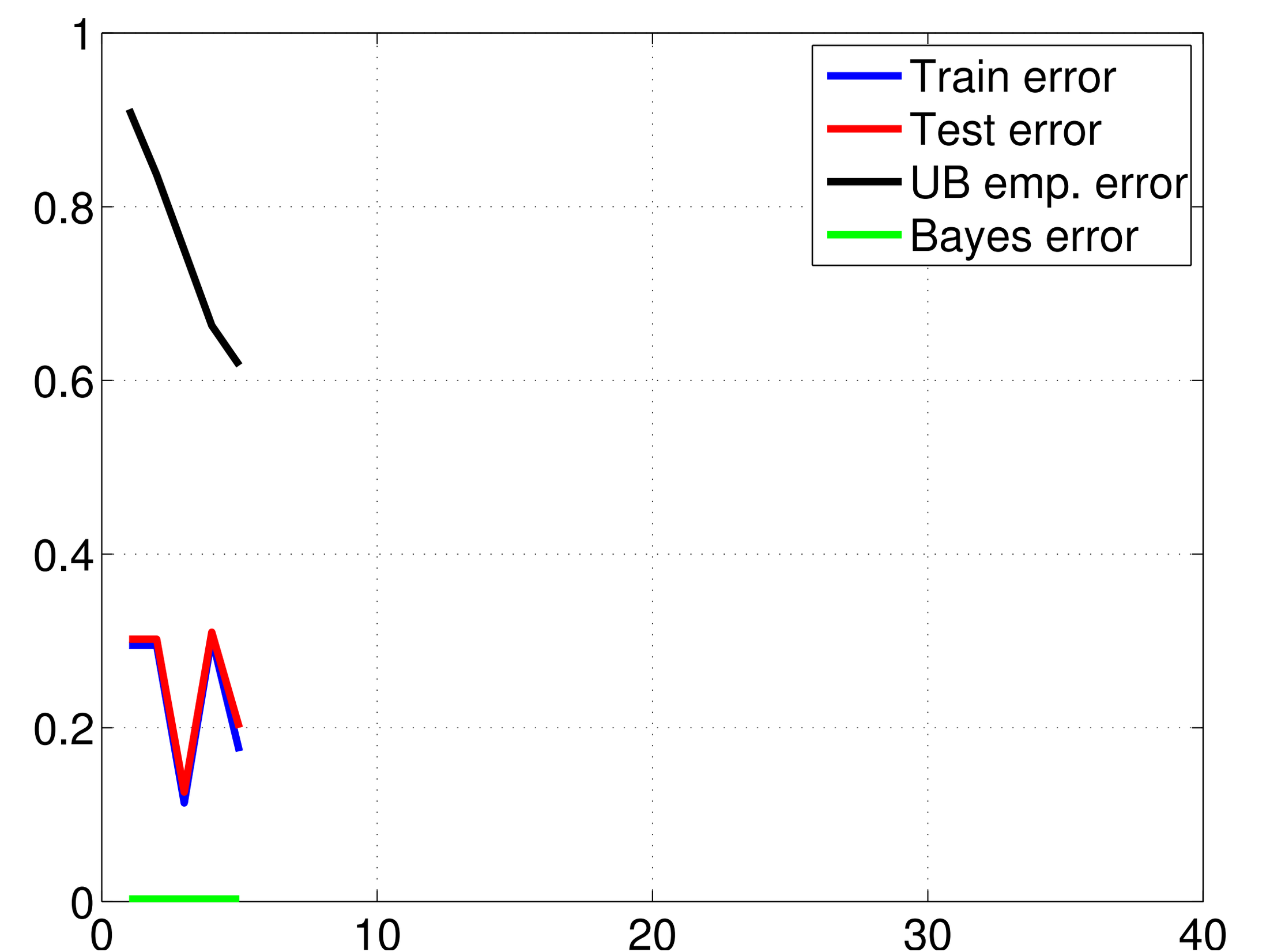


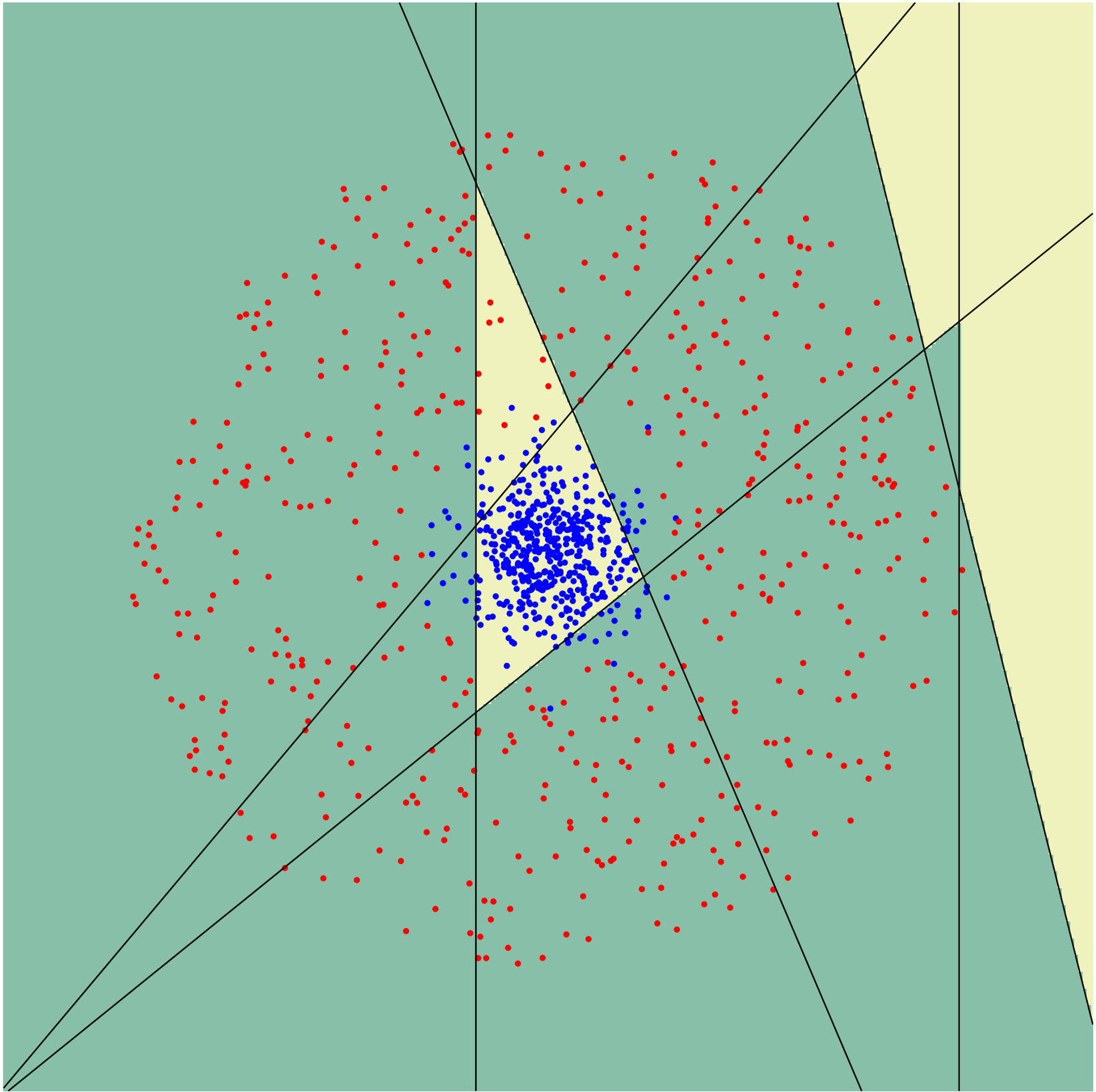


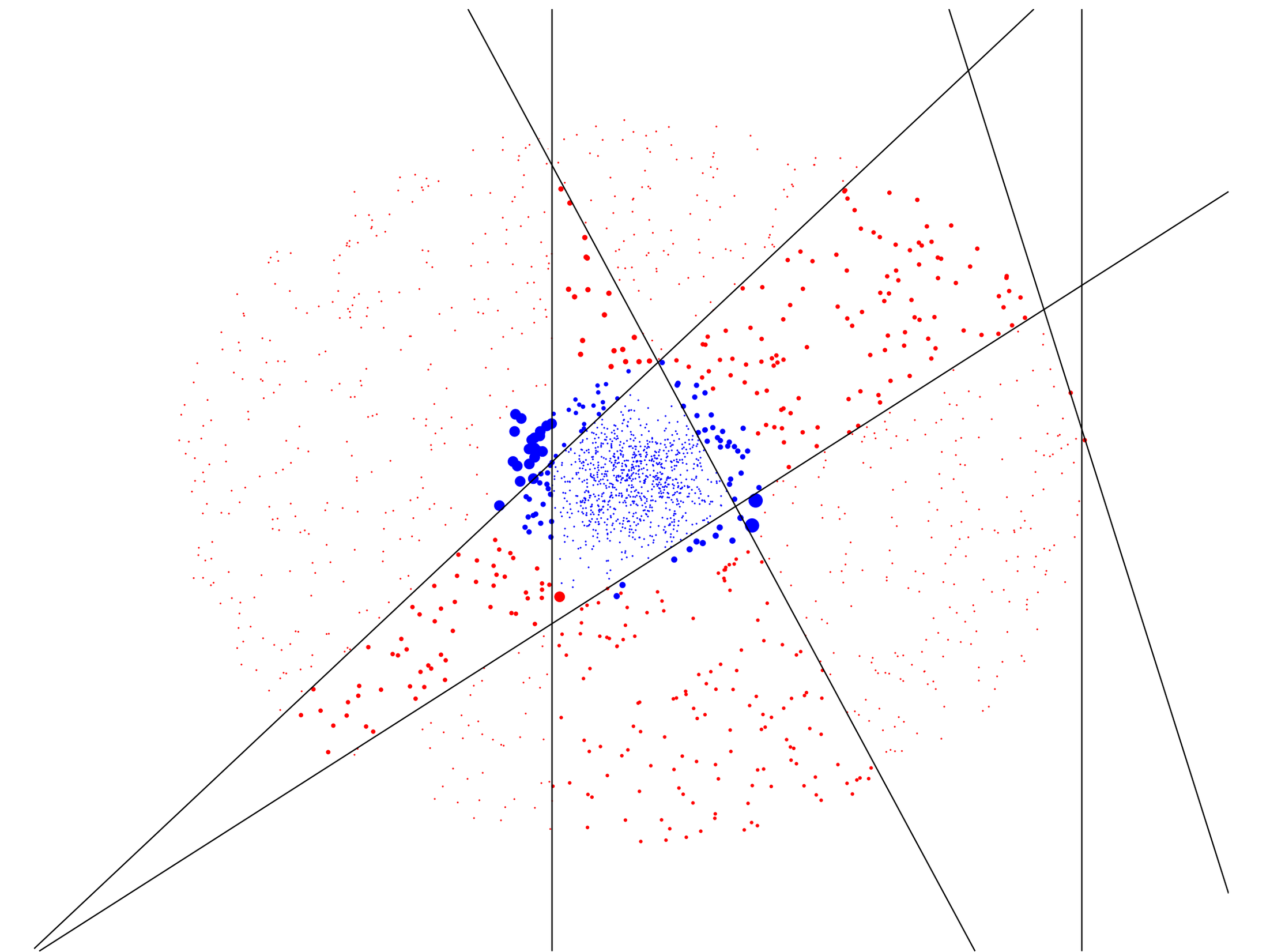

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

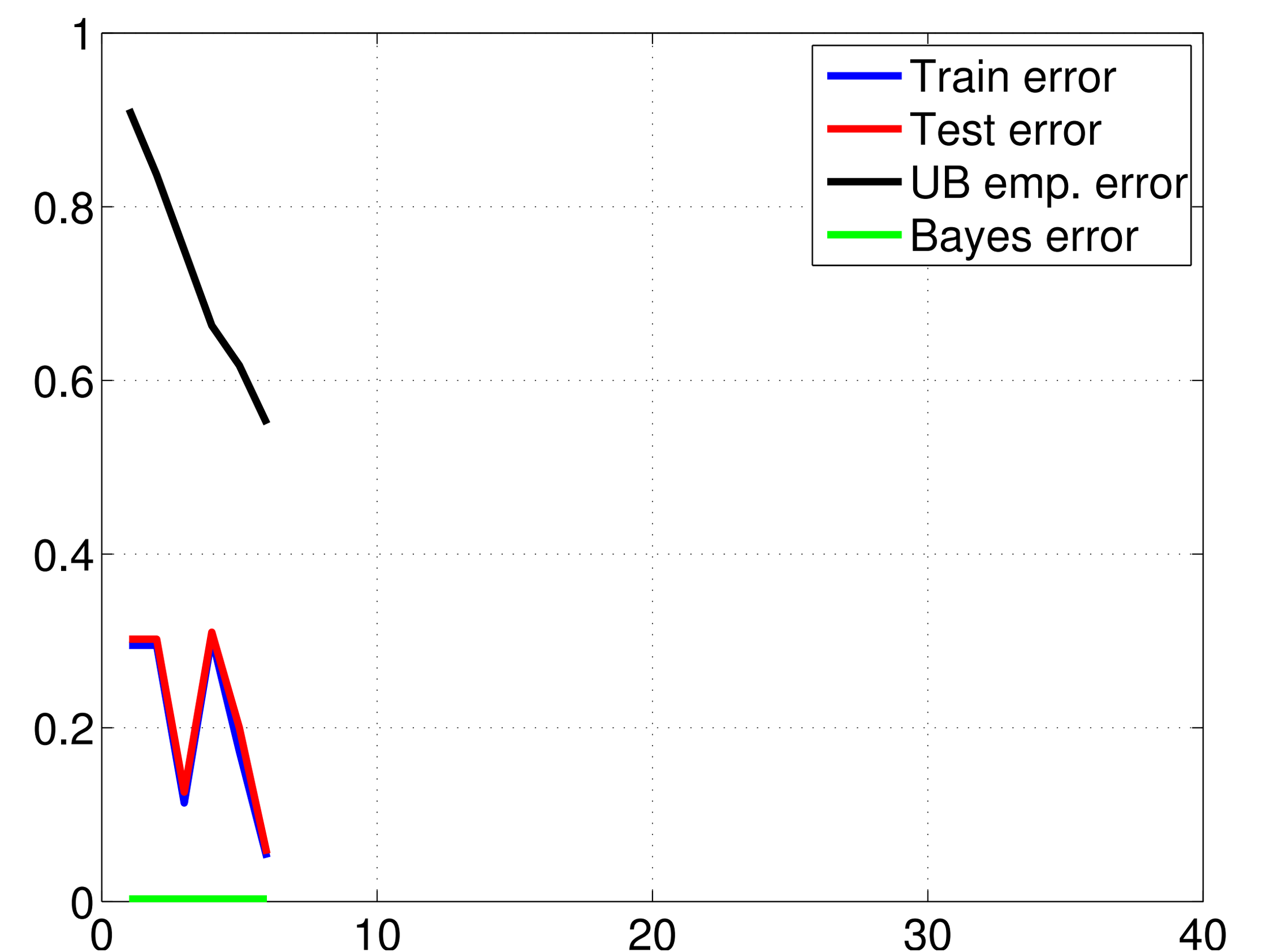


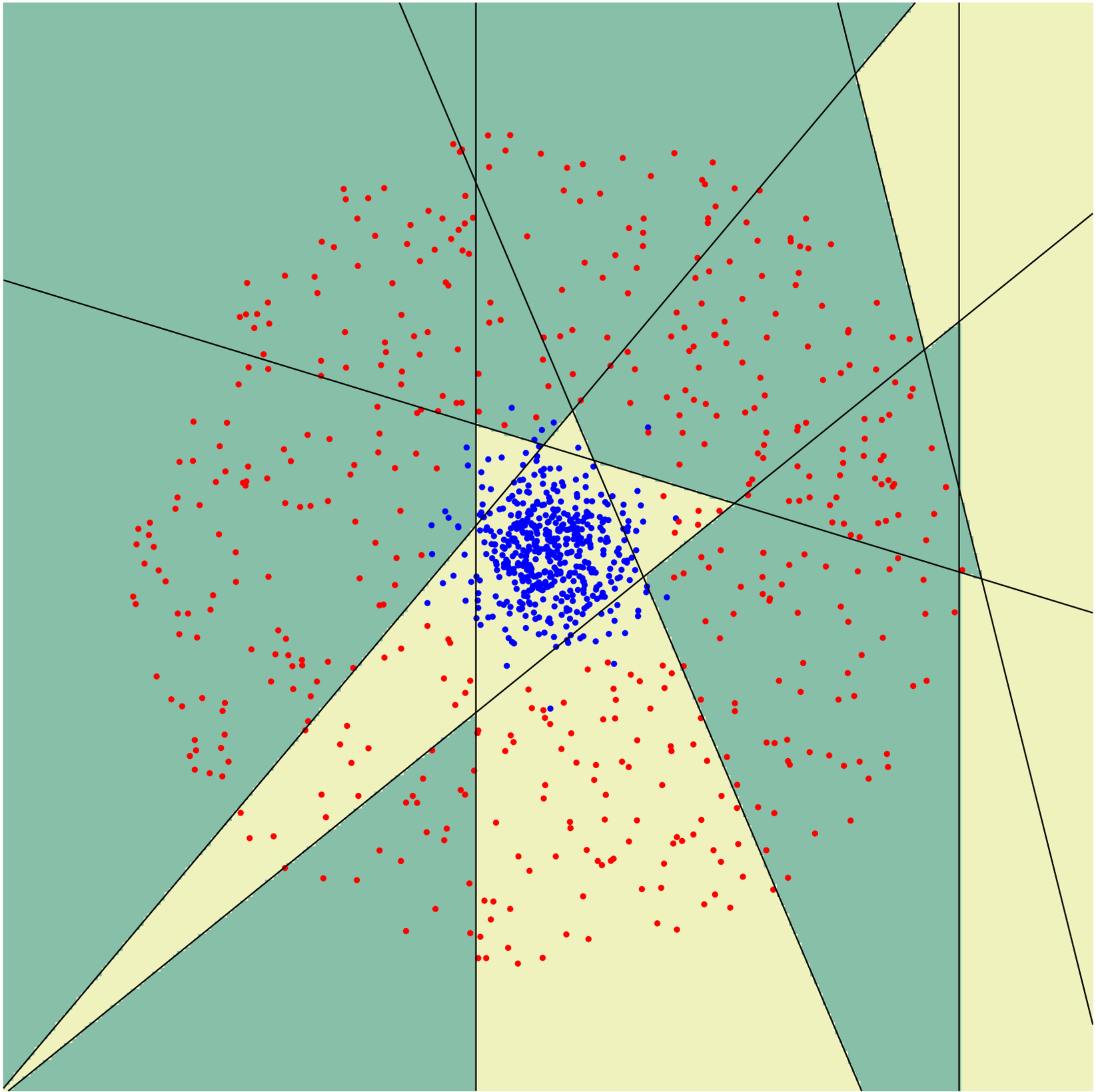


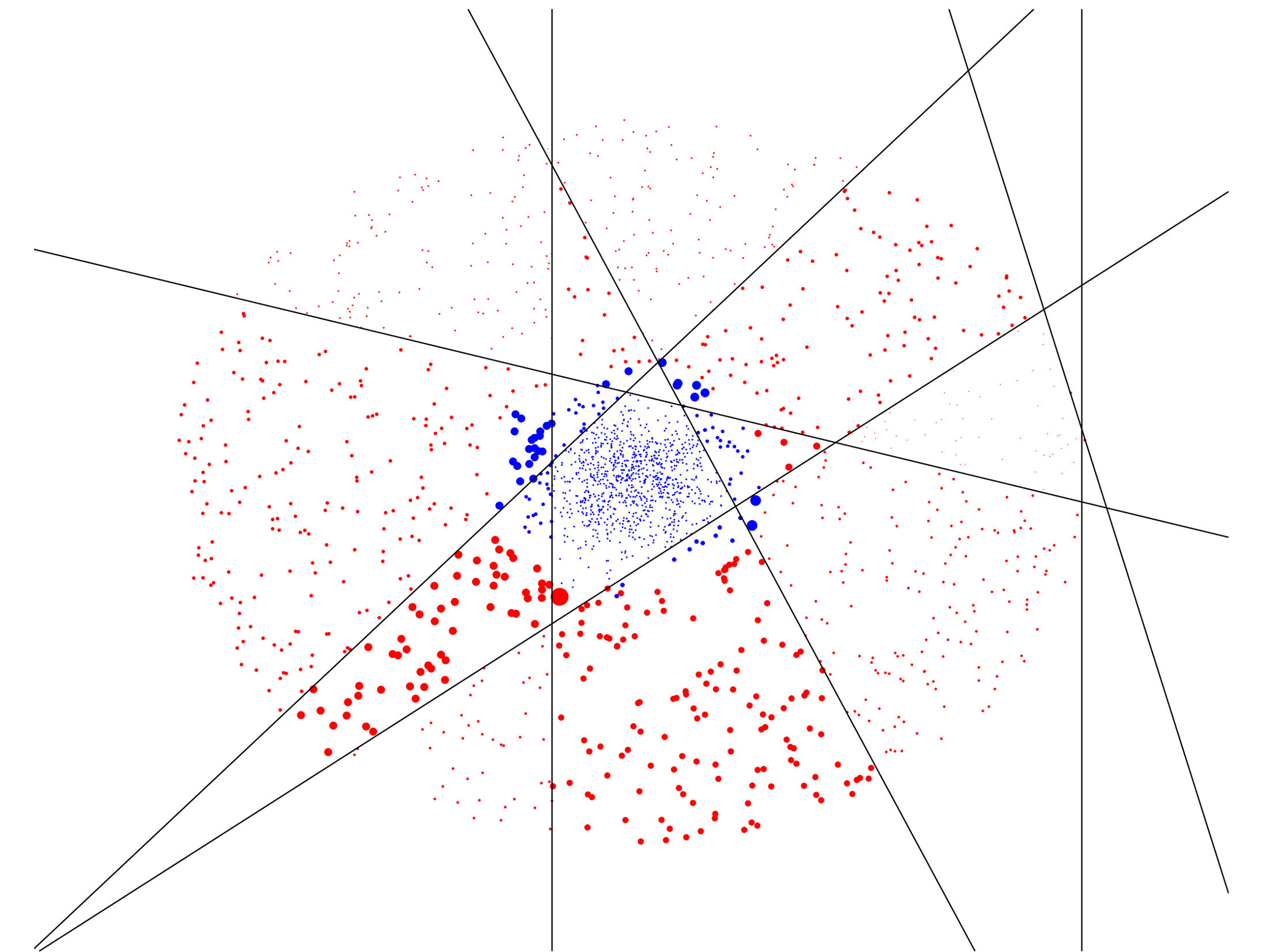

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

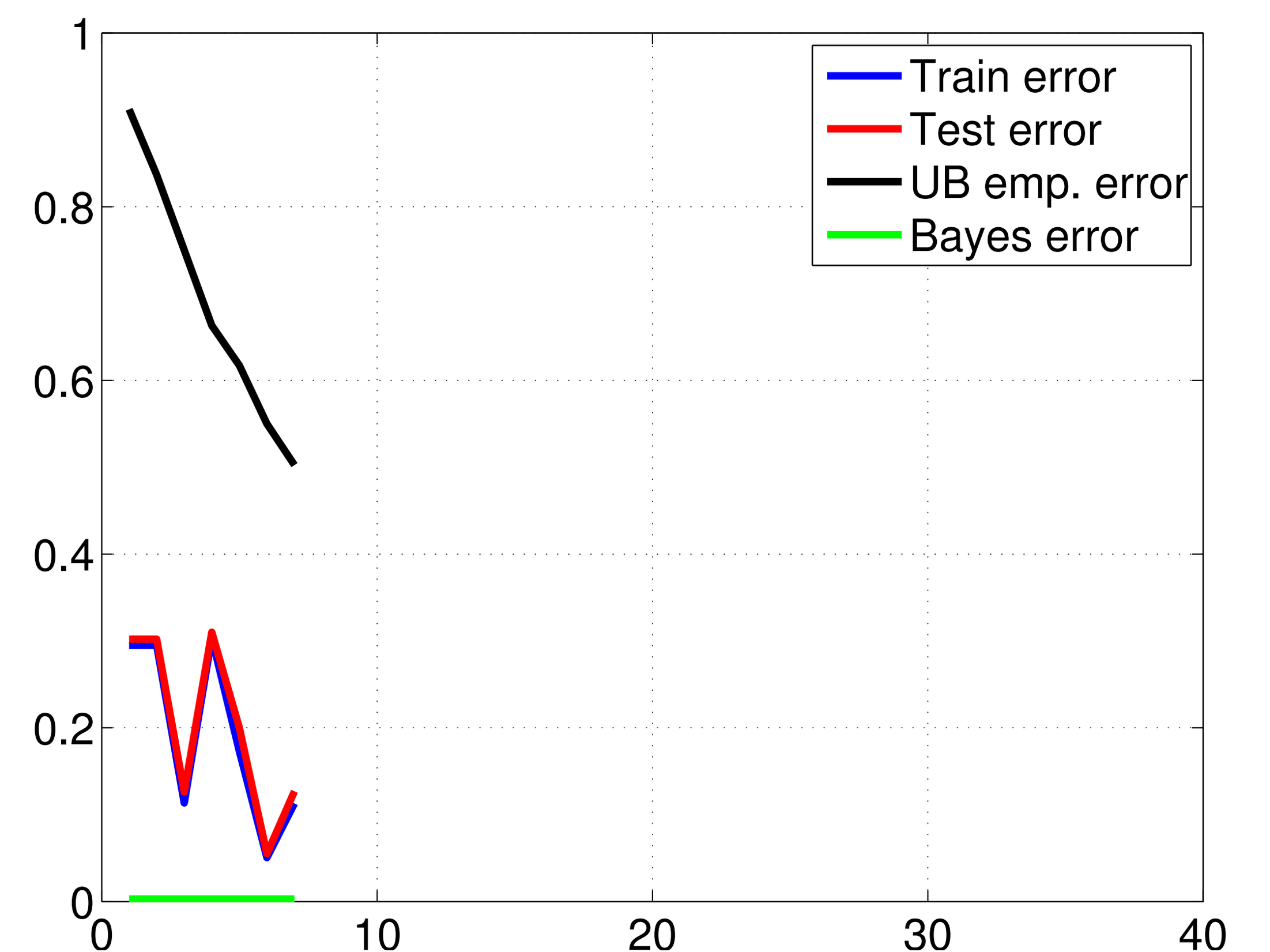


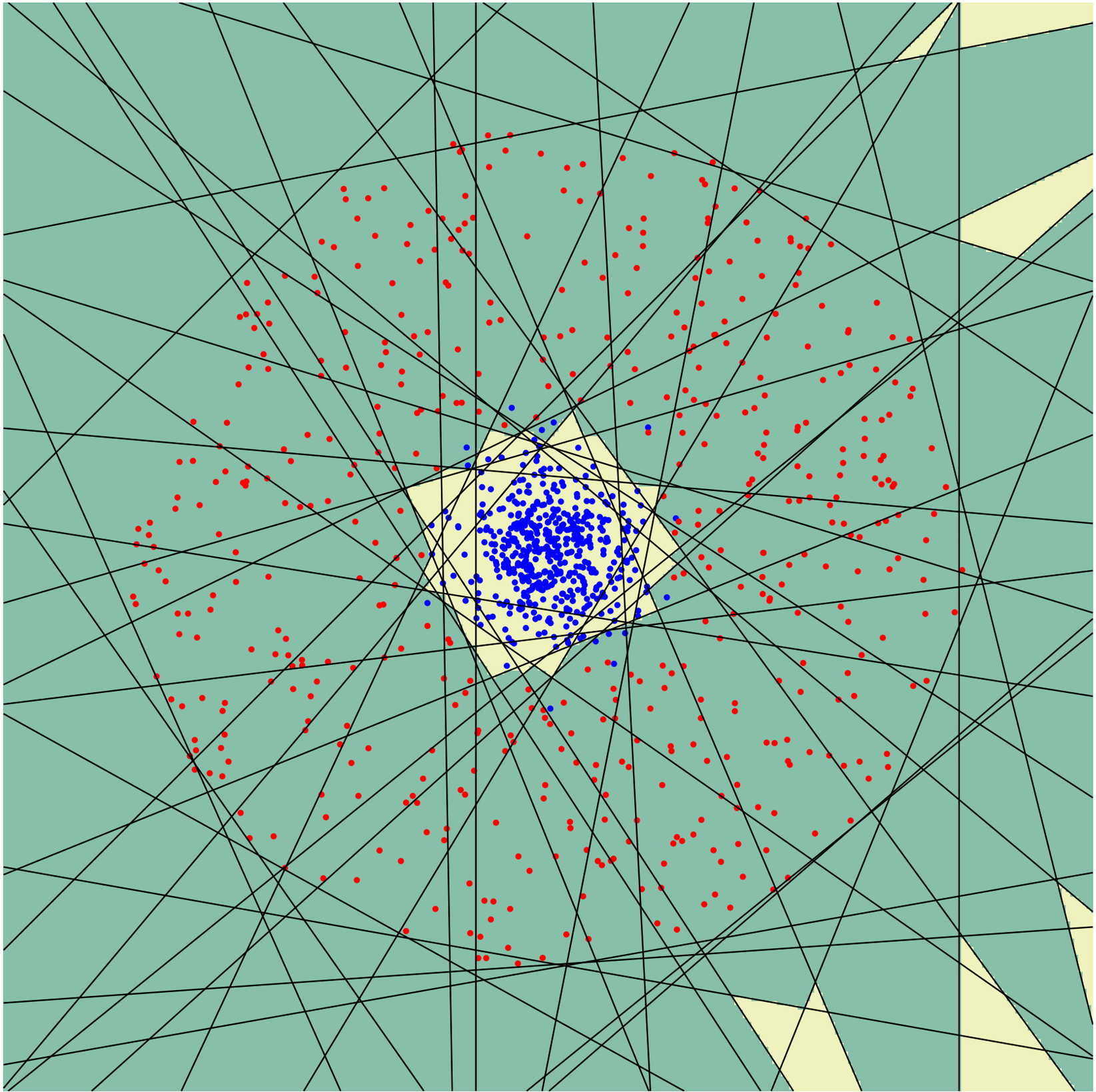


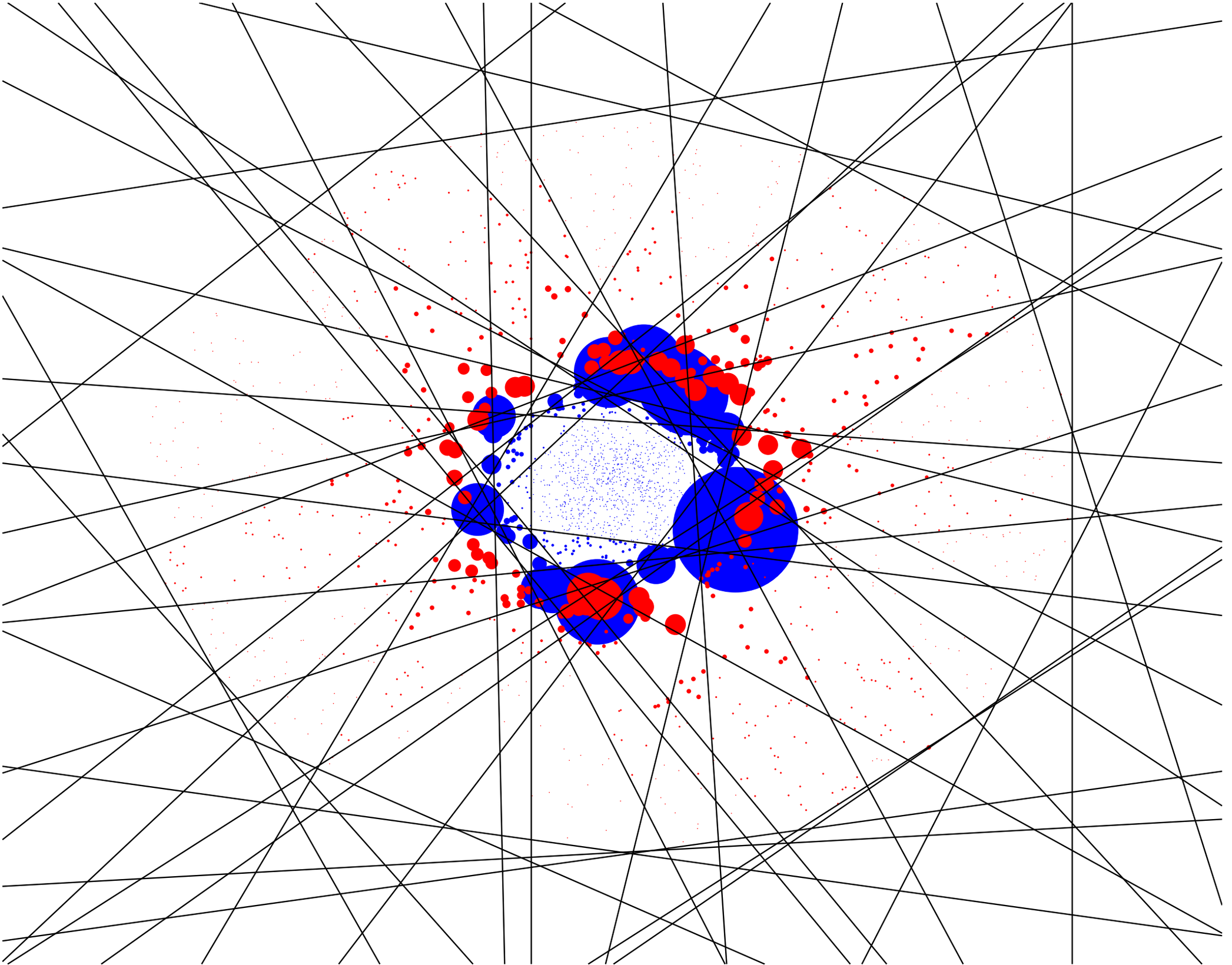

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

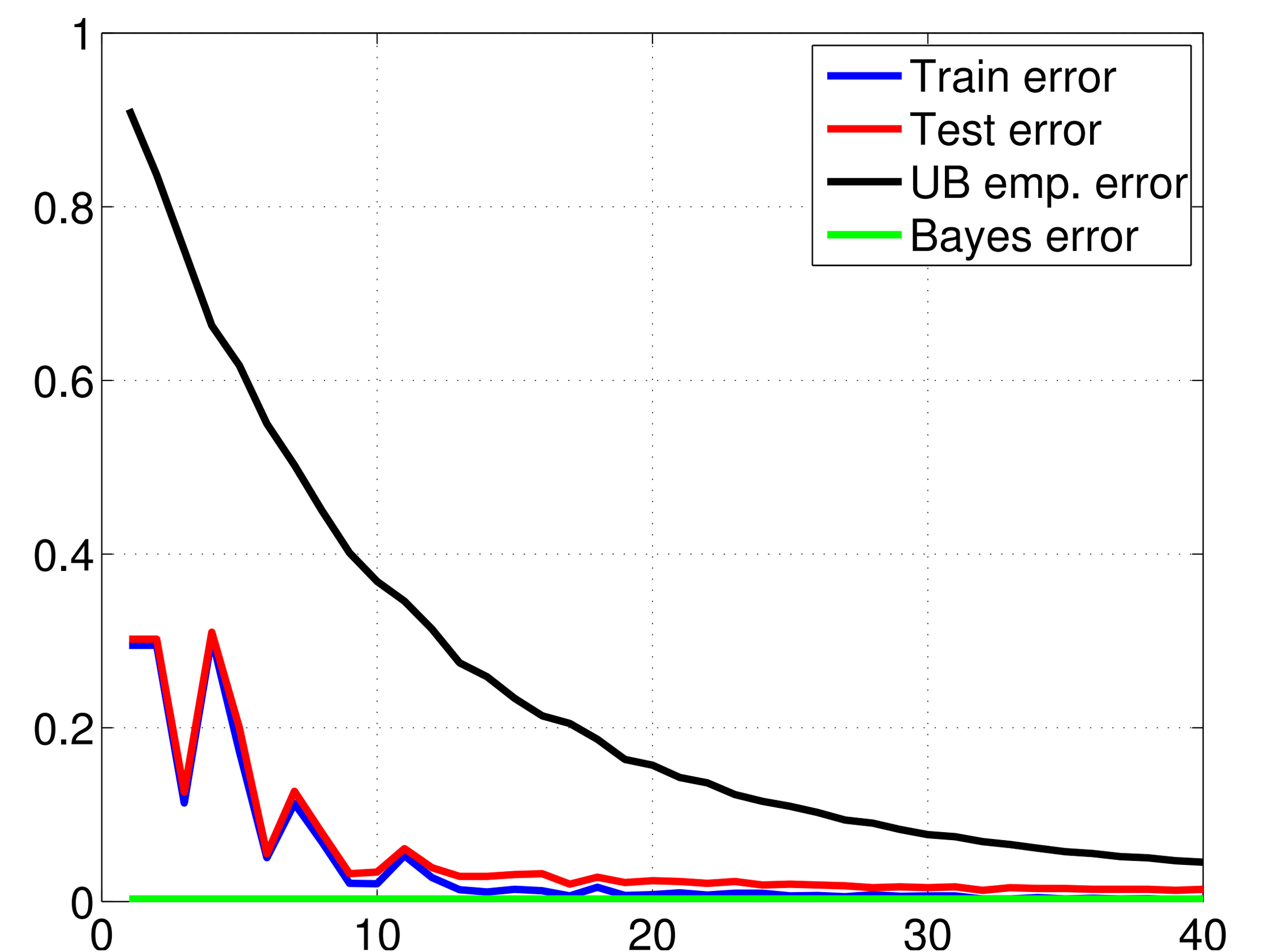


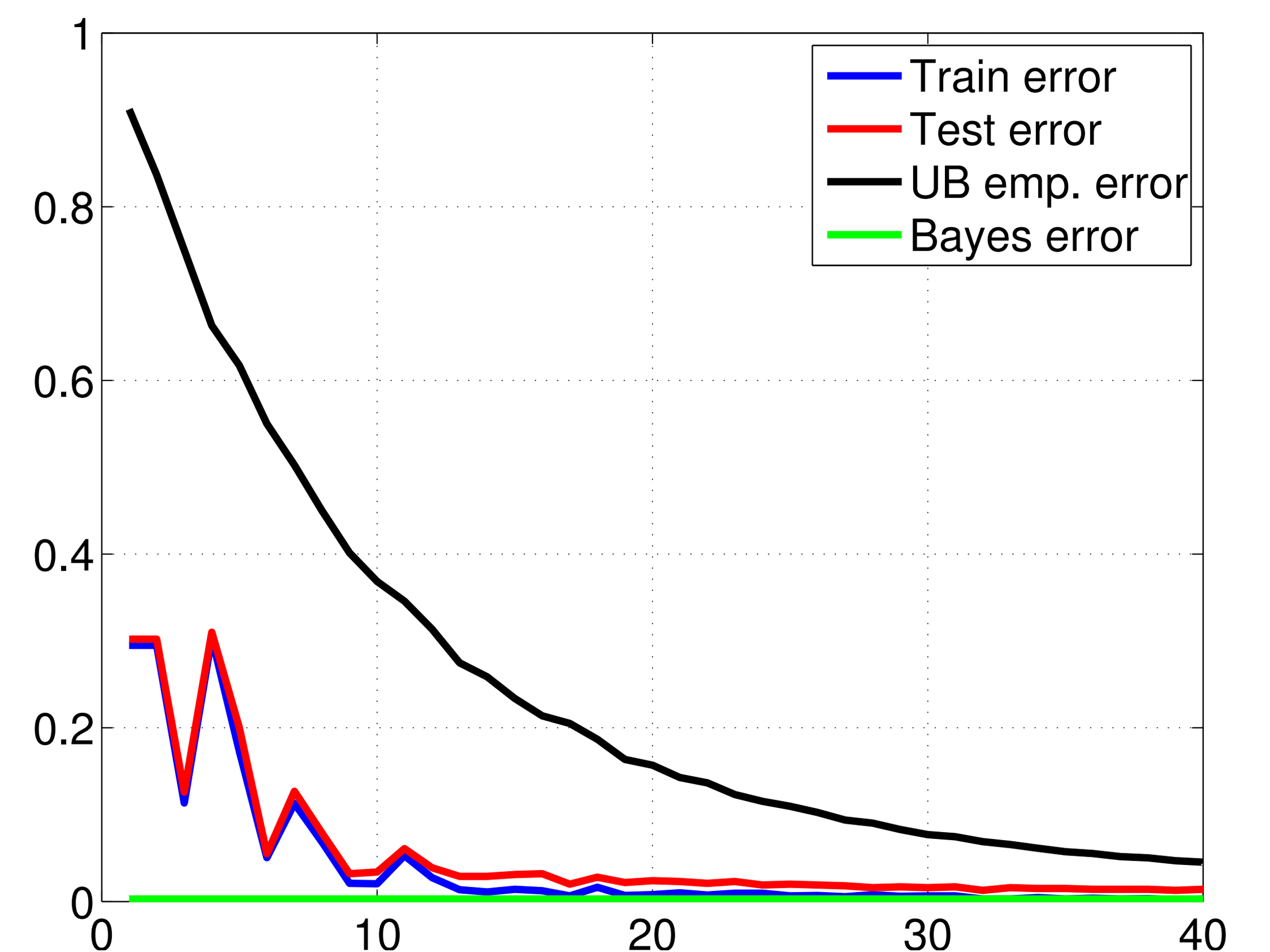


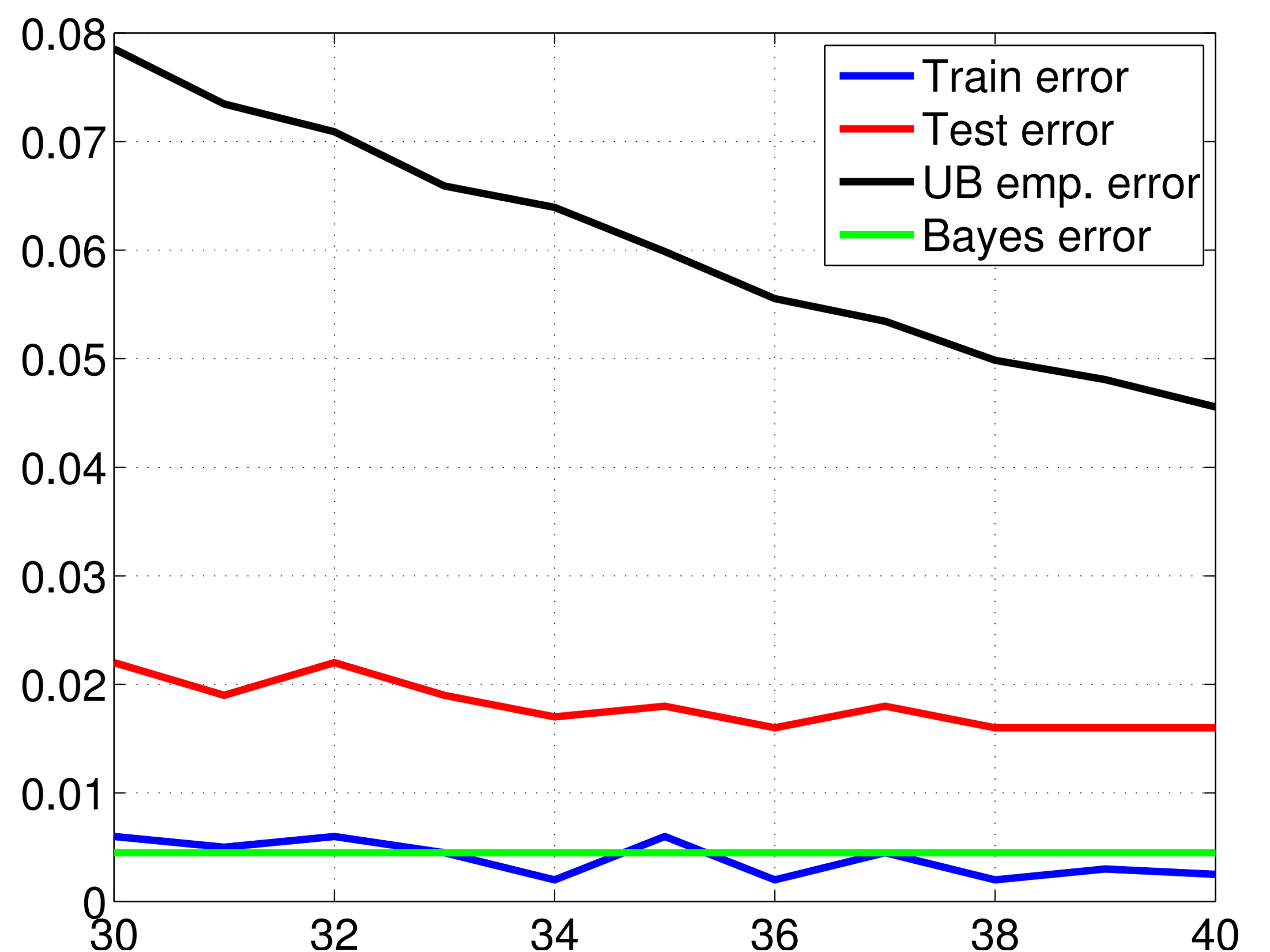

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

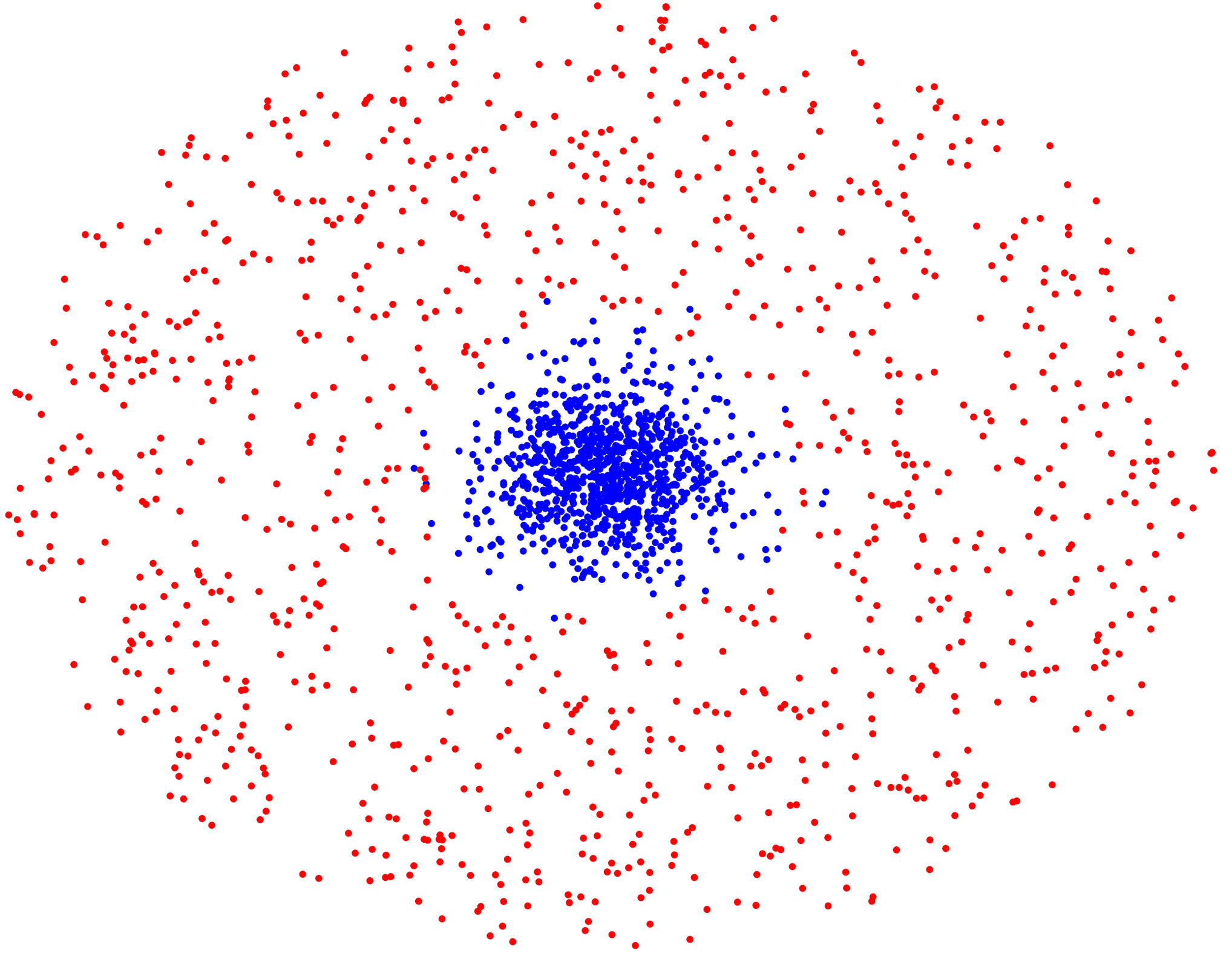


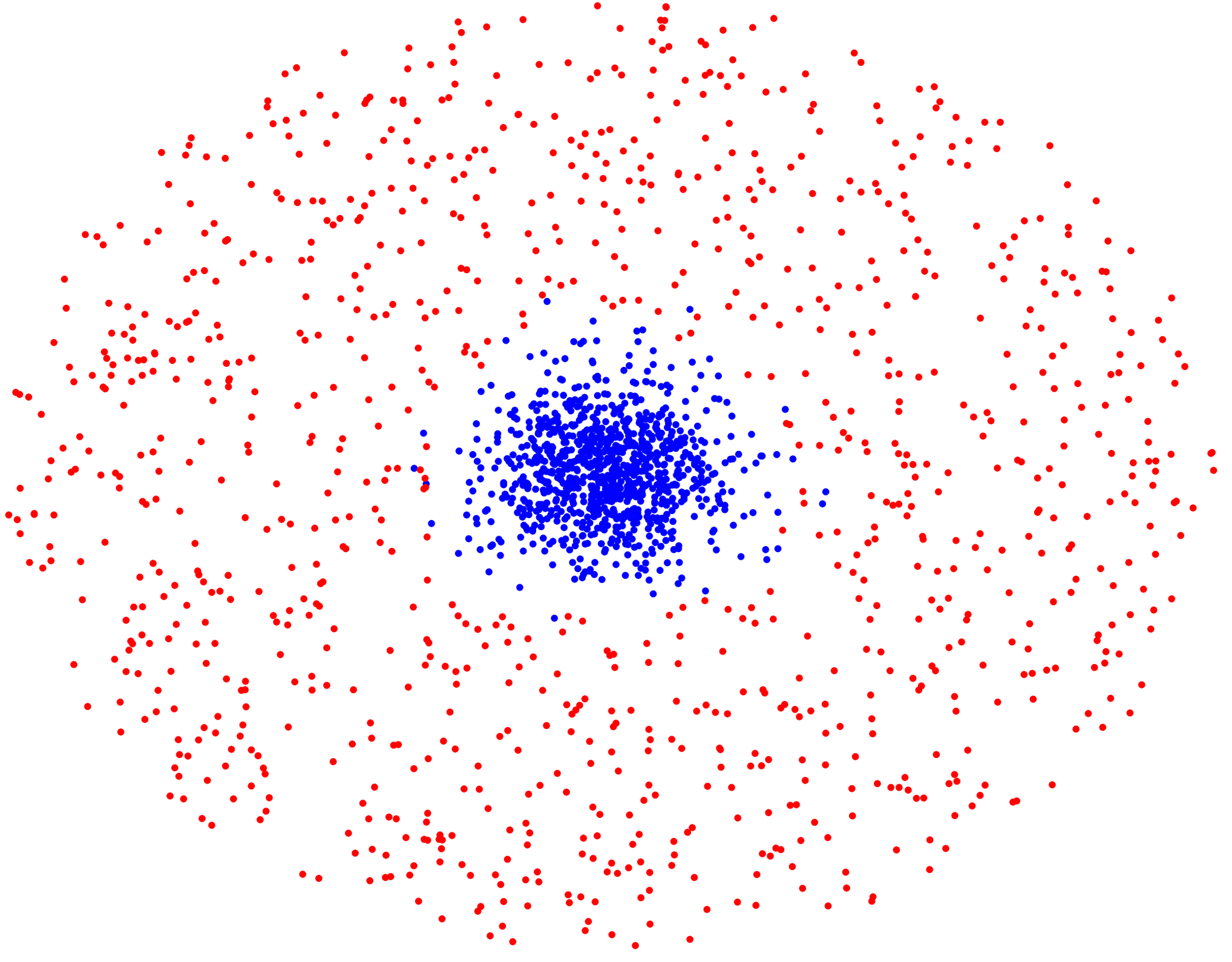


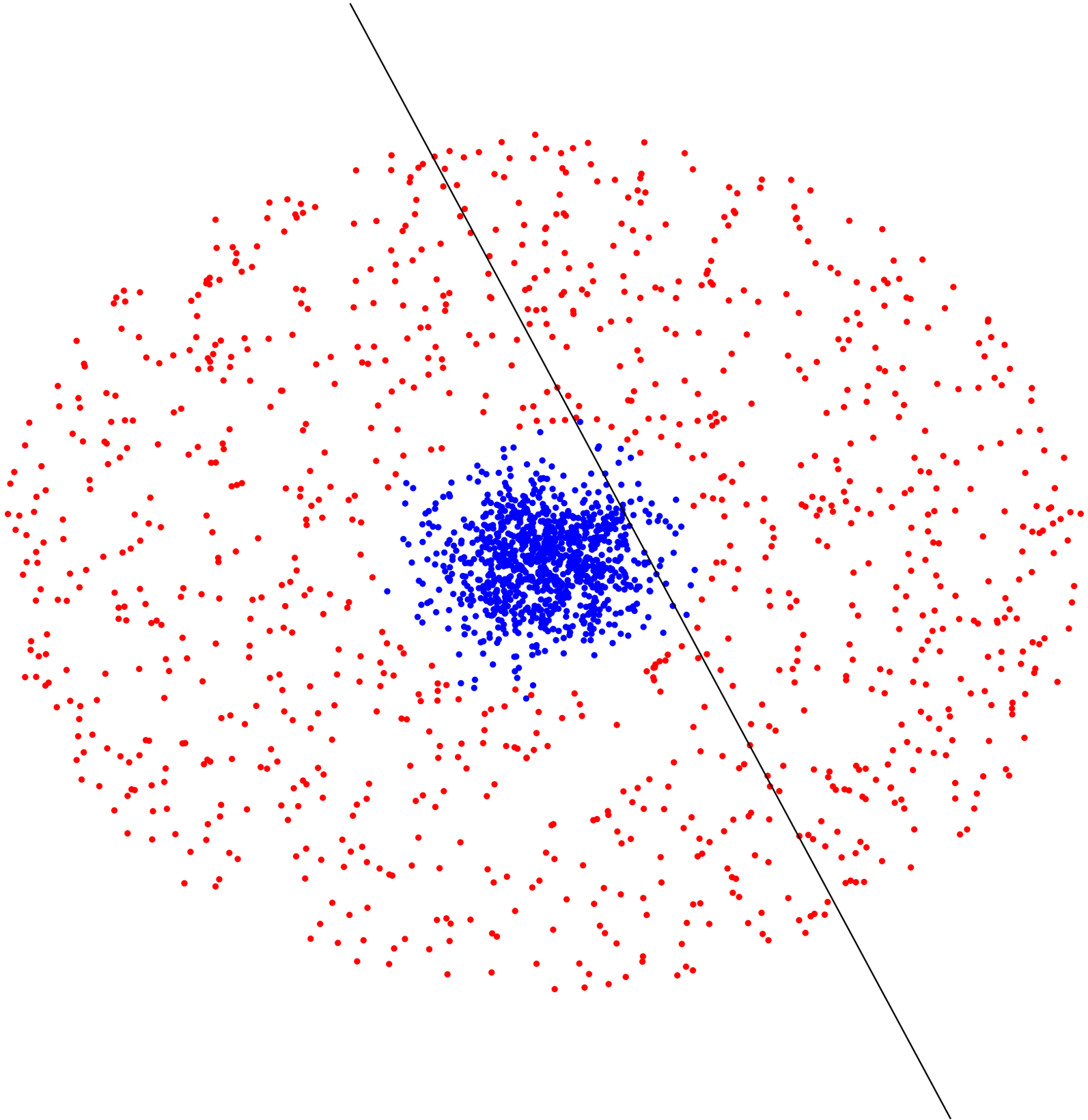

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

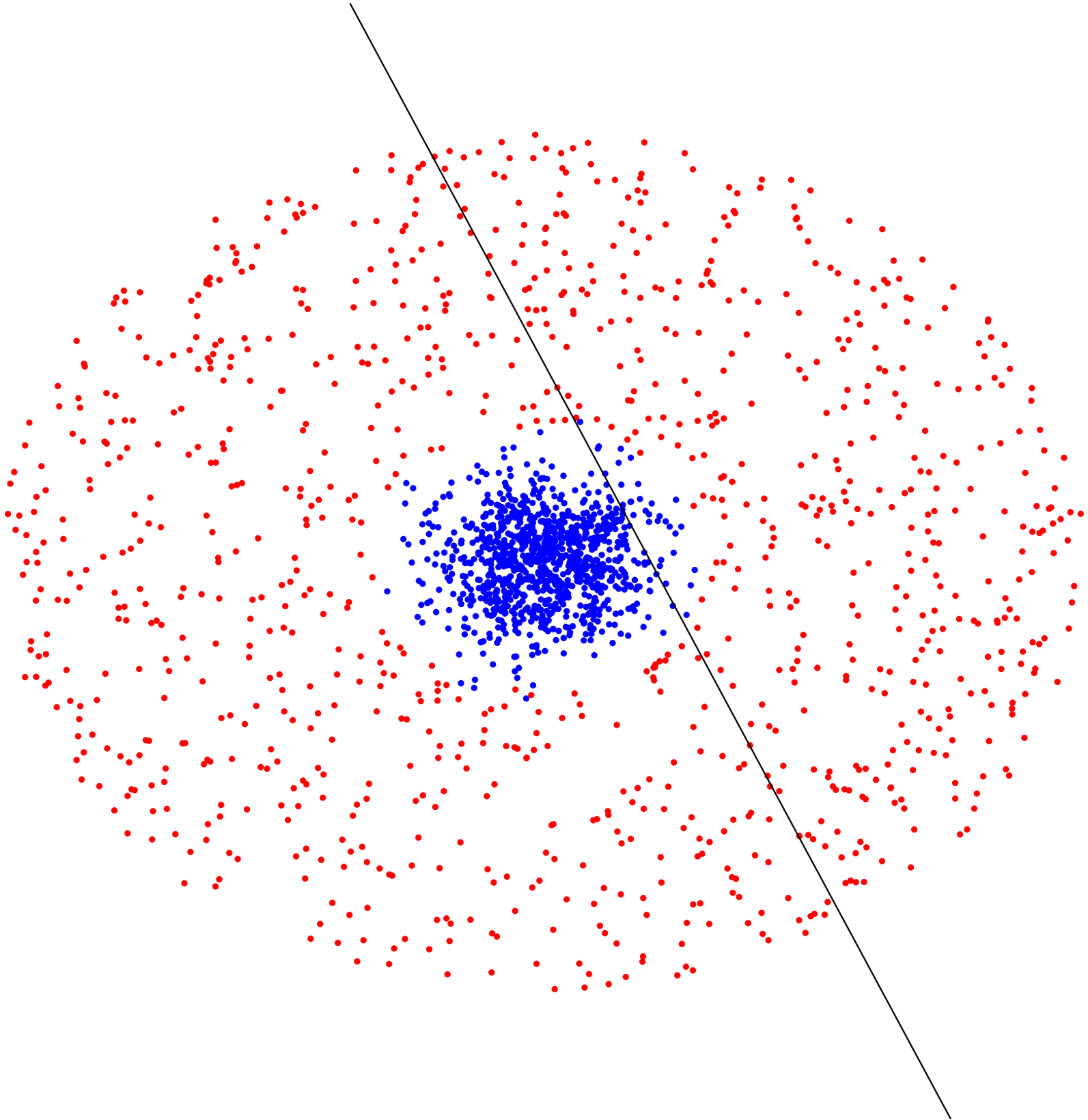

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

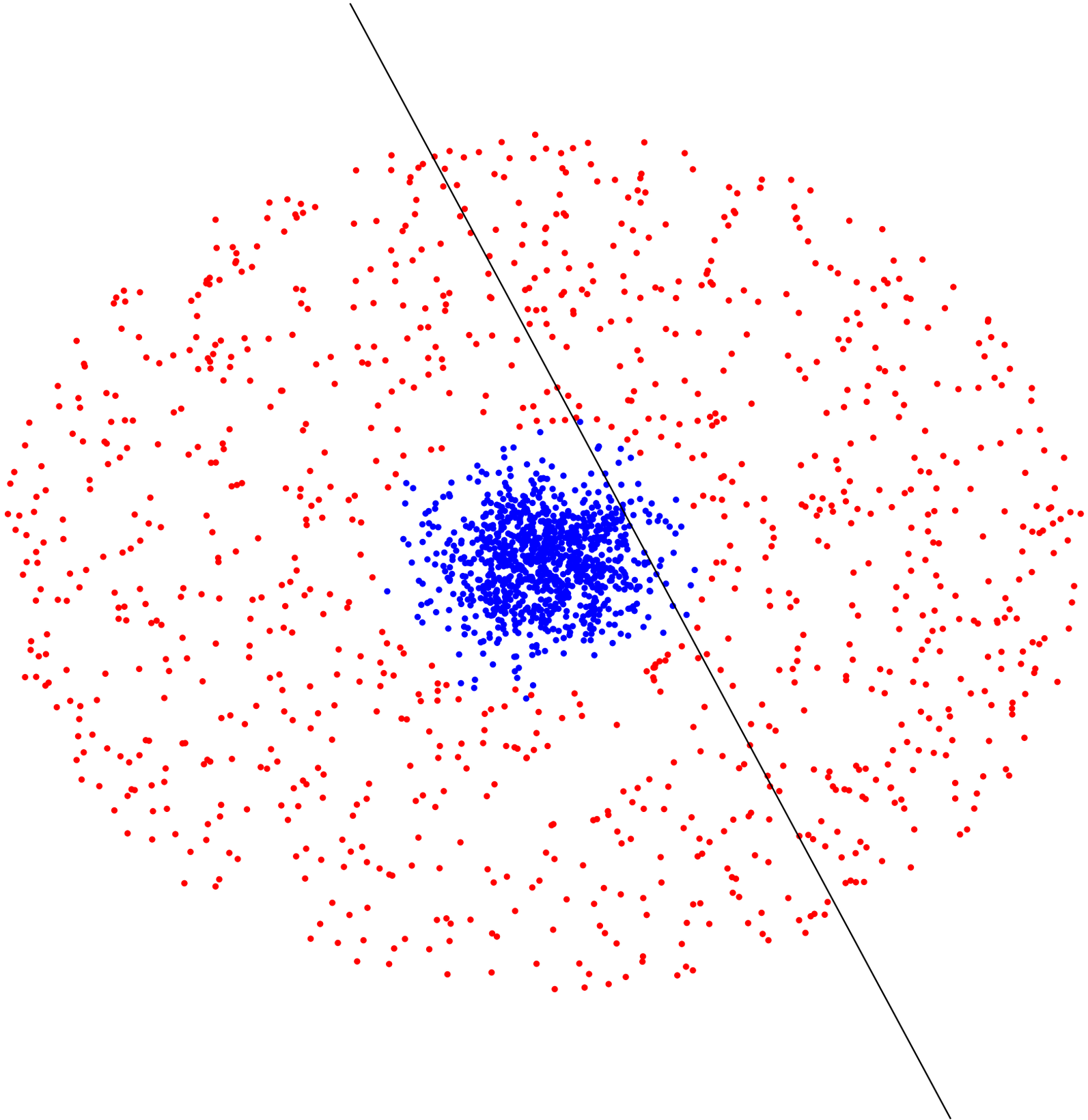

0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
30
32
34
36
38
40
Train error
Test error
UB emp. error
Bayes error

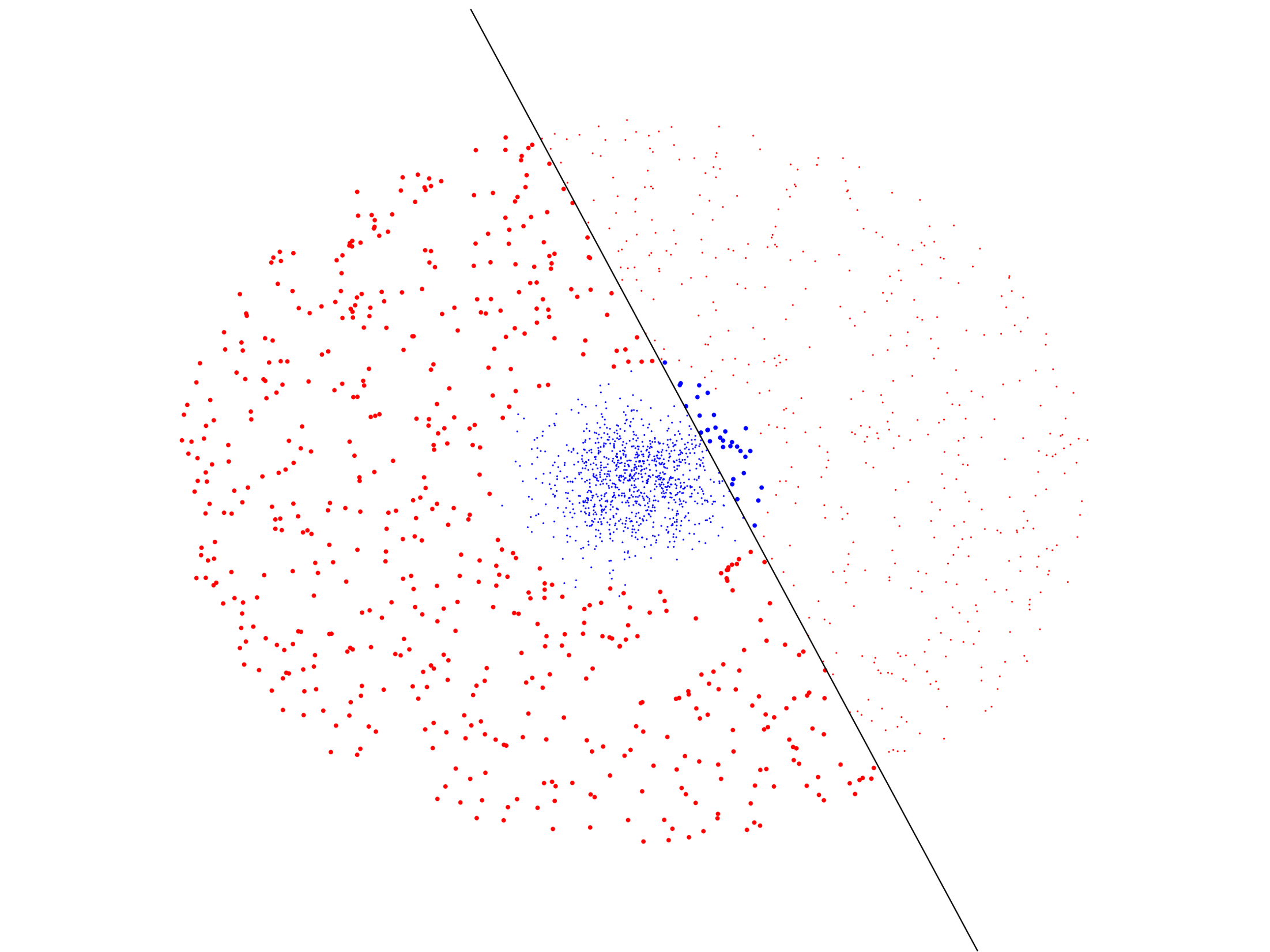


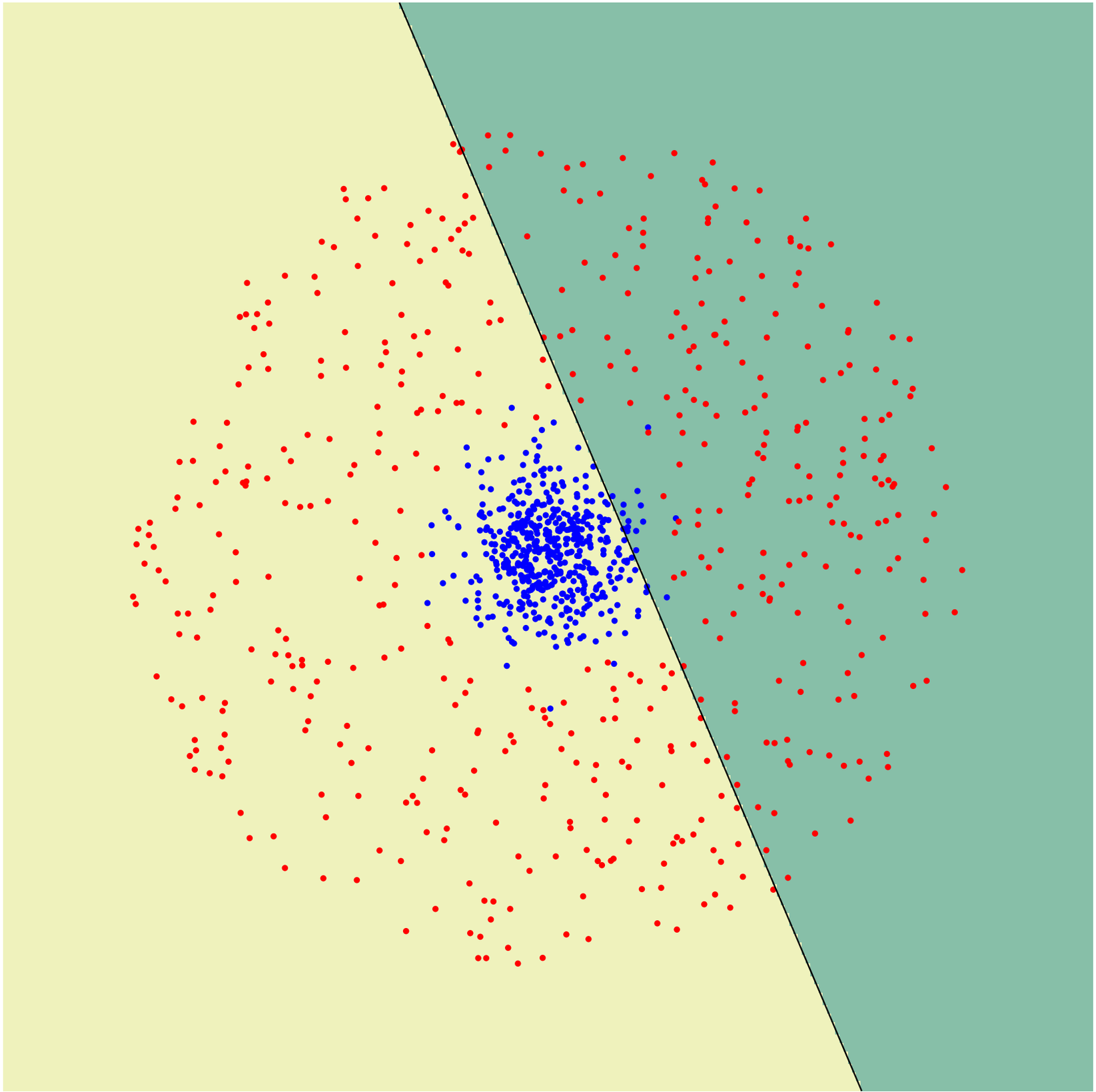


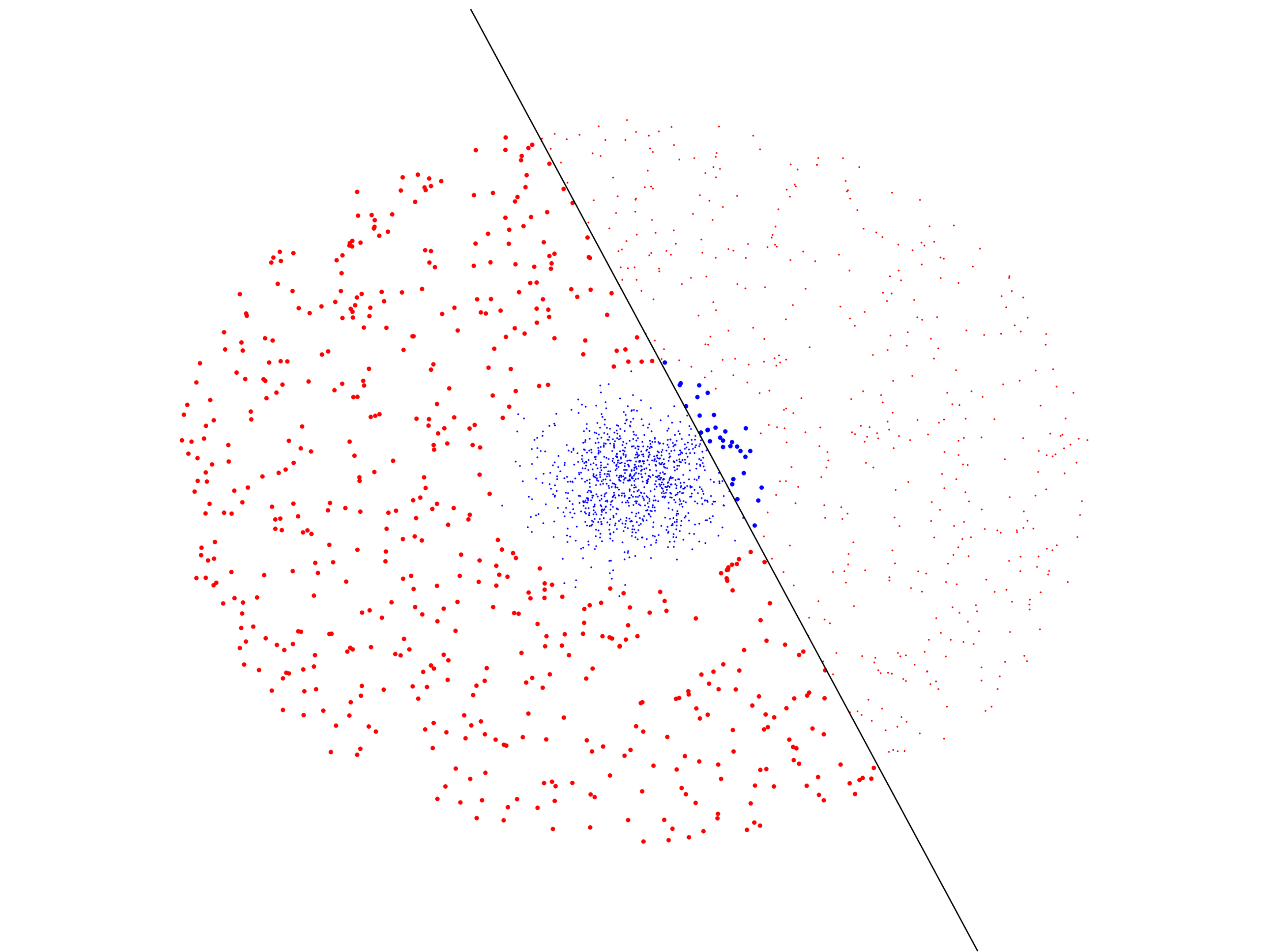


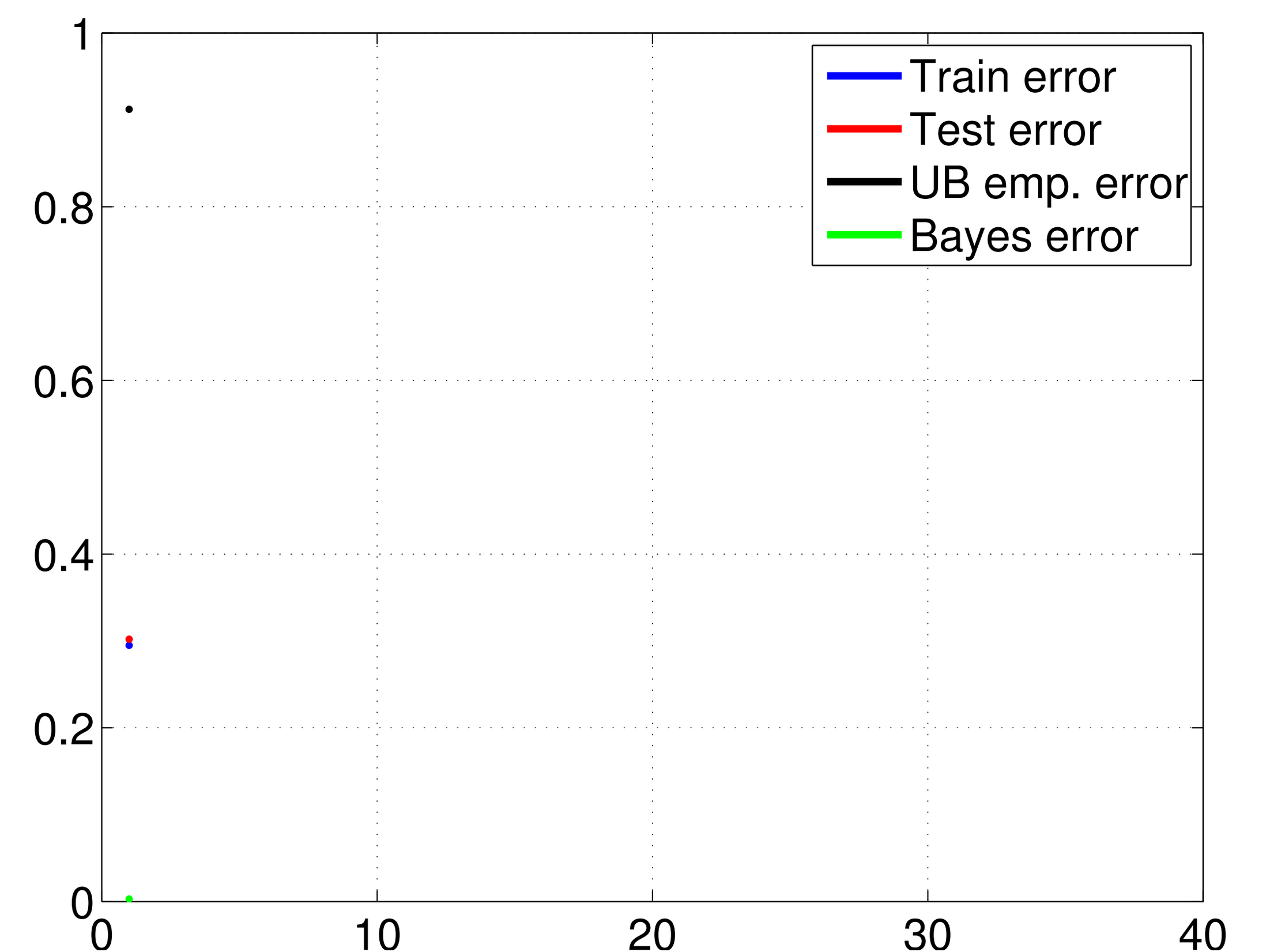


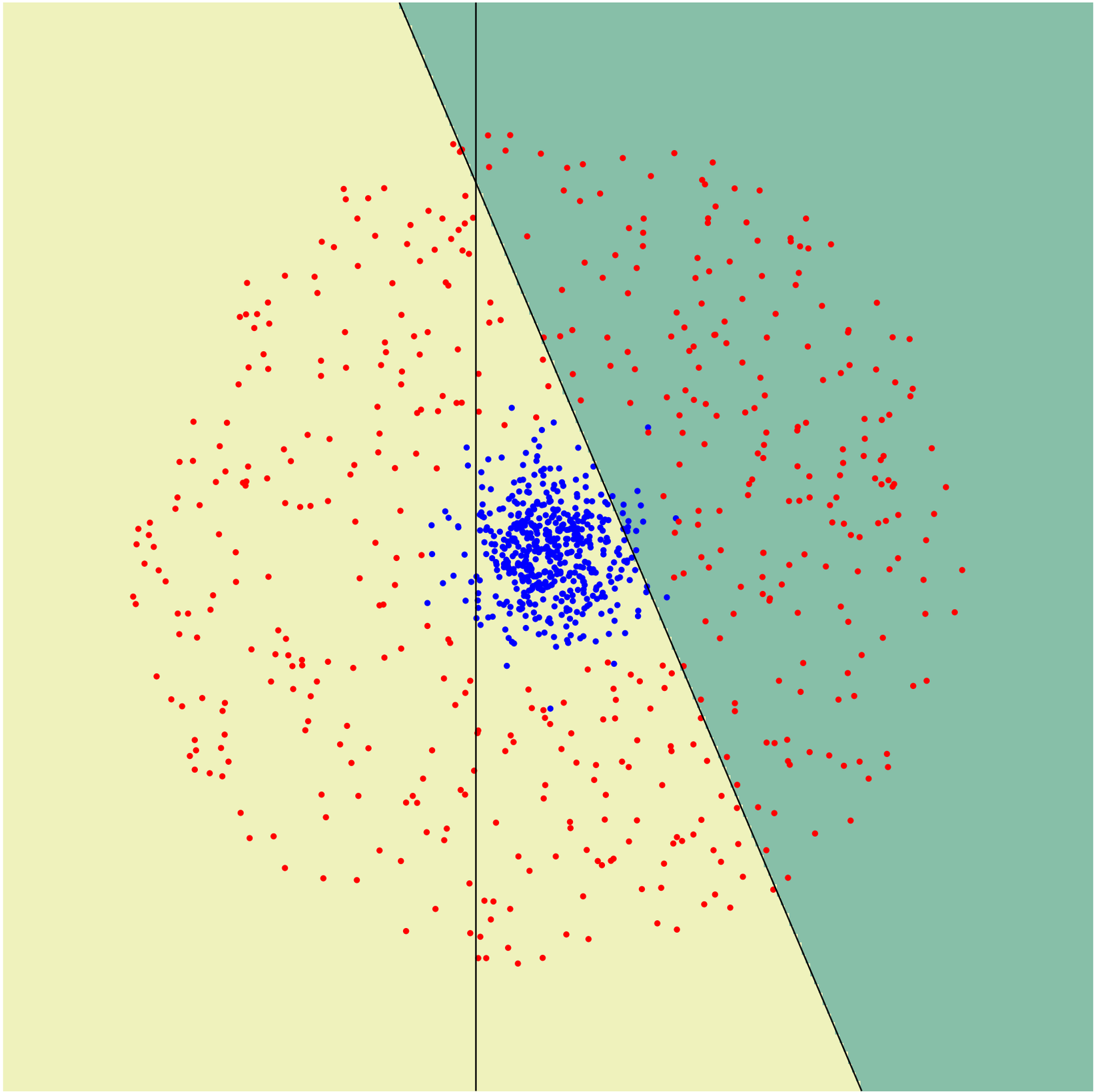


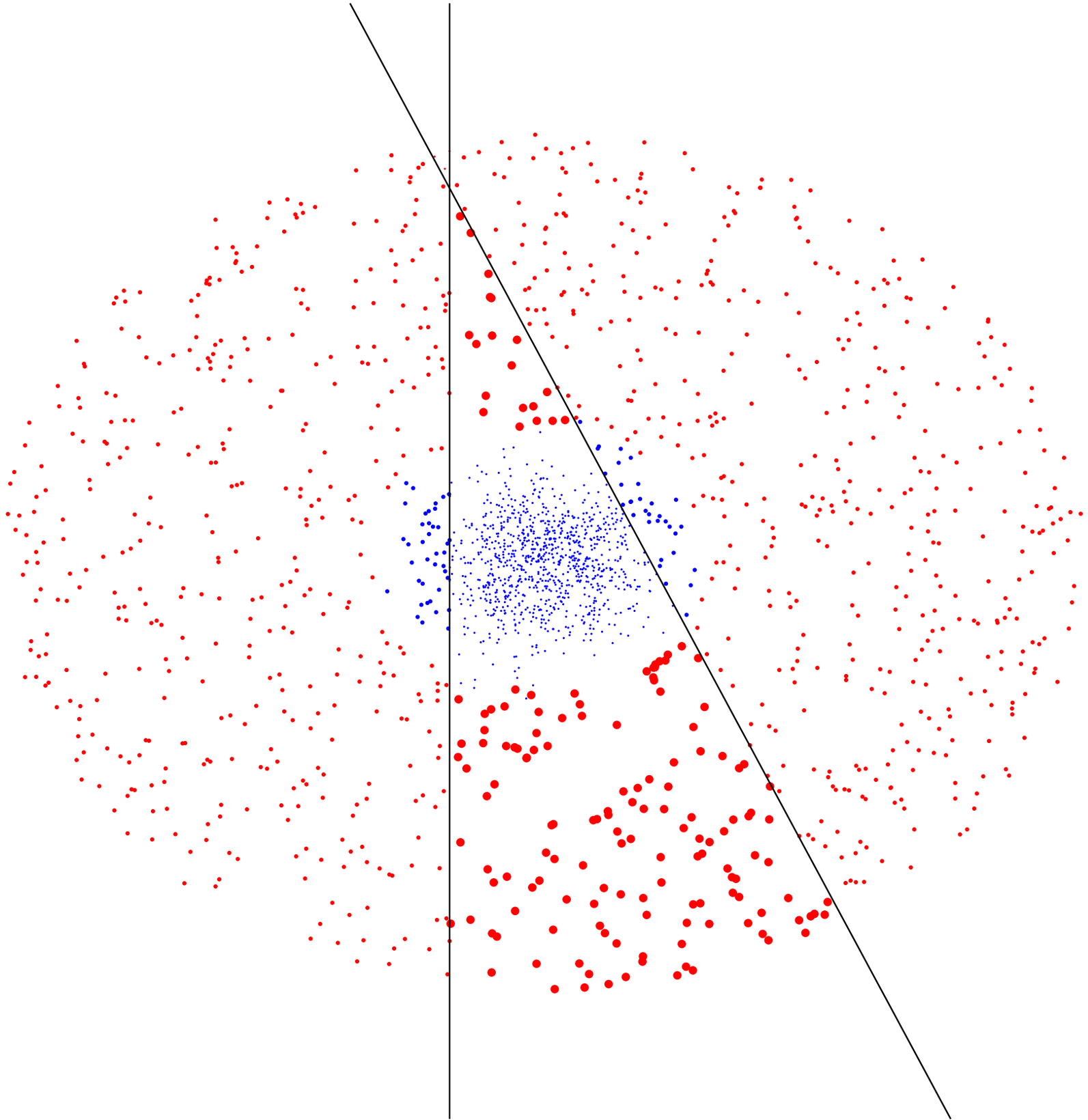


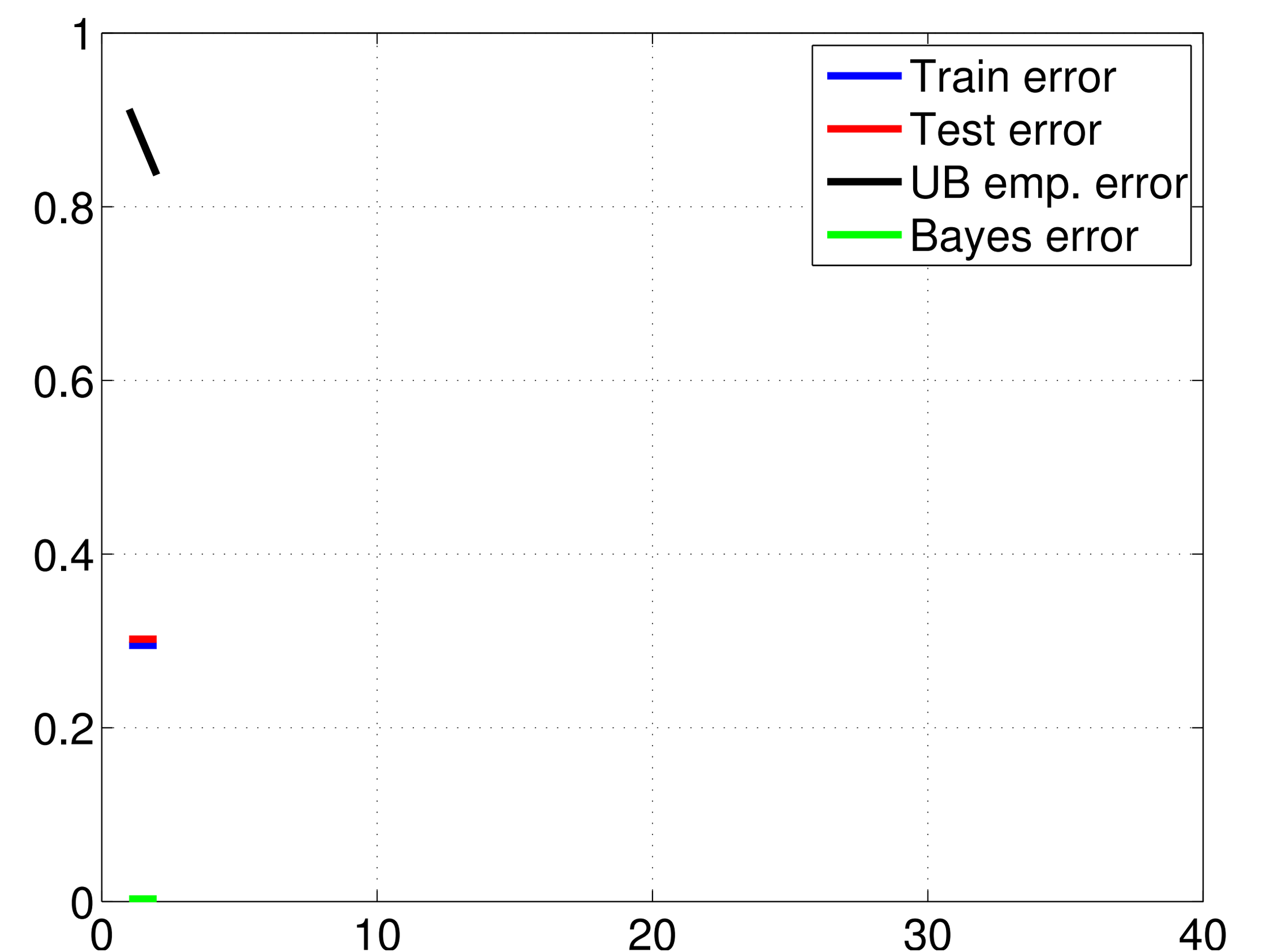


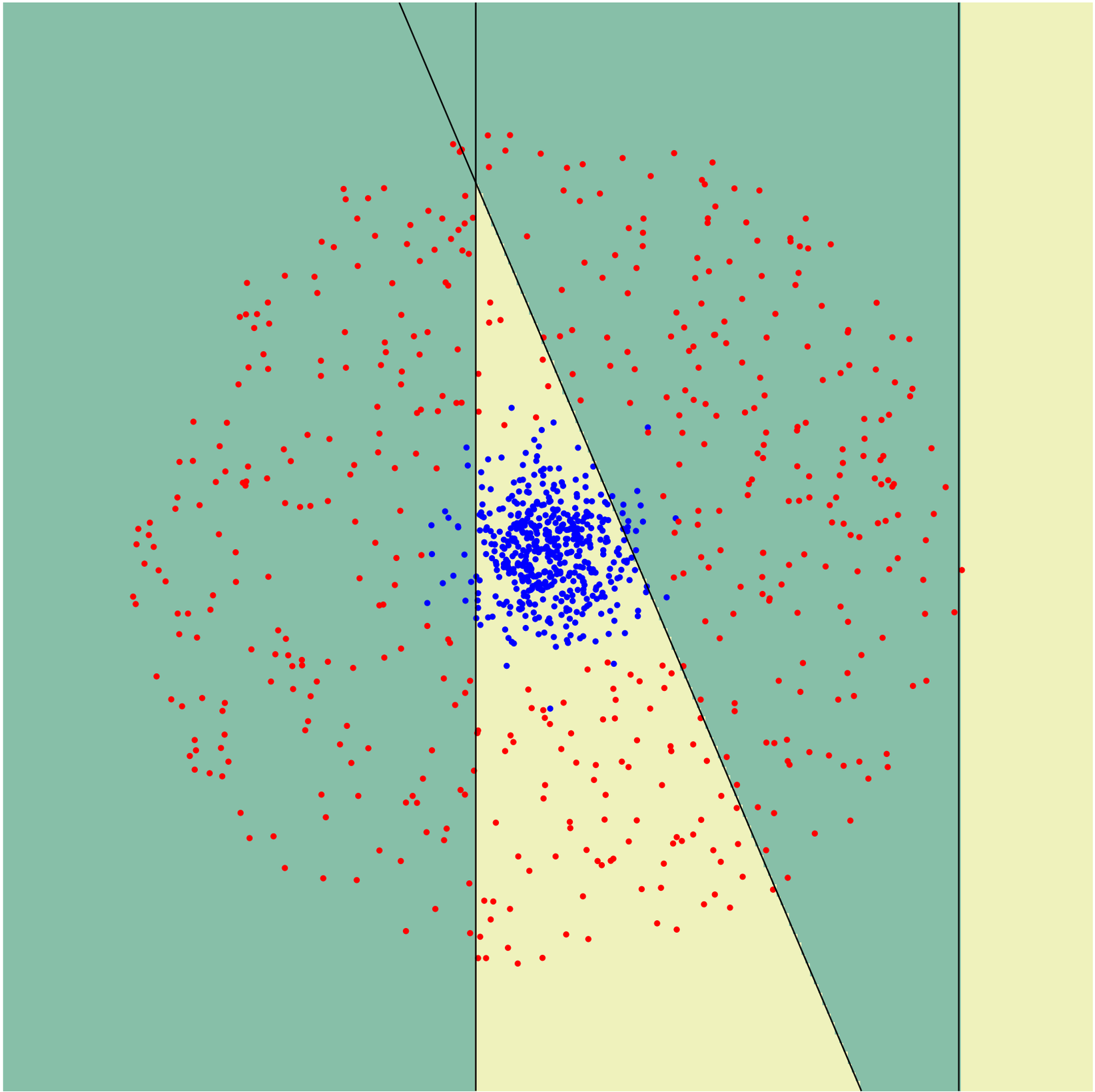


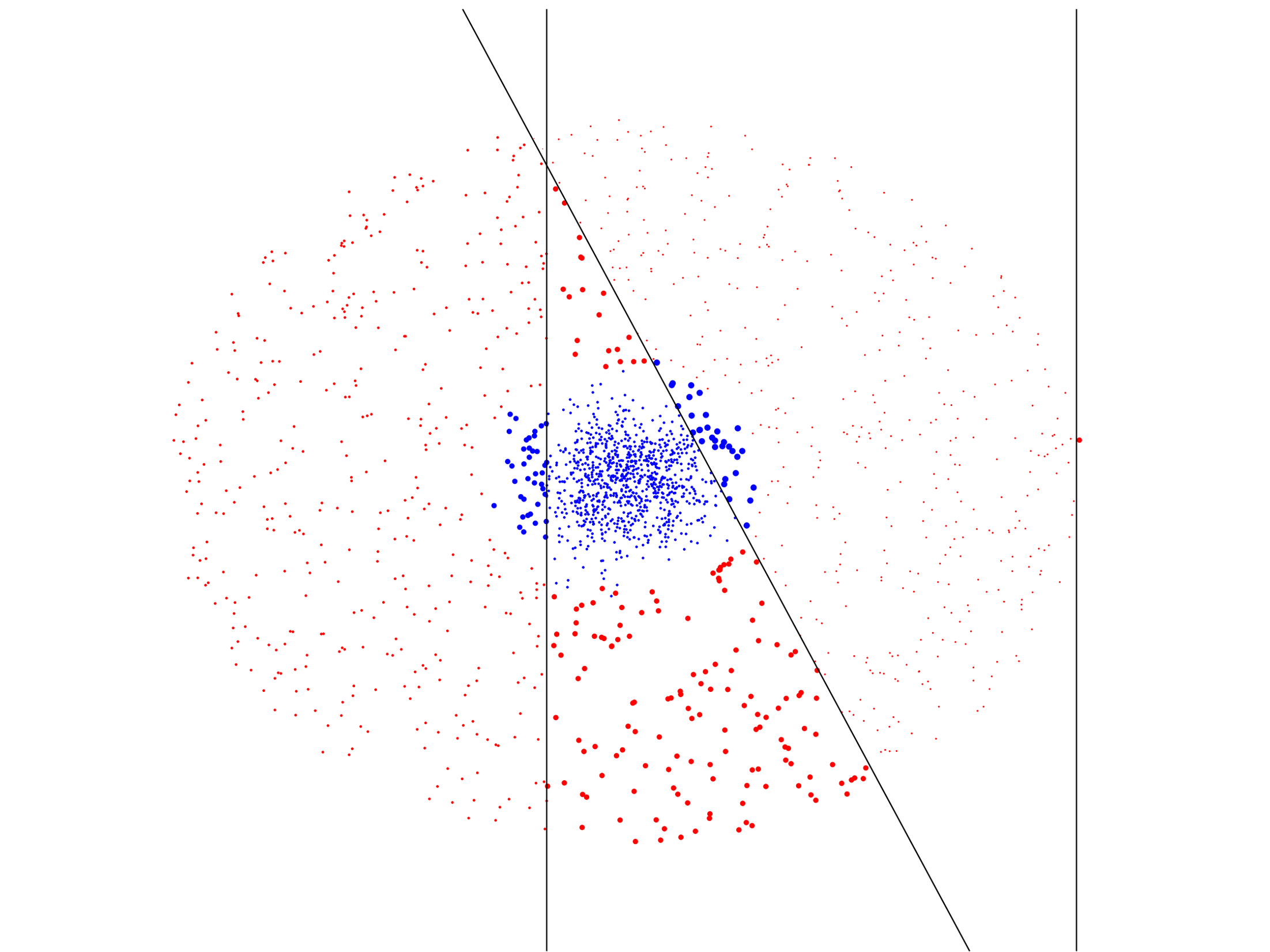

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

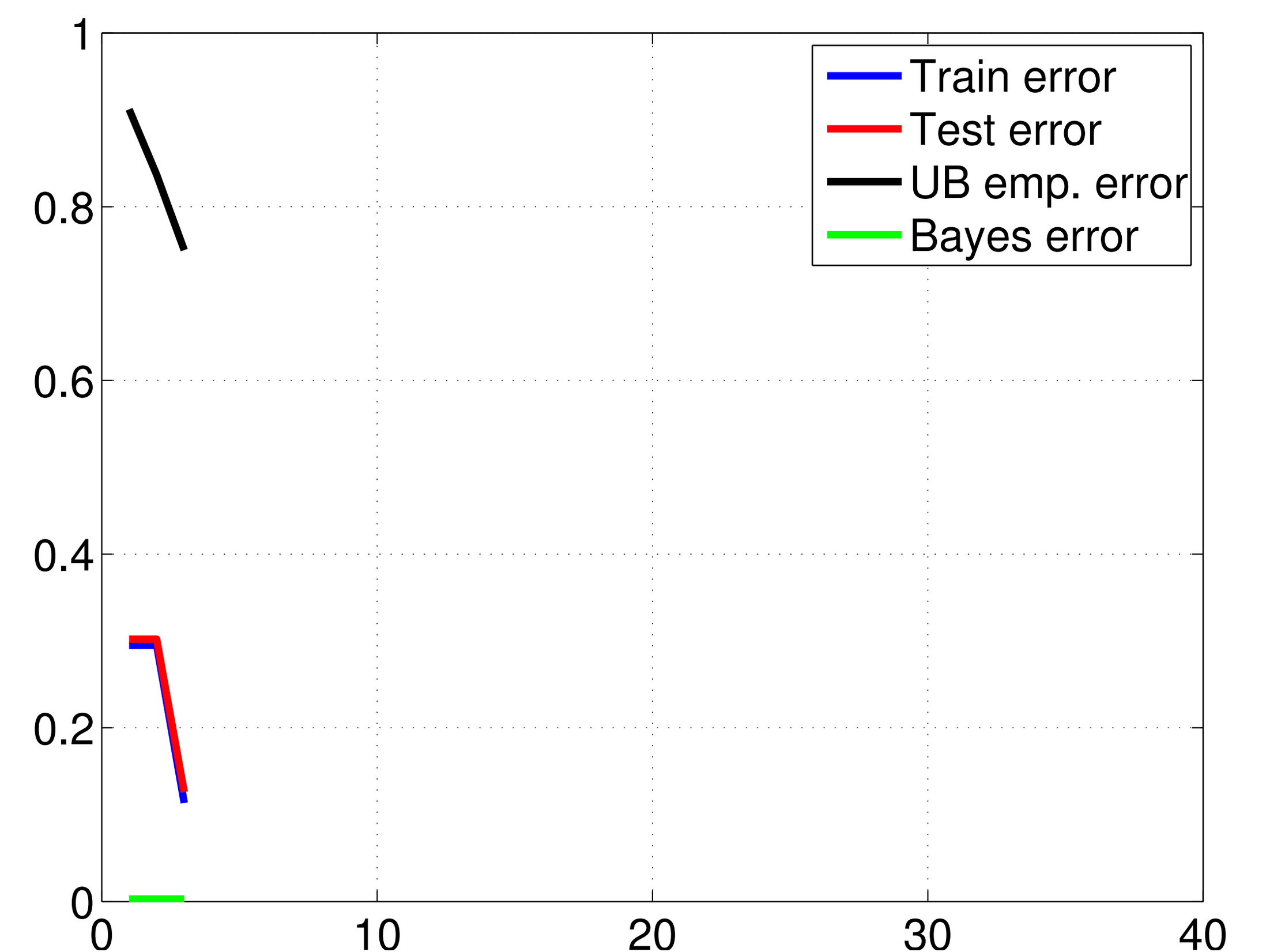


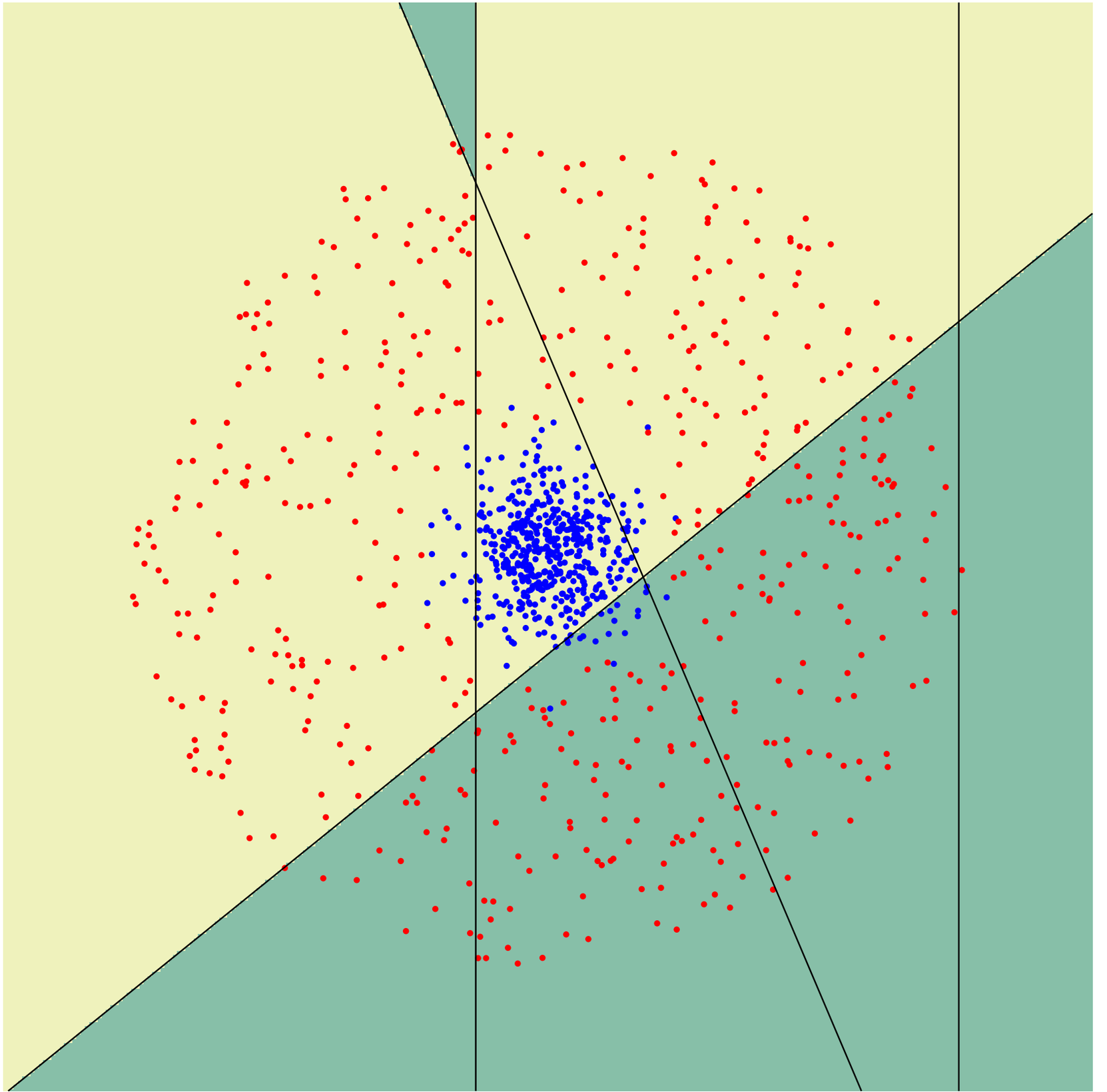


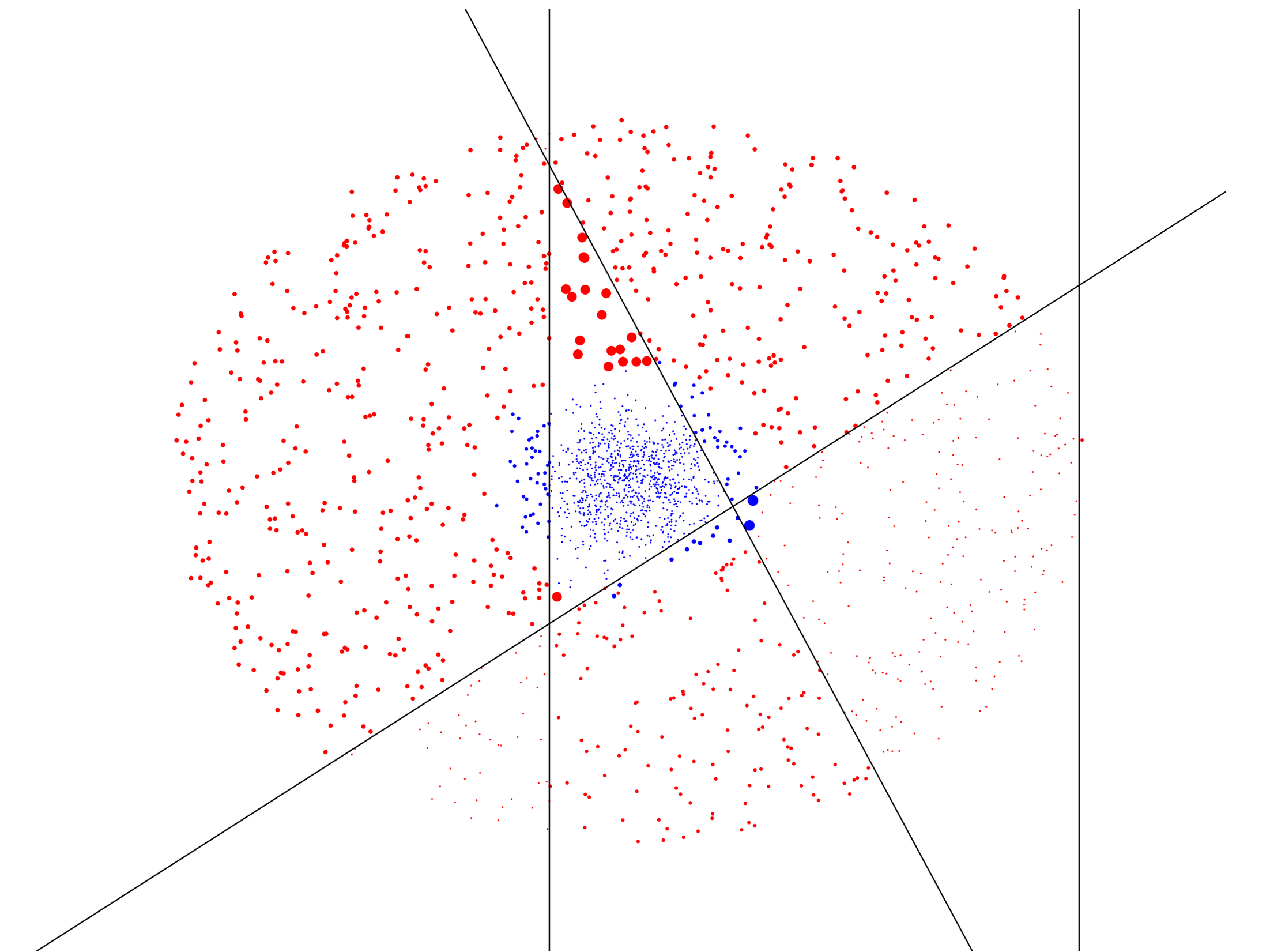

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

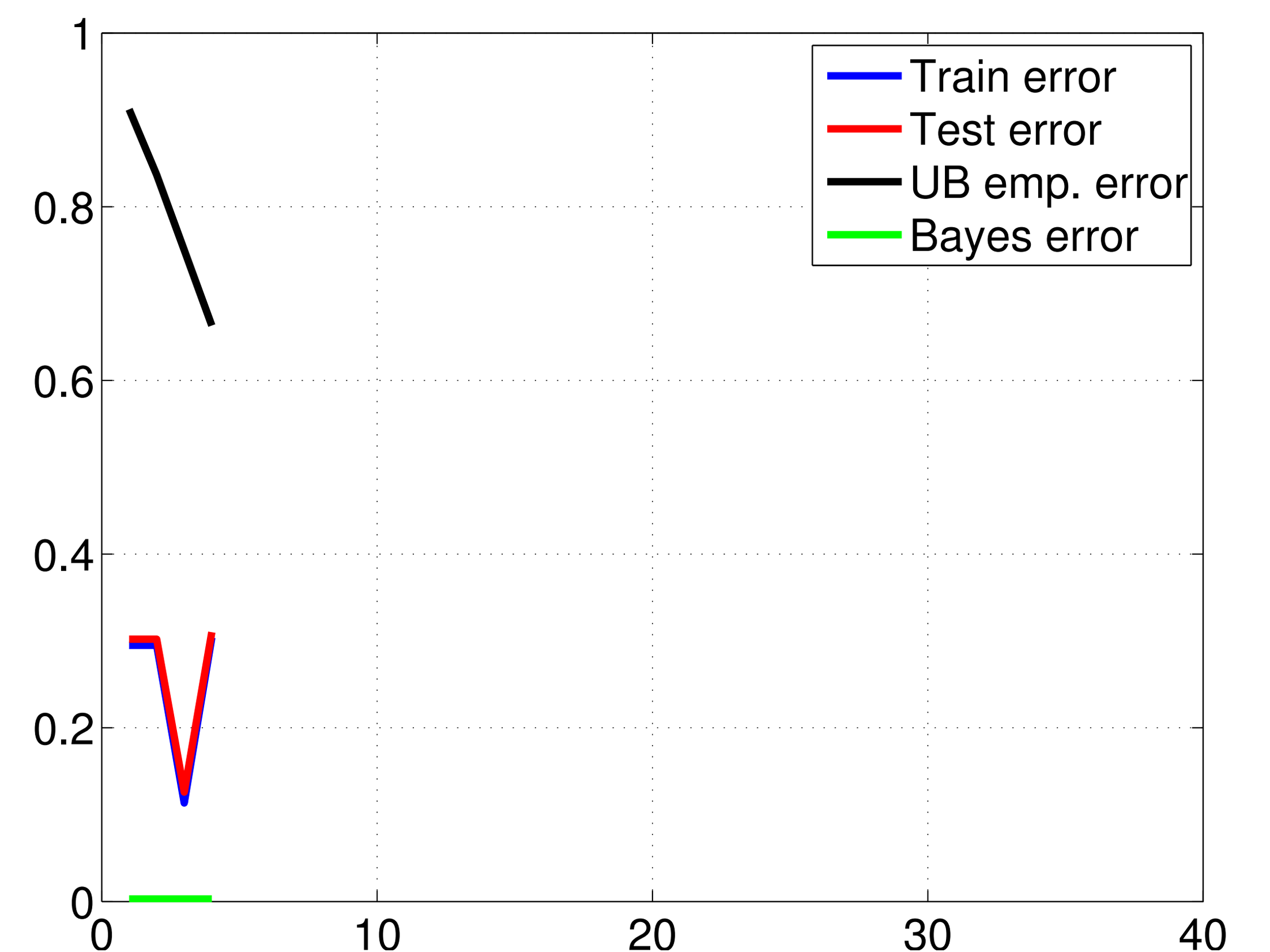


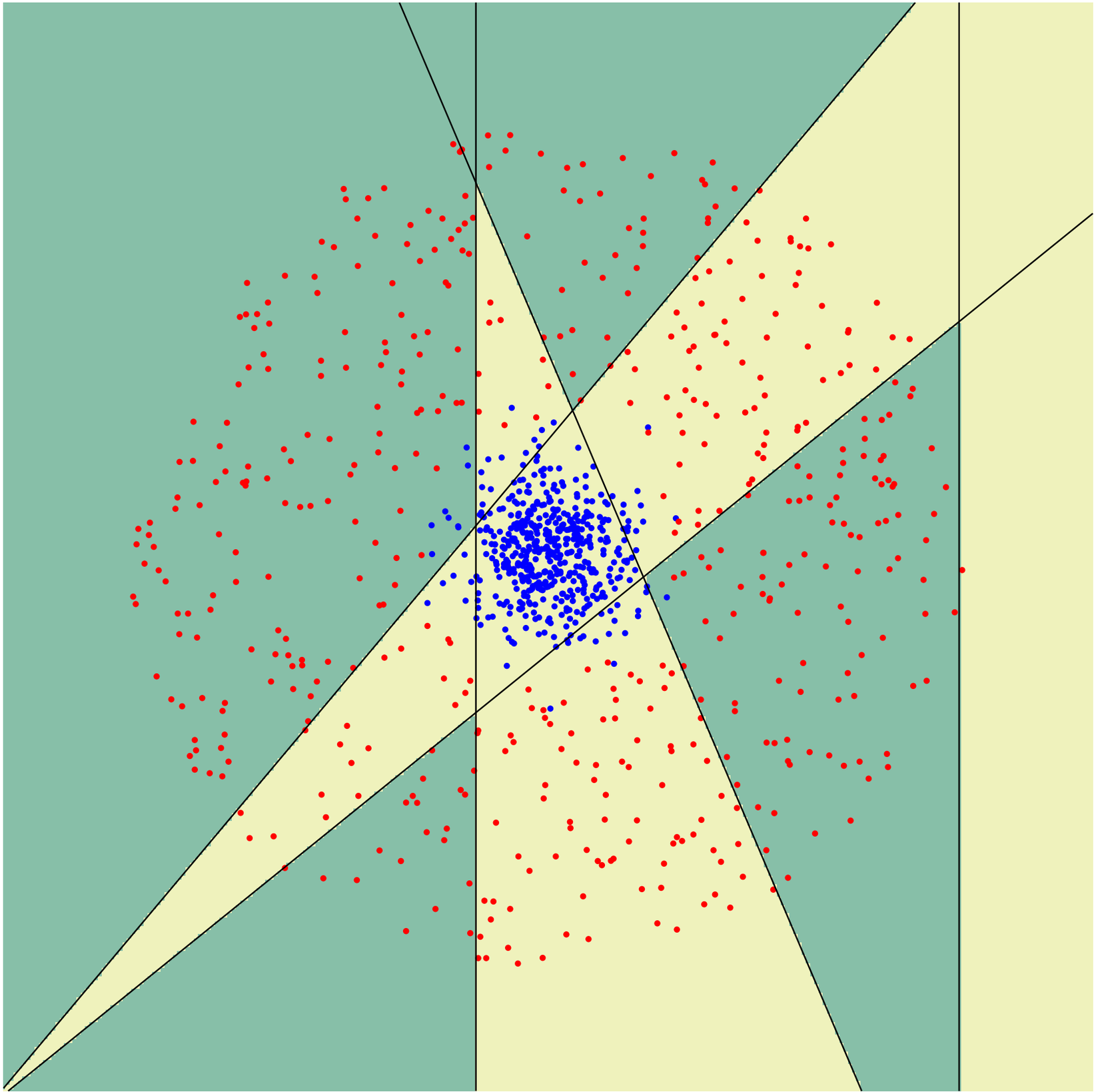


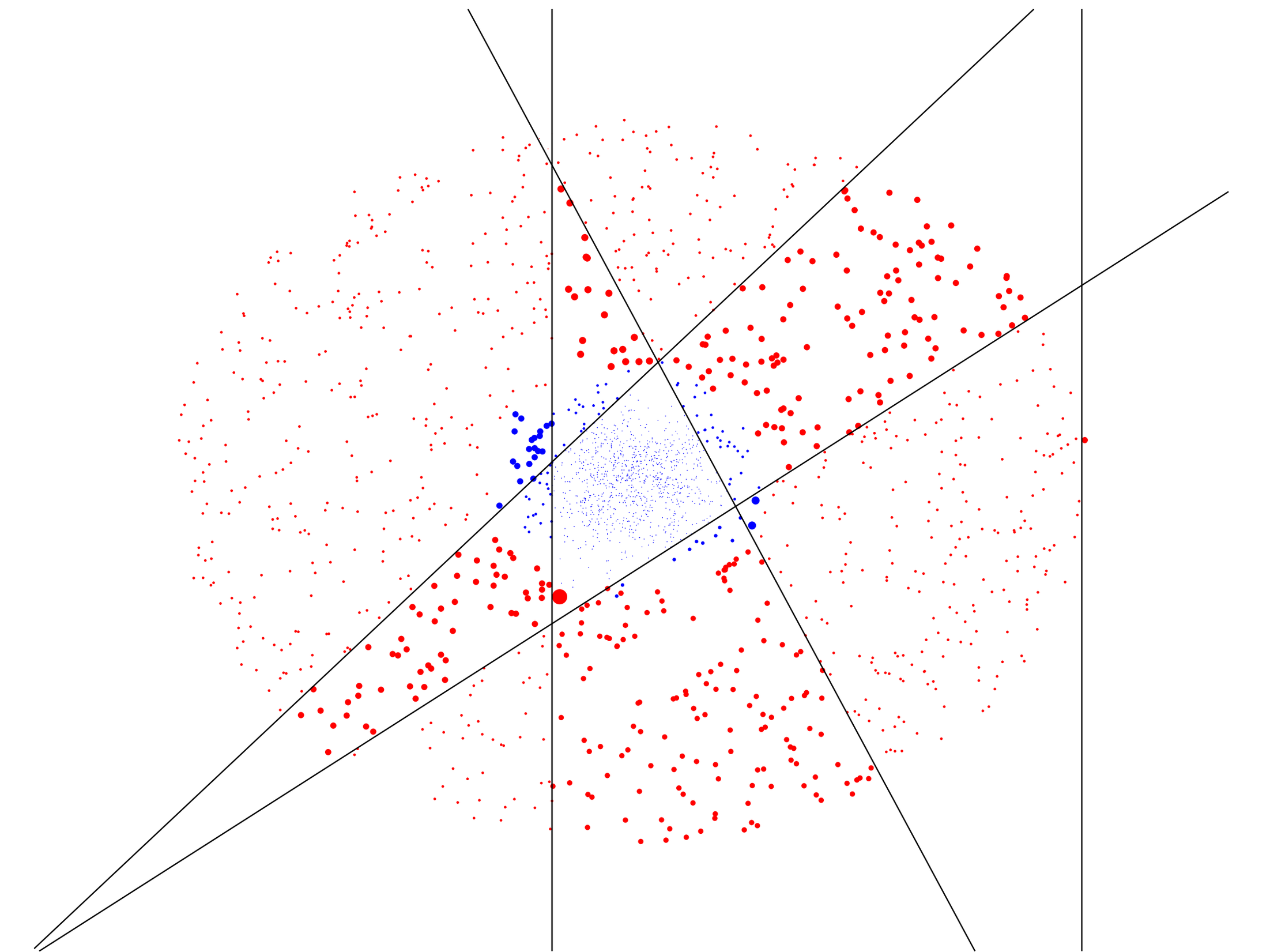

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

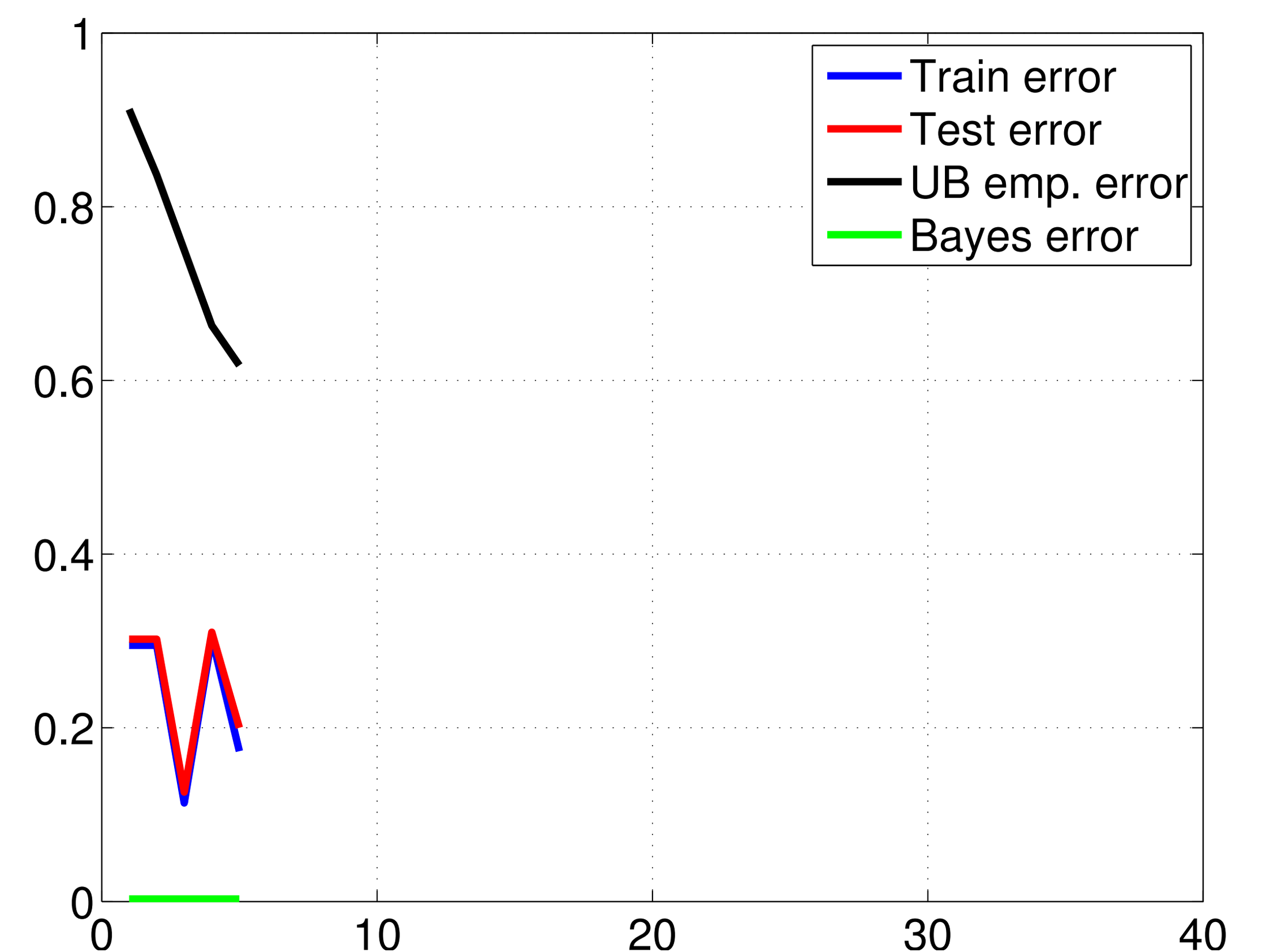


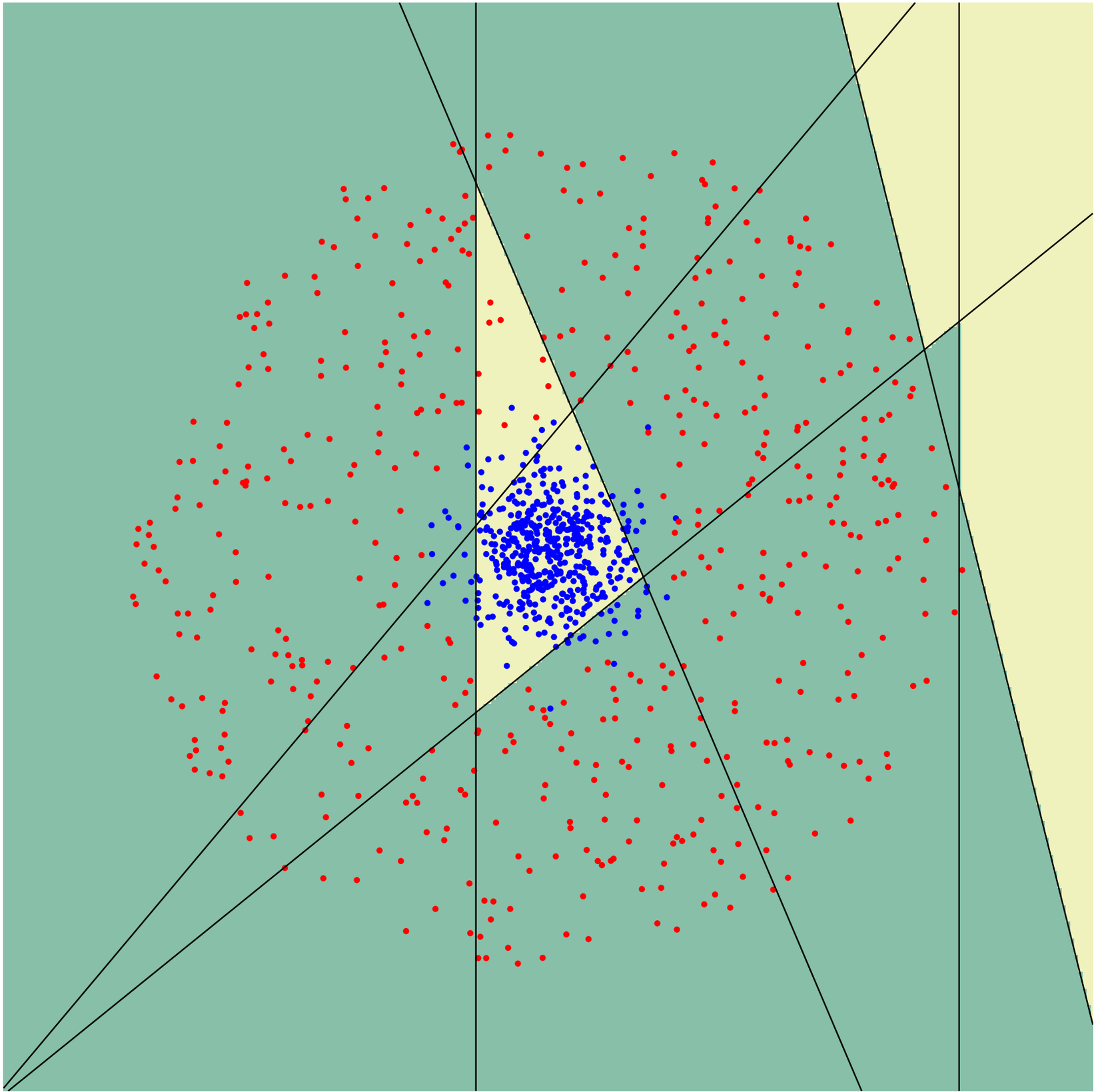


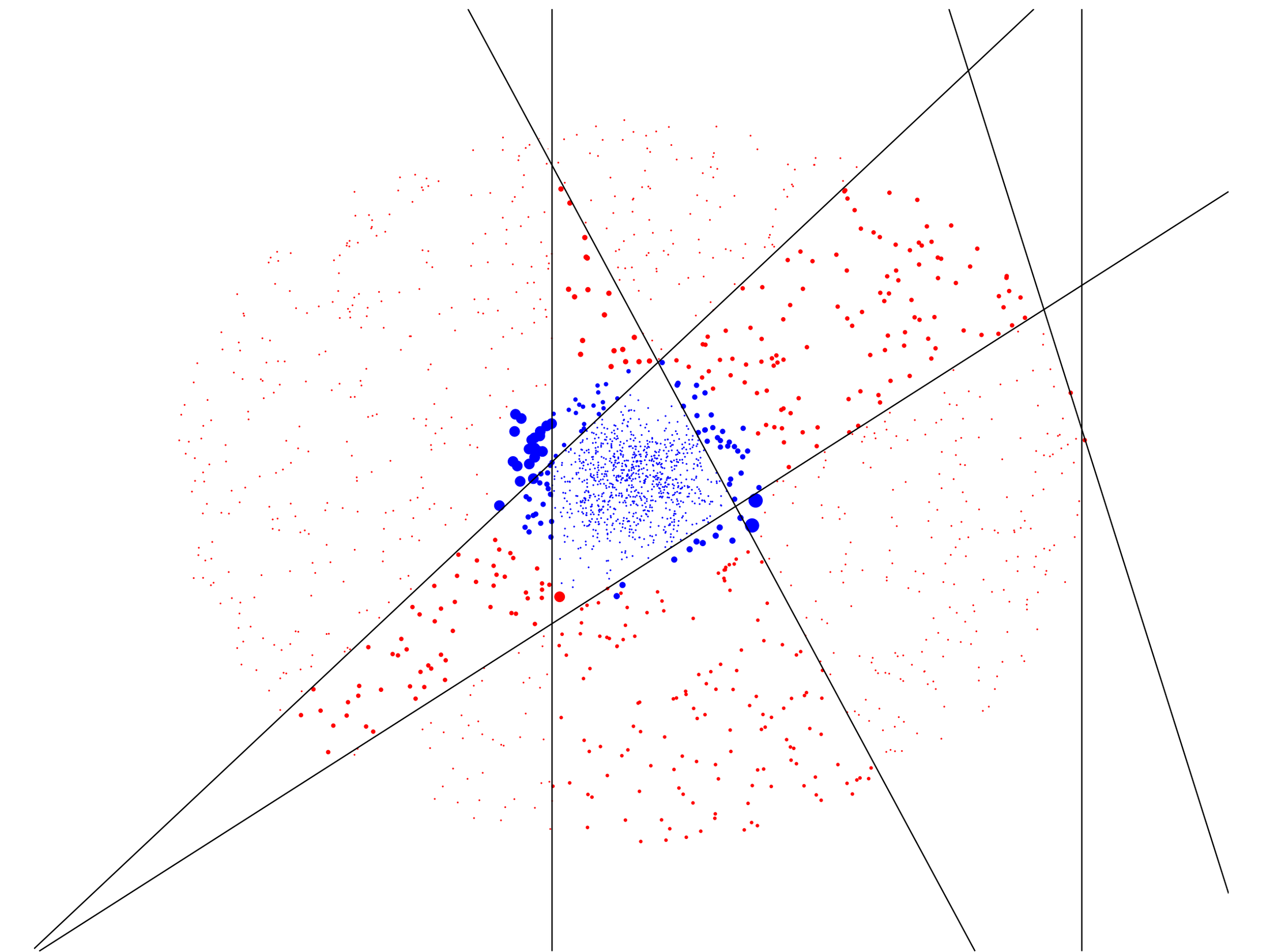

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

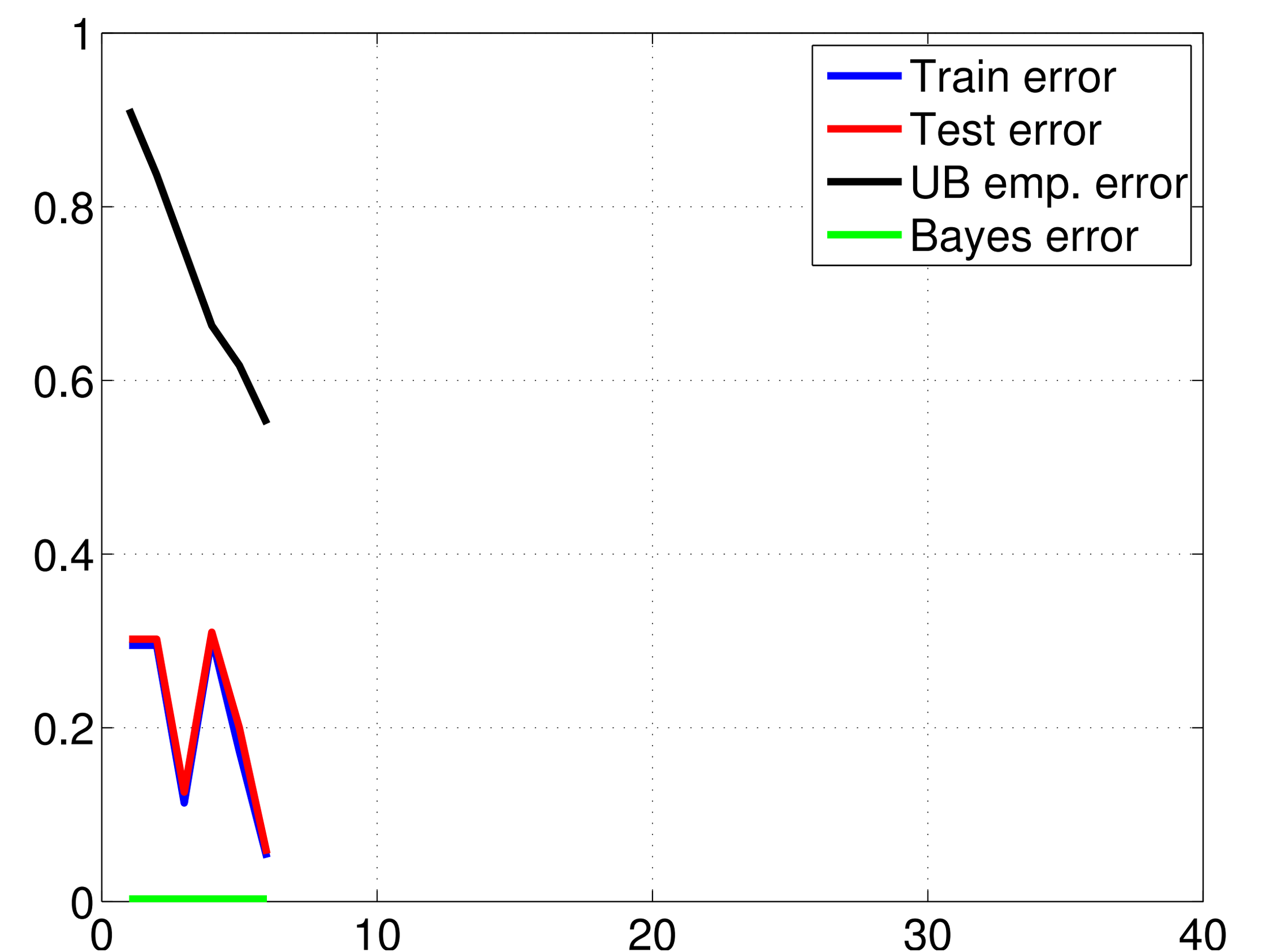


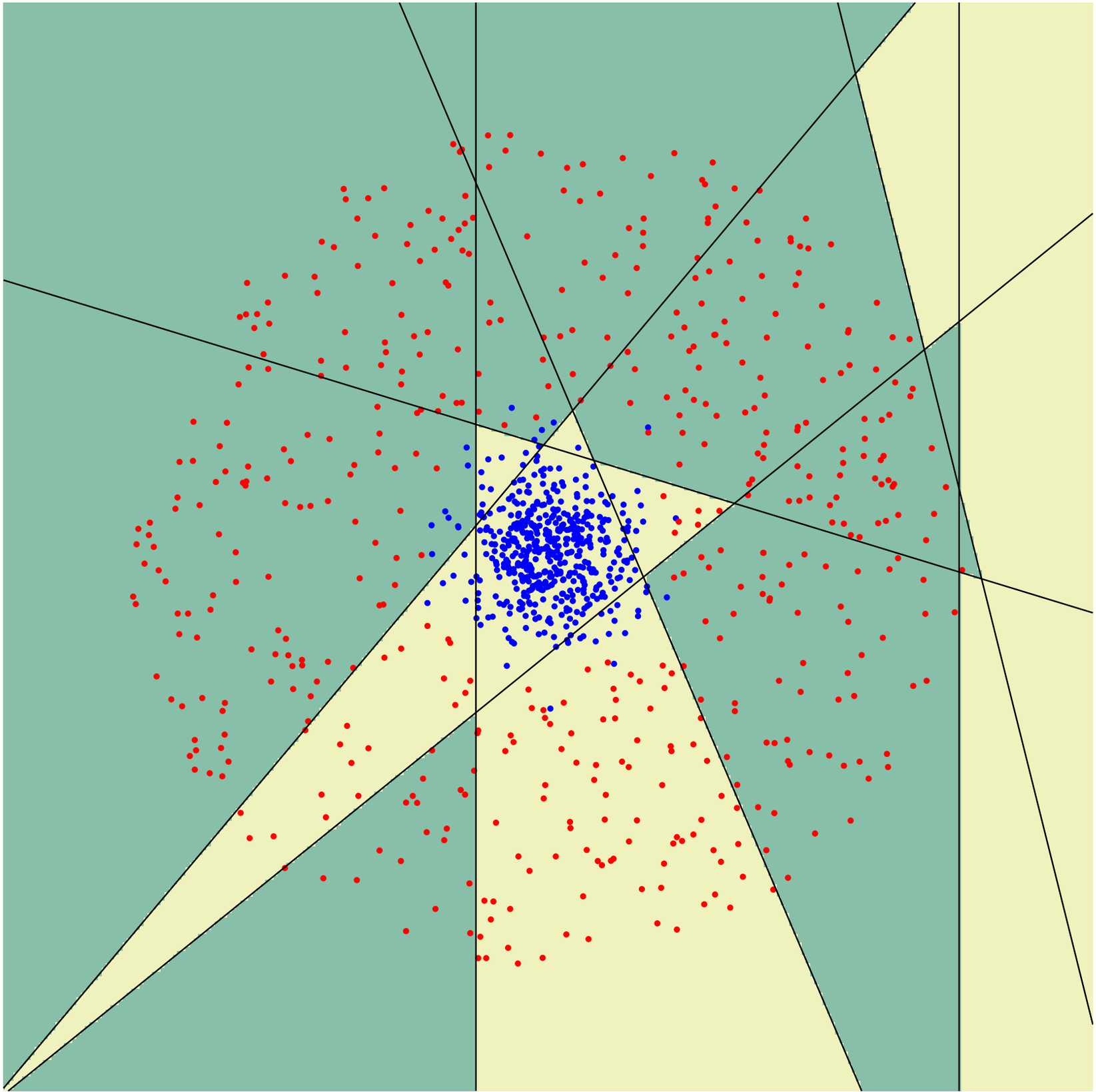


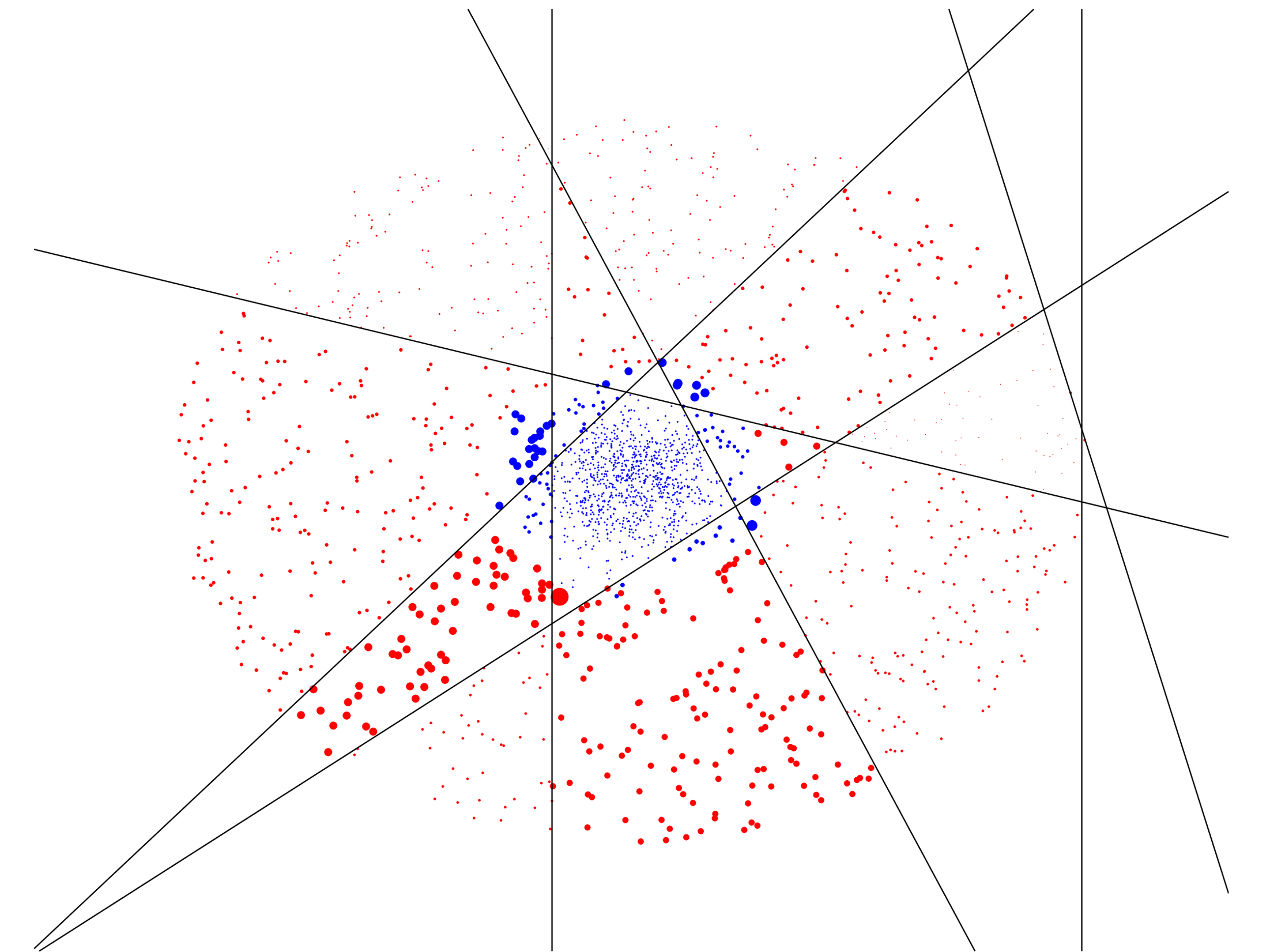

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

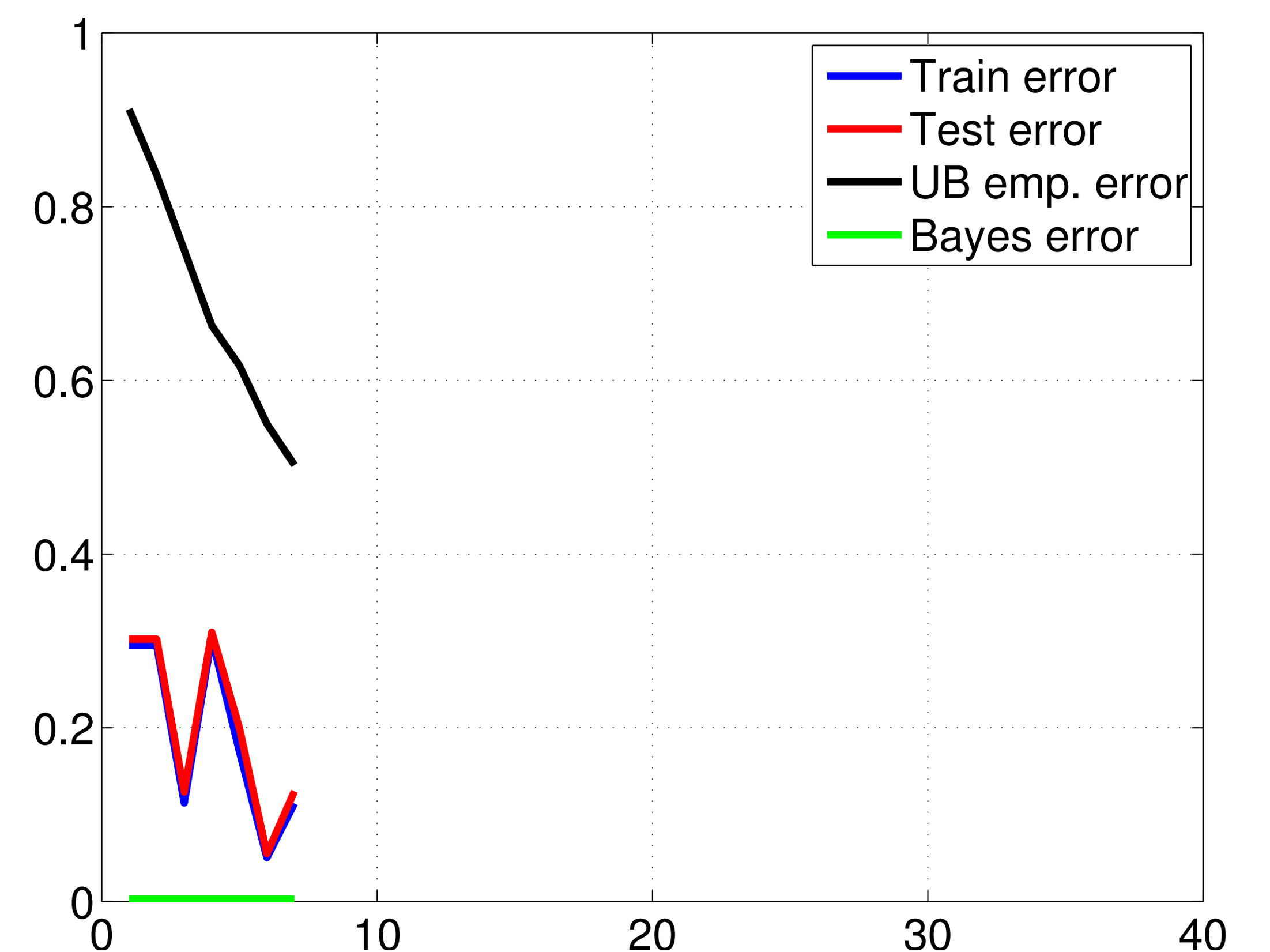


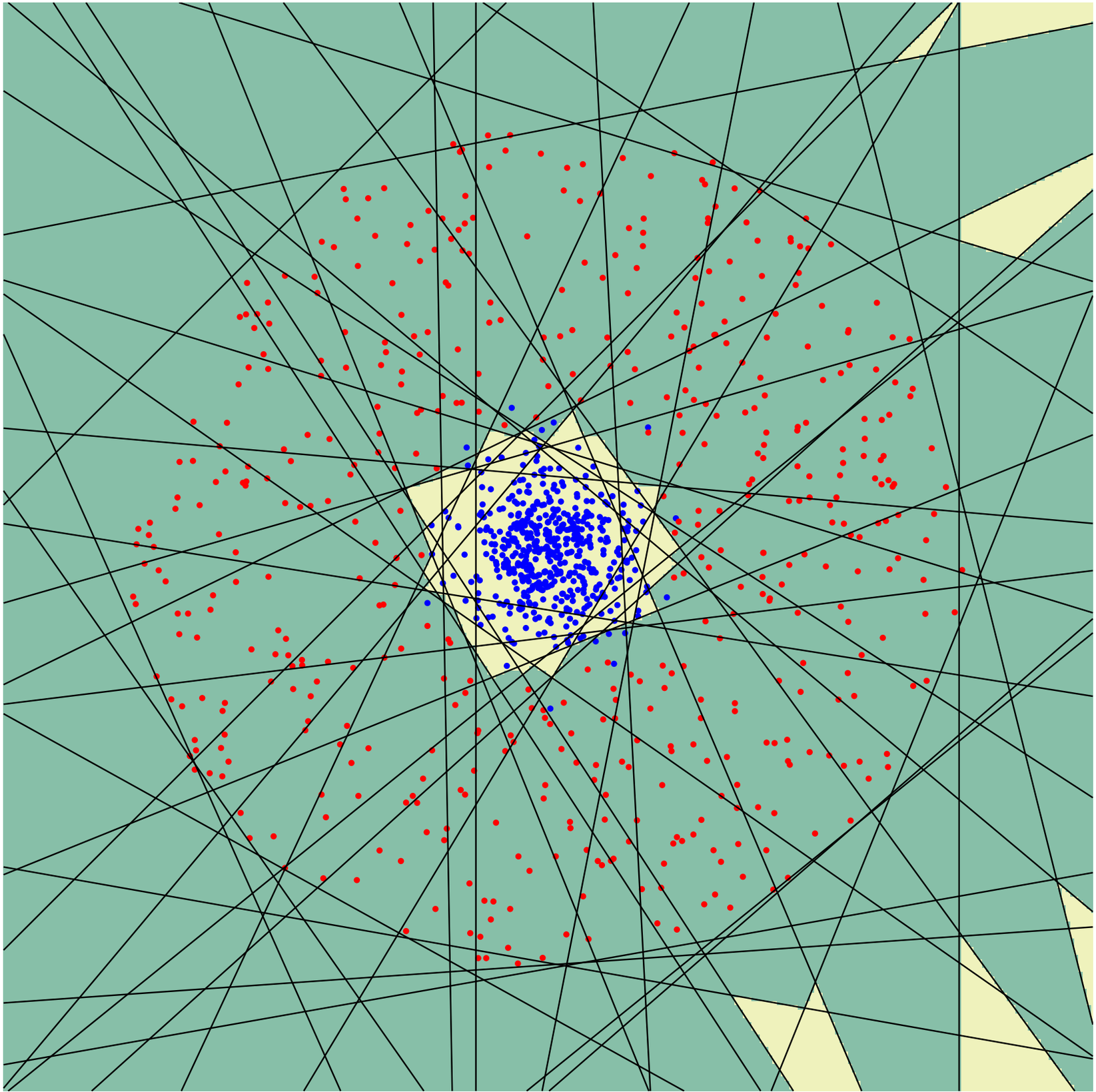


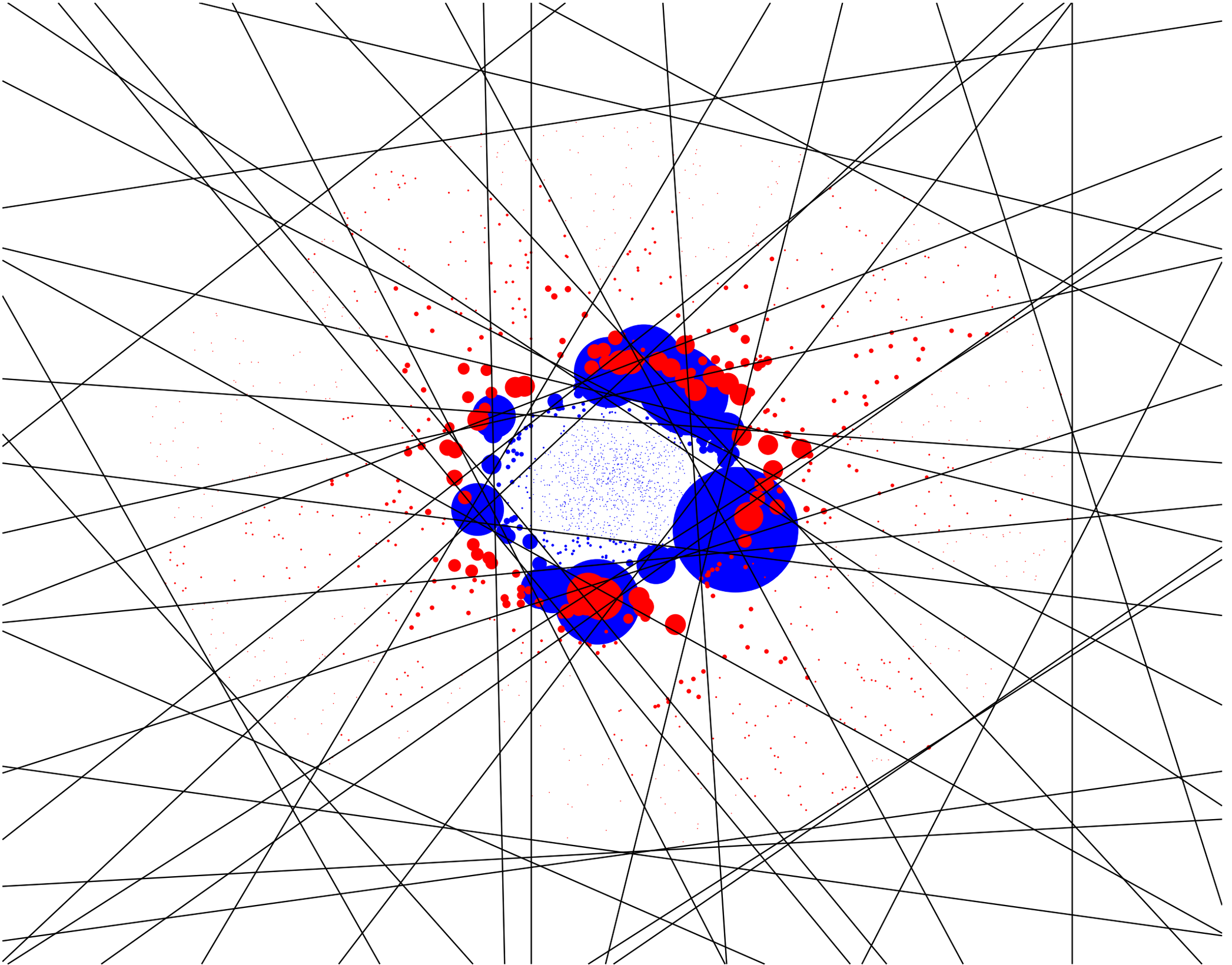

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

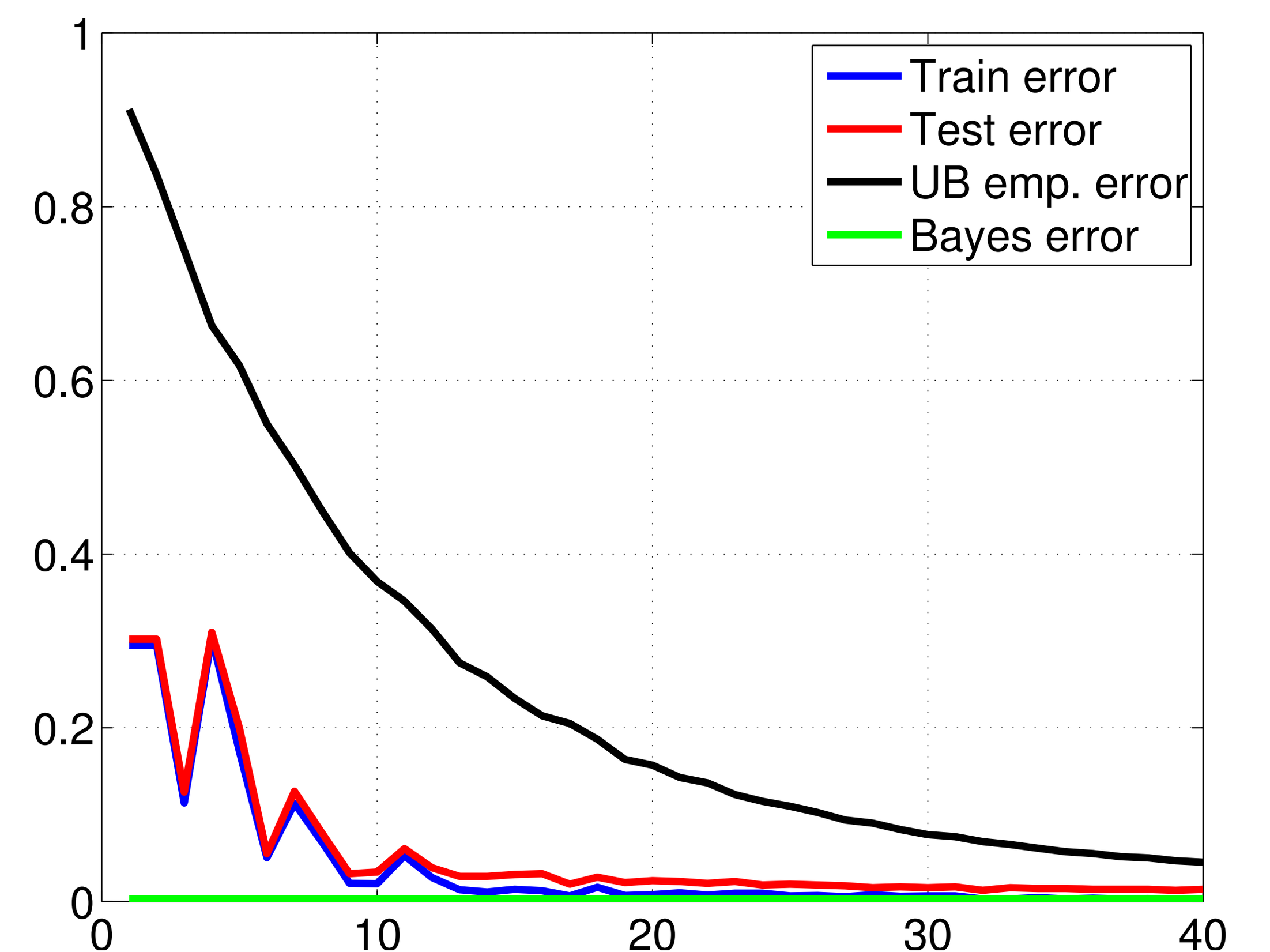


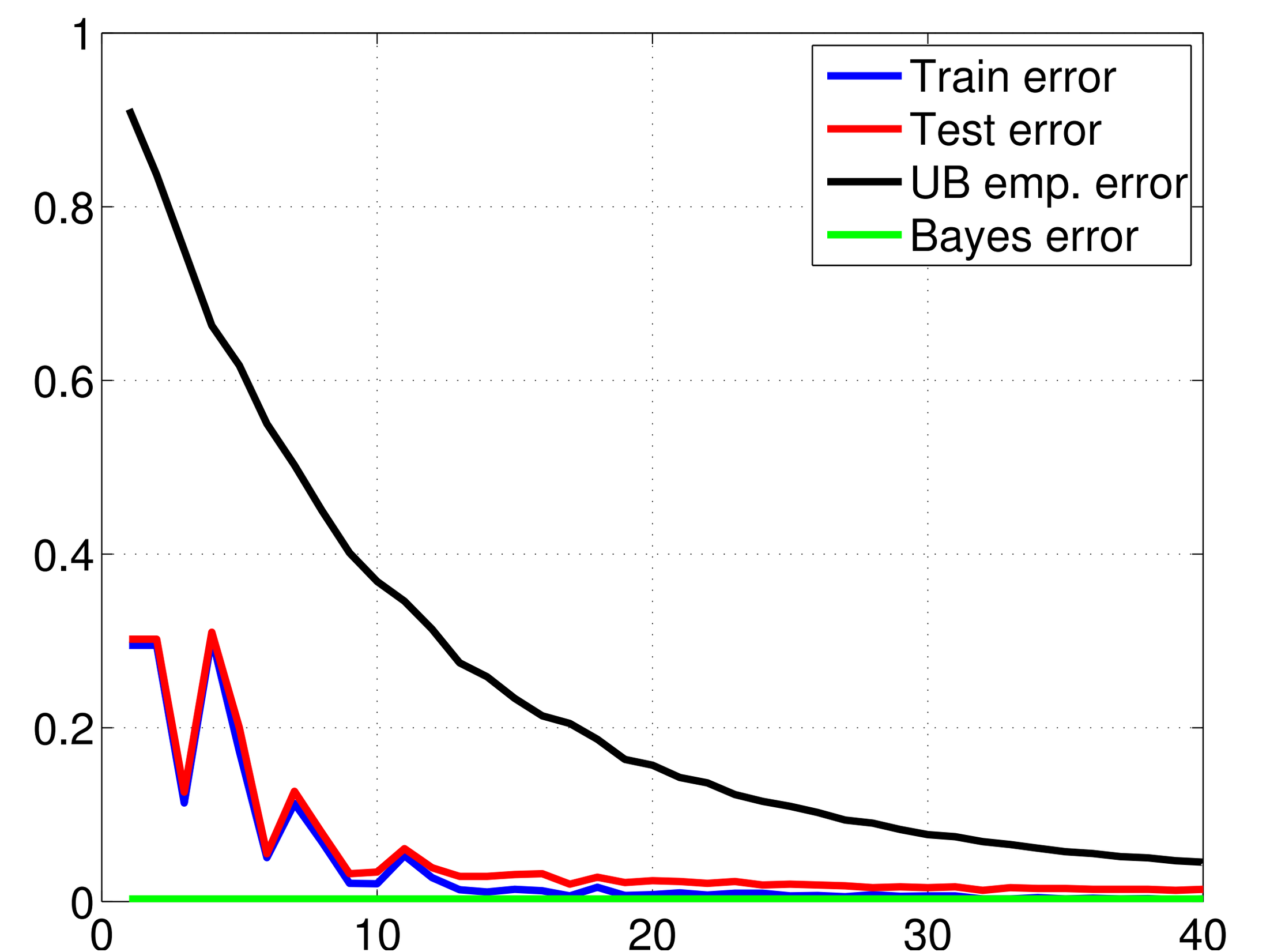


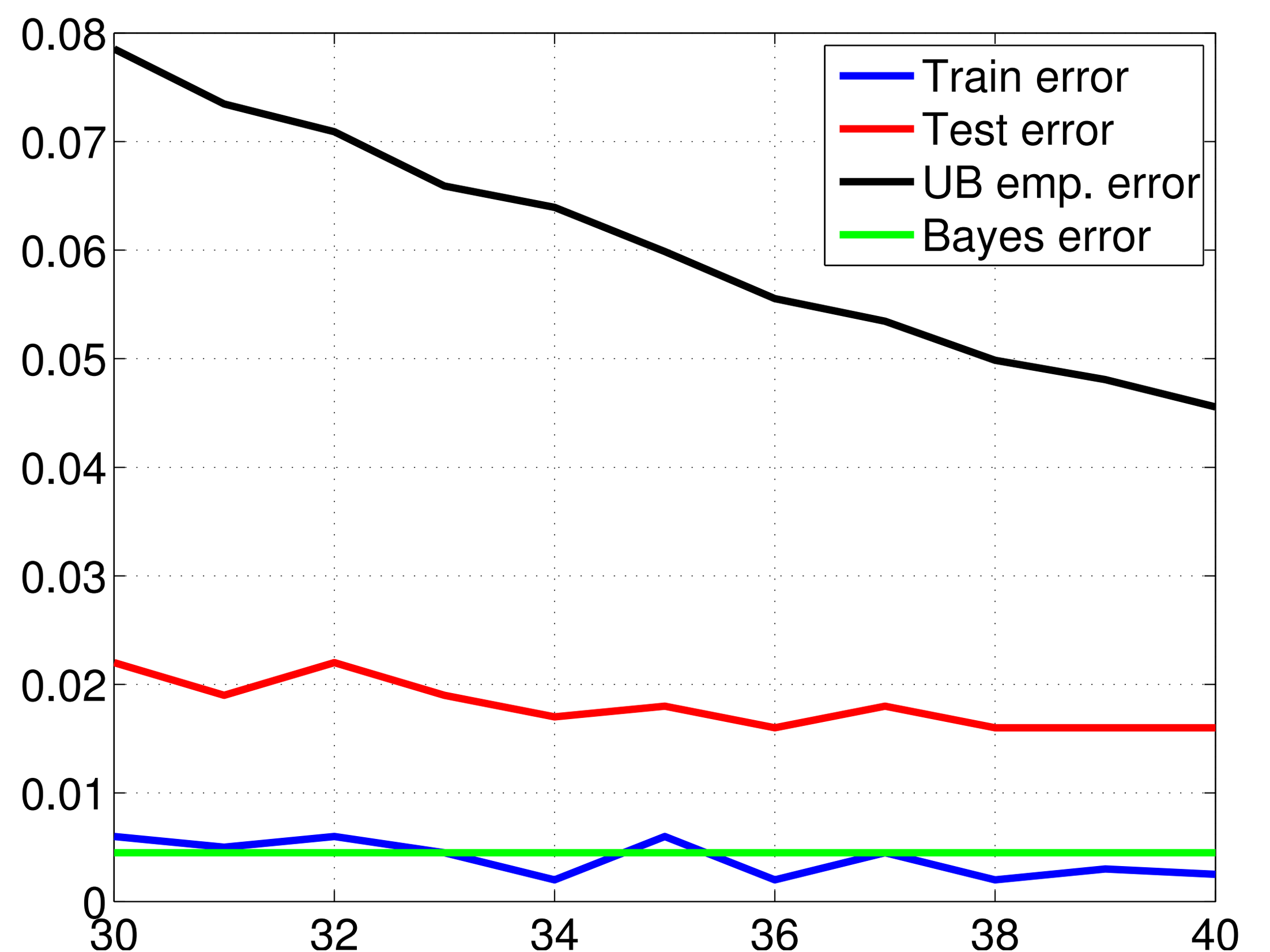

Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40

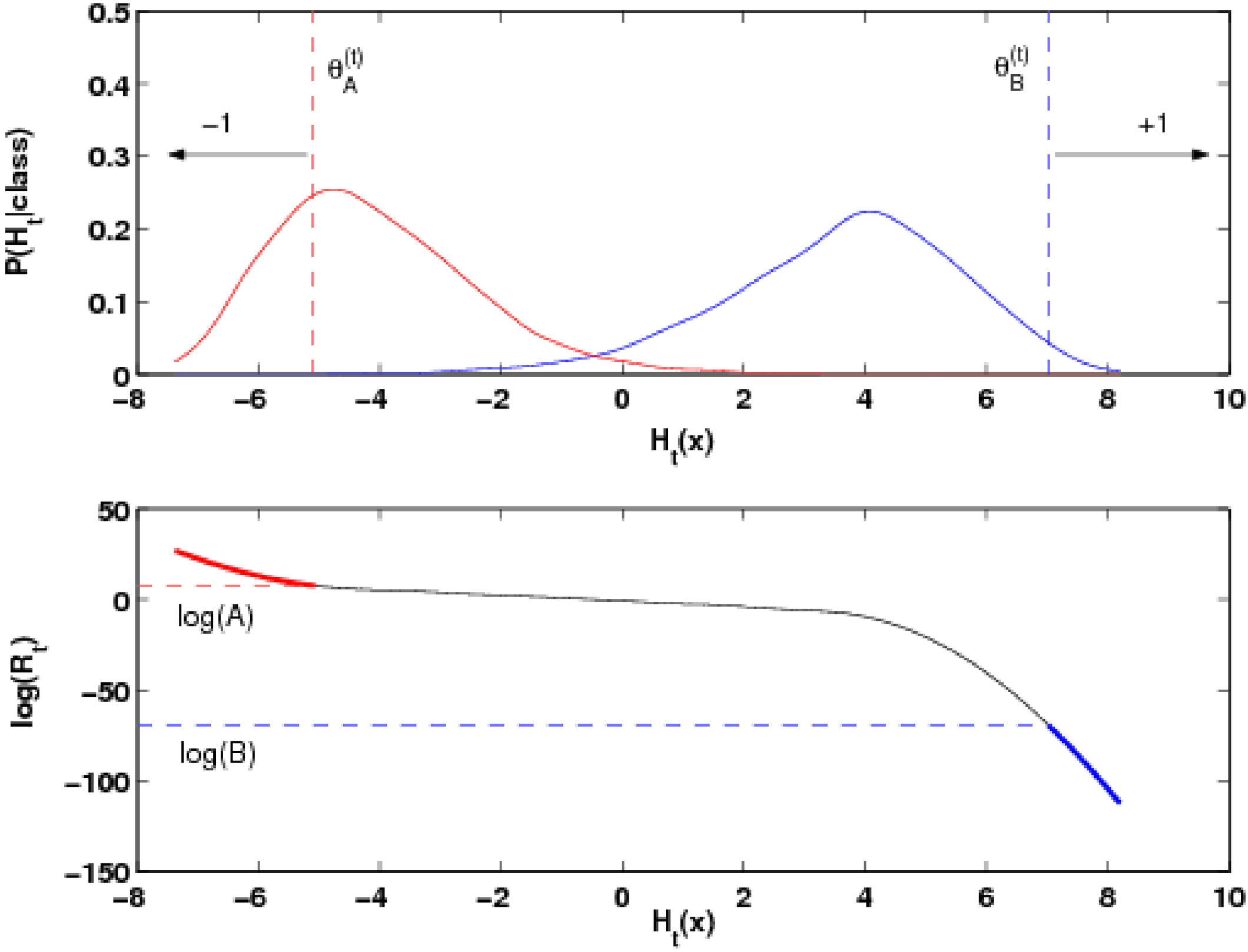


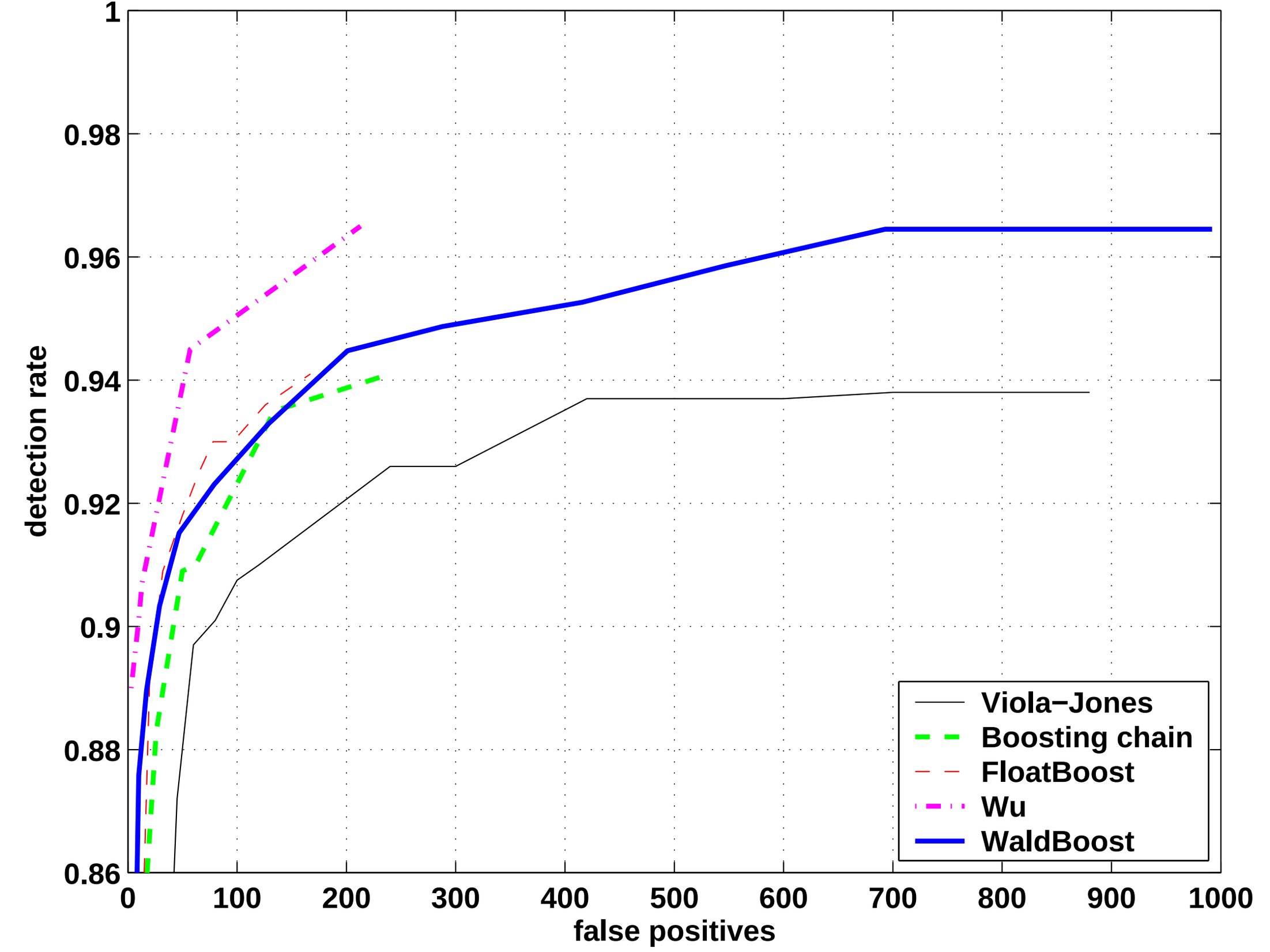



Train error
Test error
UB emp. error
Bayes error
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40


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


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


0.5
0.4
0.3
0.2
0.1
0
P(H_t|class)
θ_A^(t)
θ_B^(t)
−1
+1
−8
−6
−4
−2
0
2
4
6
8
10
H_t(x)
50
0
−50
−100
−150
log(R_t)
log(A)
log(B)
−8
−6
−4
−2
0
2
4
6
8
10
H_t(x)


1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
detection rate
0
100
200
300
400
500
600
700
800
900
1000
false positives
Viola−Jones
Boosting chain
FloatBoost
Wu
WaldBoost

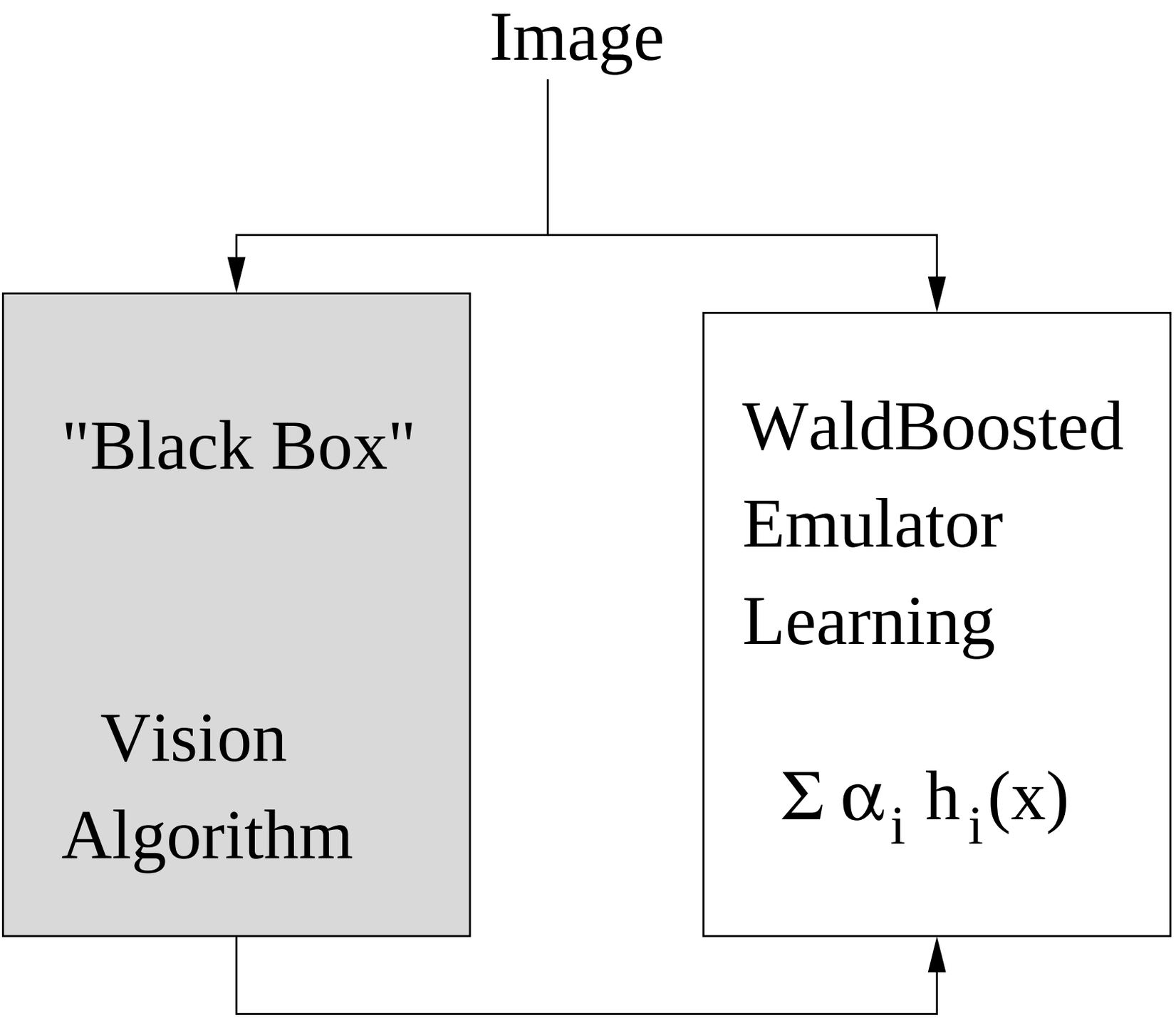
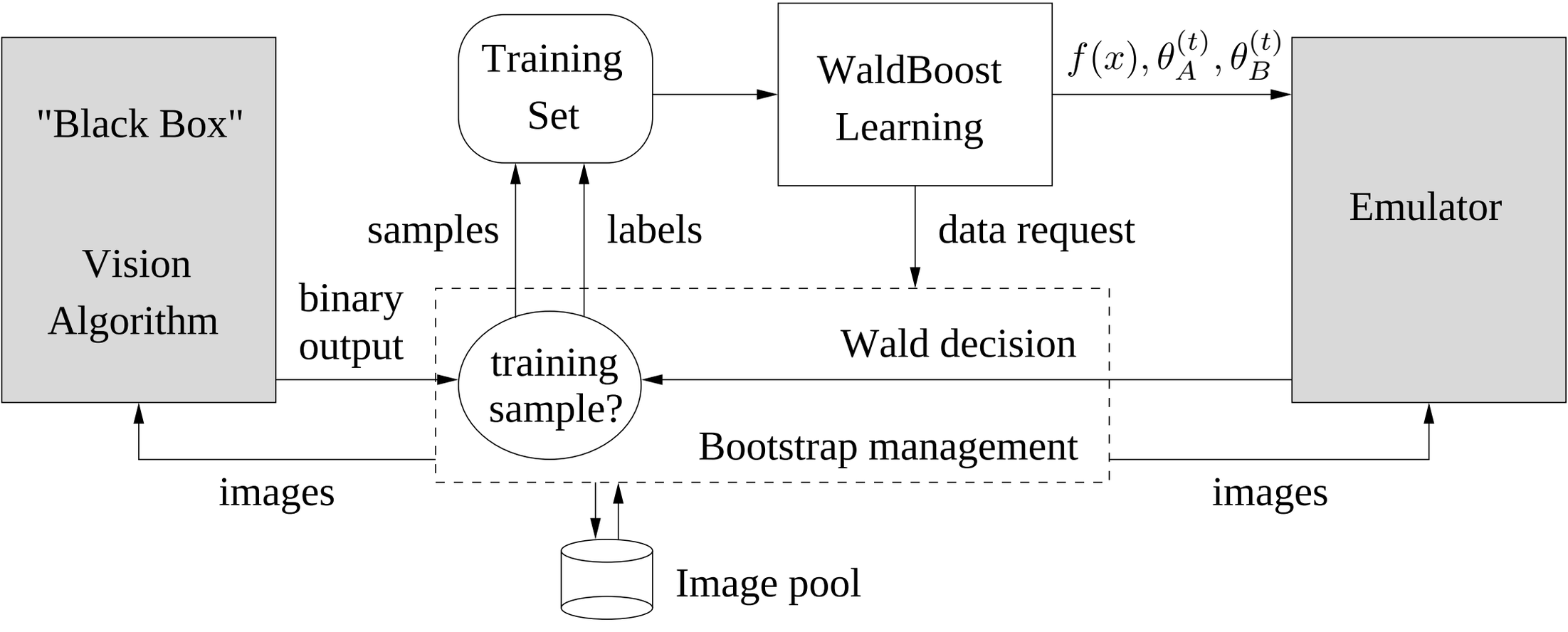
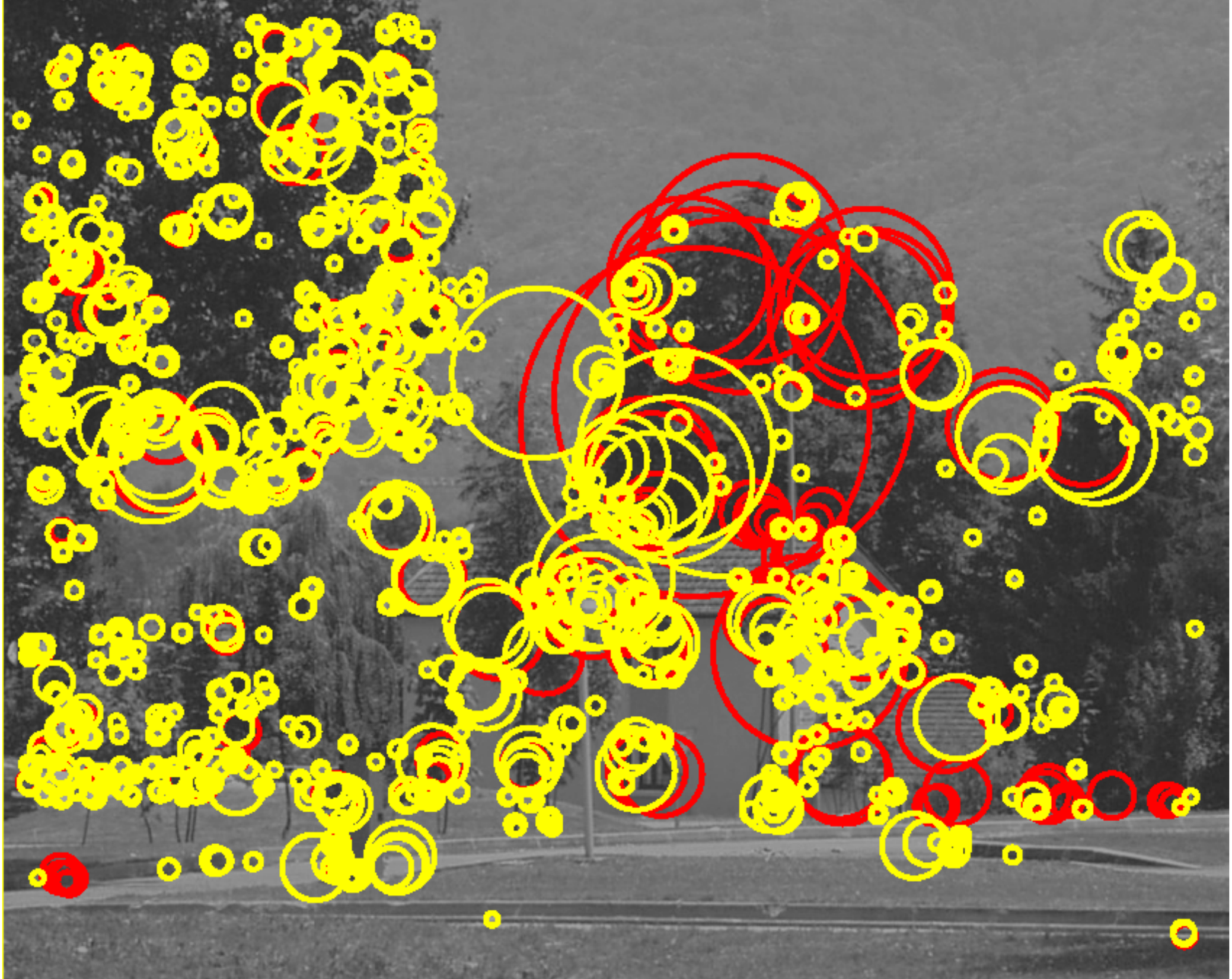
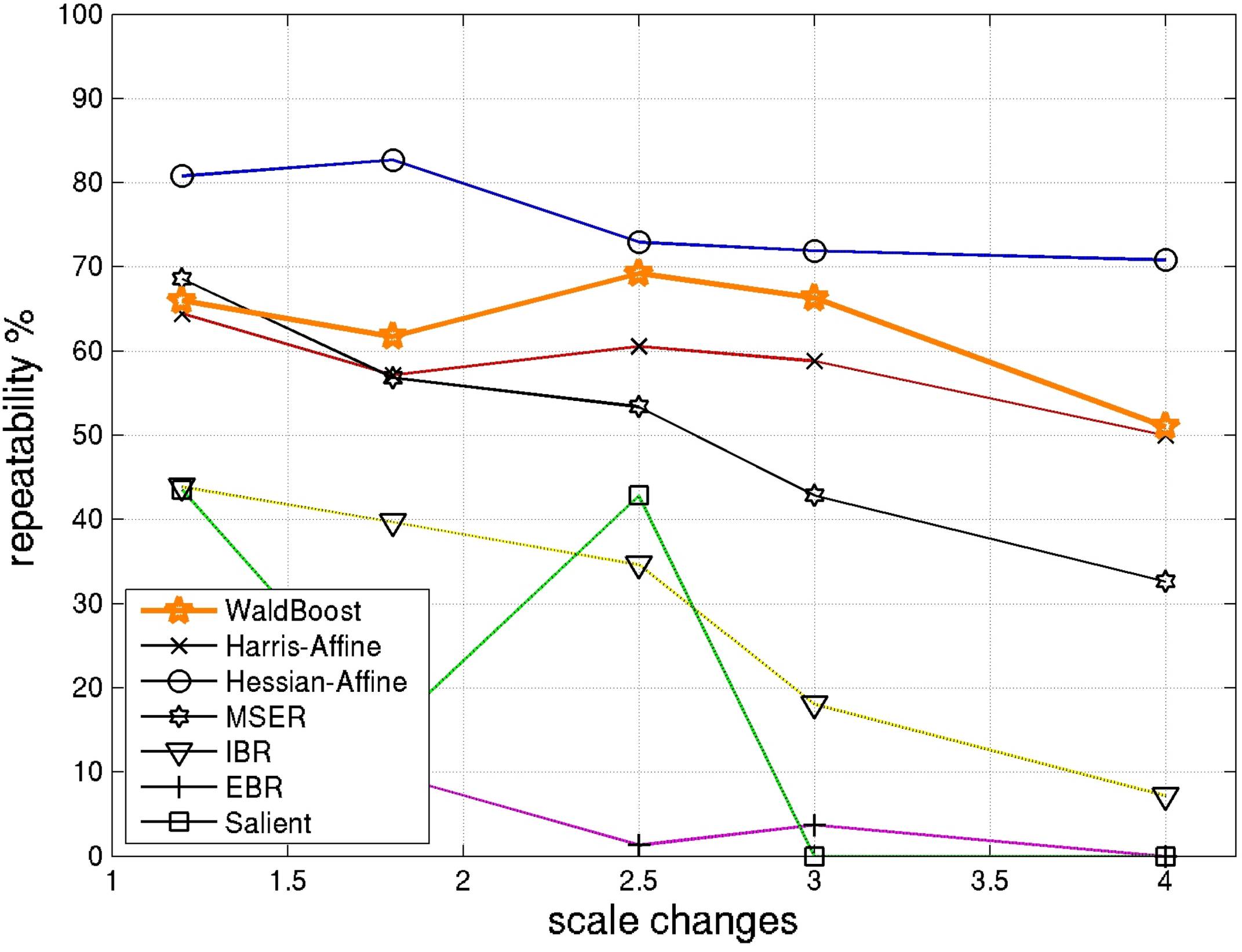
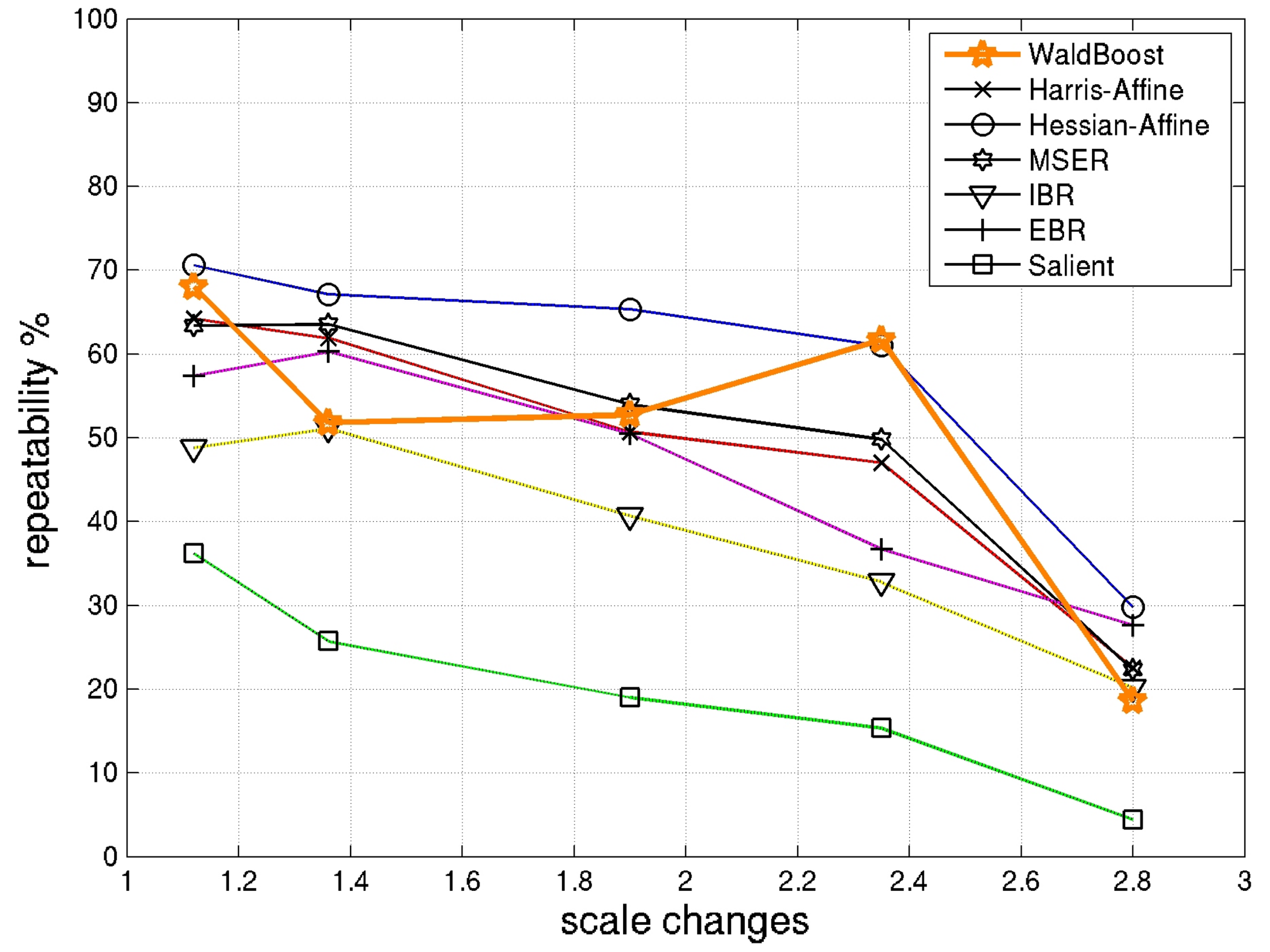


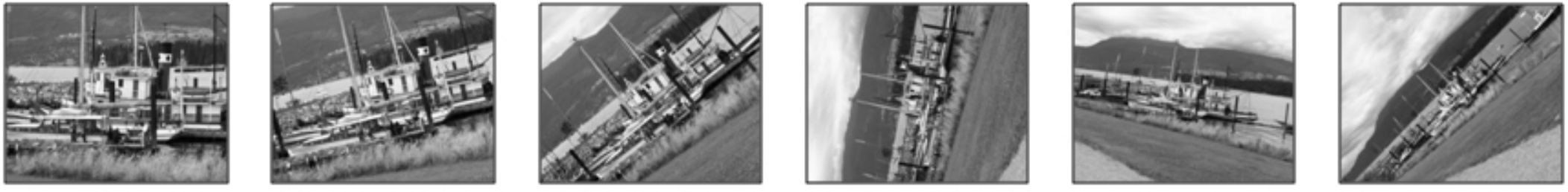
Image
"Black Box"
Vision
Algorithm
WaldBoosted
Emulator
Learning
$\Sigma \alpha_i h_i(x)$

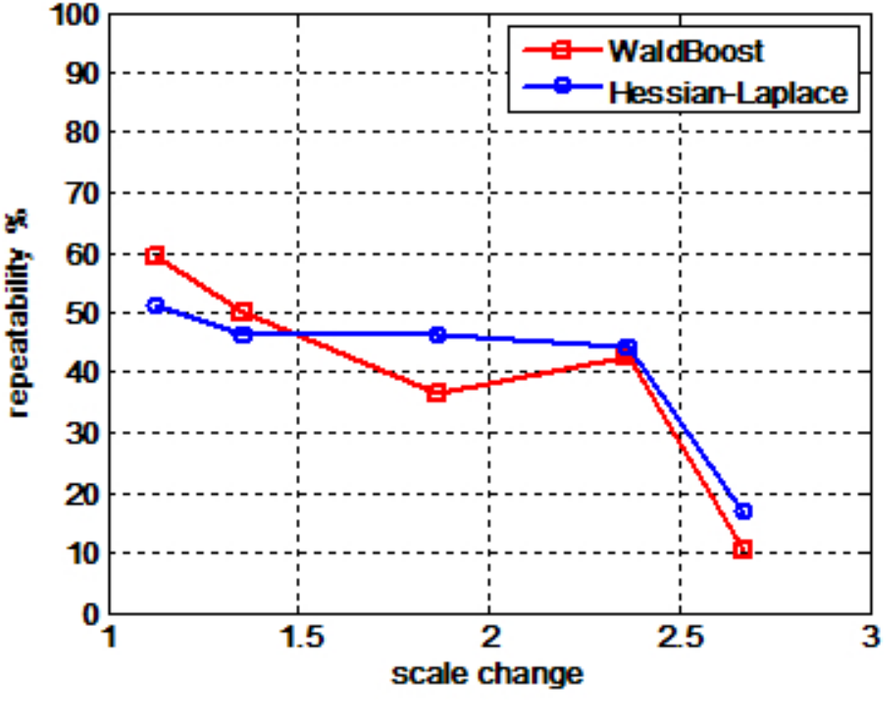

"Black Box"
Vision
Algorithm
Training
Set
WaldBoost
Learning
$f(x), \theta_A^{(t)}, \theta_B^{(t)}$
Emulator
samples
labels
data request
binary
output
Wald decision
training
sample?
Bootstrap management
images
images
Image pool

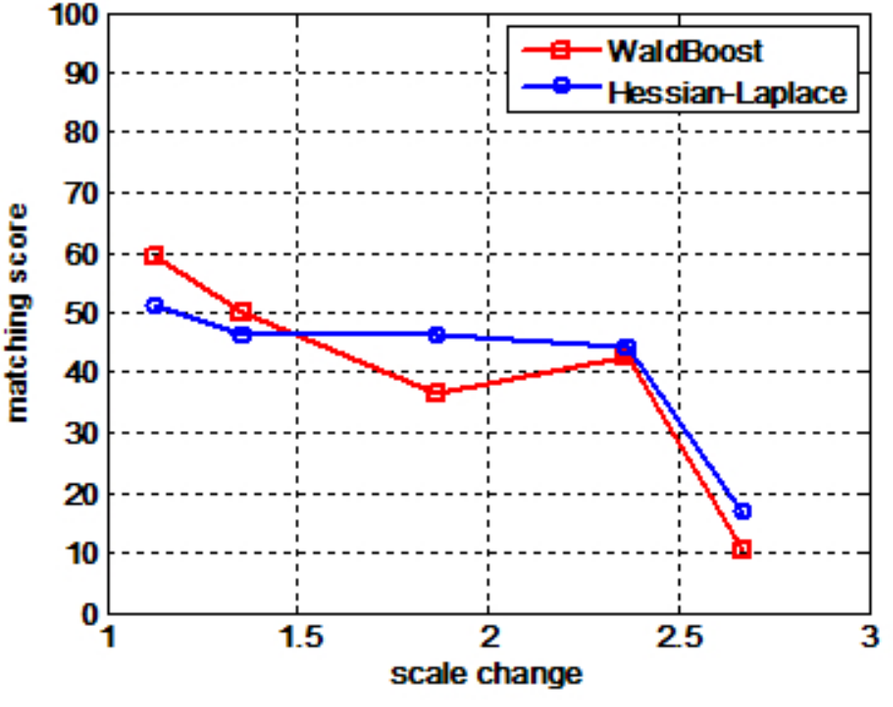


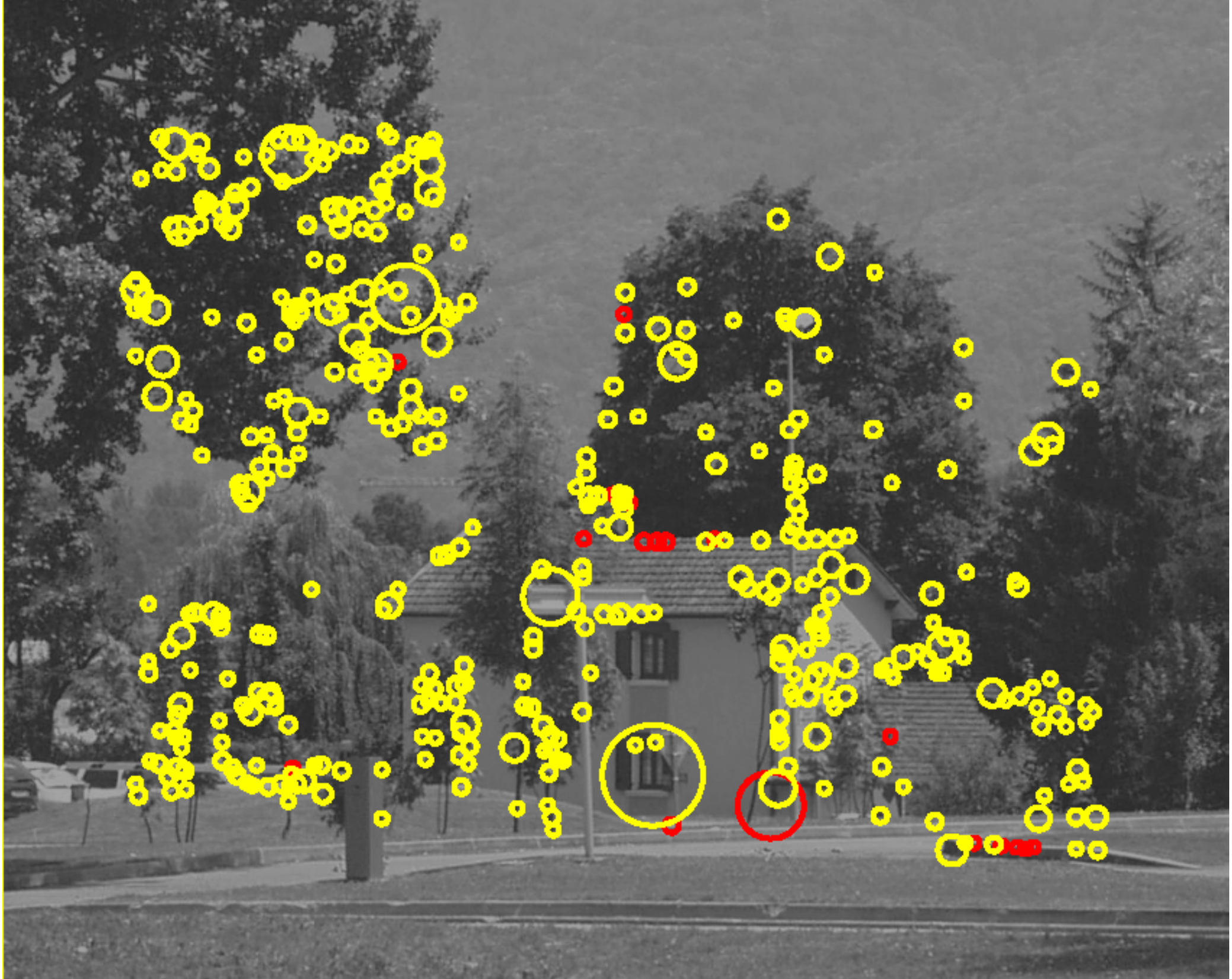

repeatability %
scale changes
WaldBoost
Harris-Affine
Hessian-Affine
MSER
IBR
EBR
Salient
100
90
80
70
60
50
40
30
20
10
0
1
1.5
2
2.5
3
3.5
4

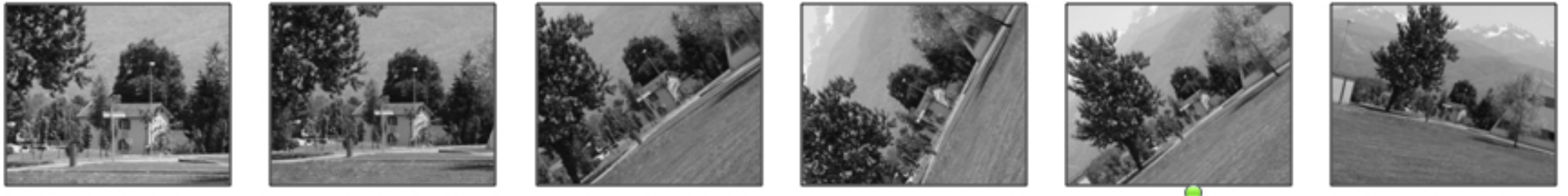

WaldBoost
Harris-Affine
Hessian-Affine
MSER
IBR
EBR
Salient
repeatability %
scale changes
100
90
80
70
60
50
40
30
20
10
0
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3

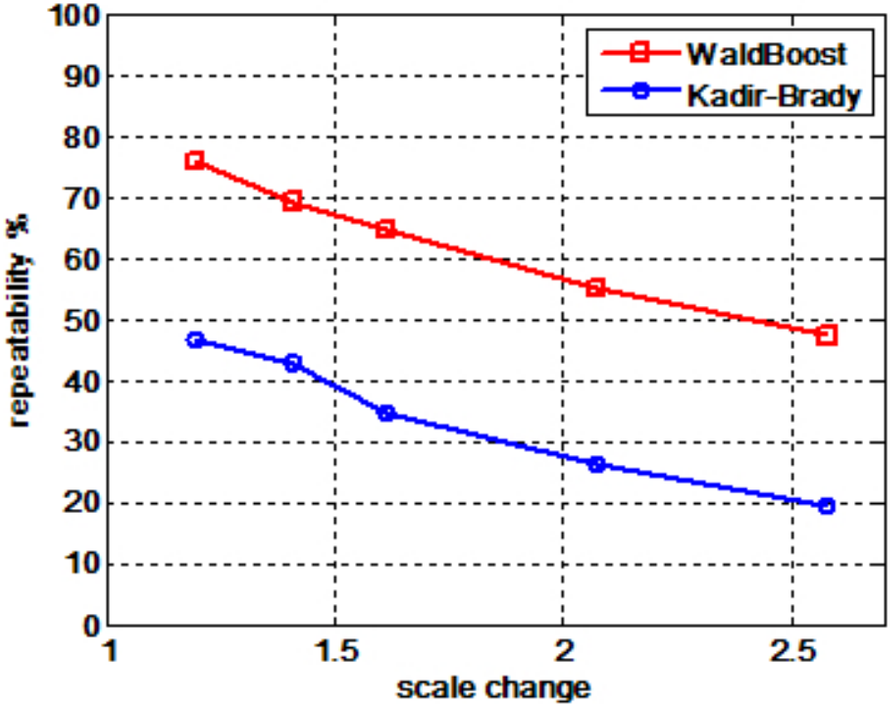


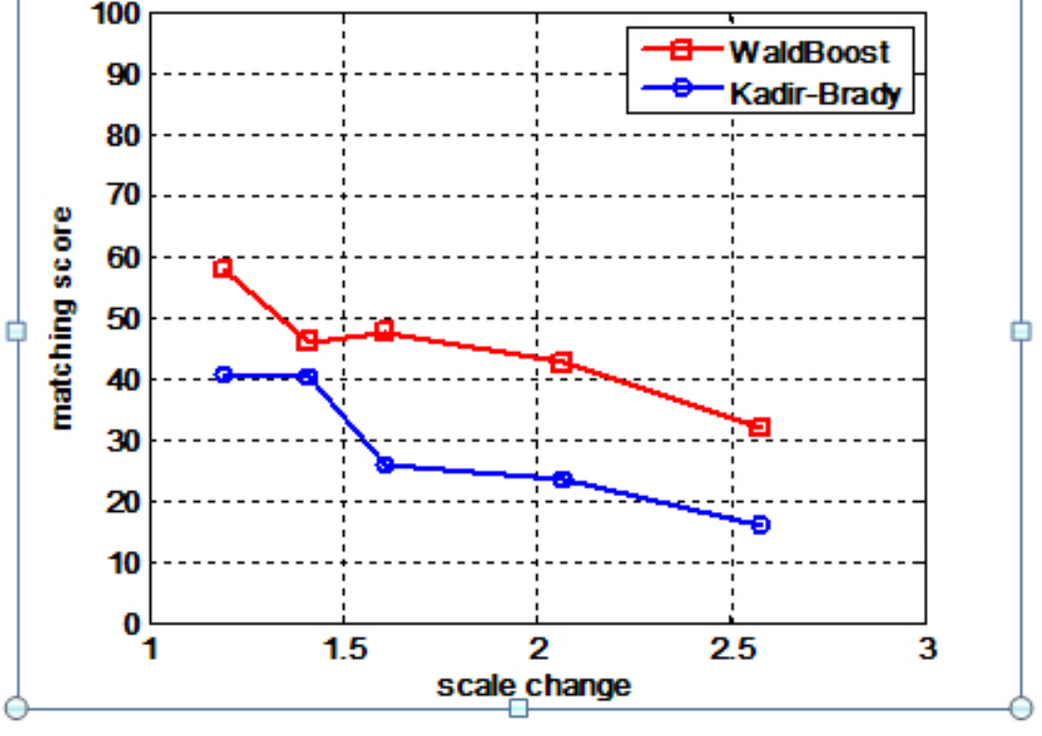

WaldBoost
Hessian-Laplace
repeatability %
scale change
0
10
20
30
40
50
60
70
80
90
100
1
1.5
2
2.5
3

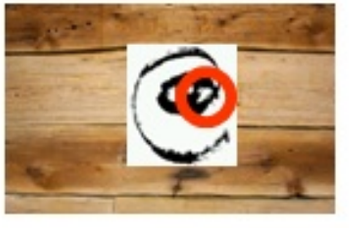

WaldBoost
Hessian-Laplace
matching score
scale change
0
10
20
30
40
50
60
70
80
90
100
1
1.5
2
2.5
3

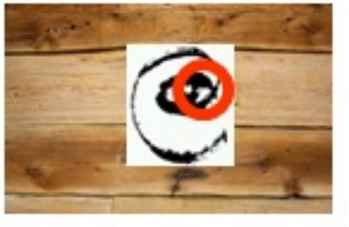


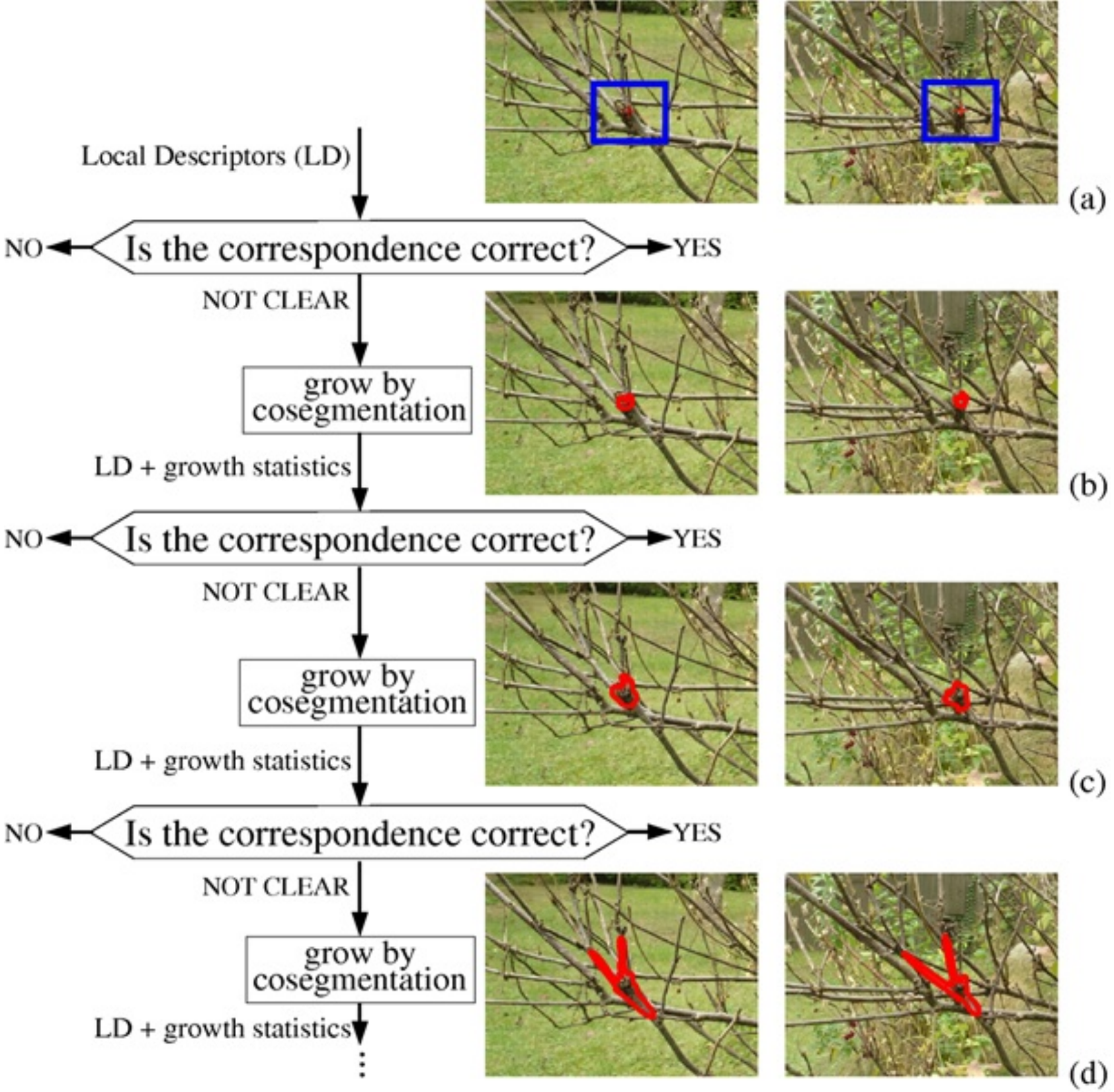


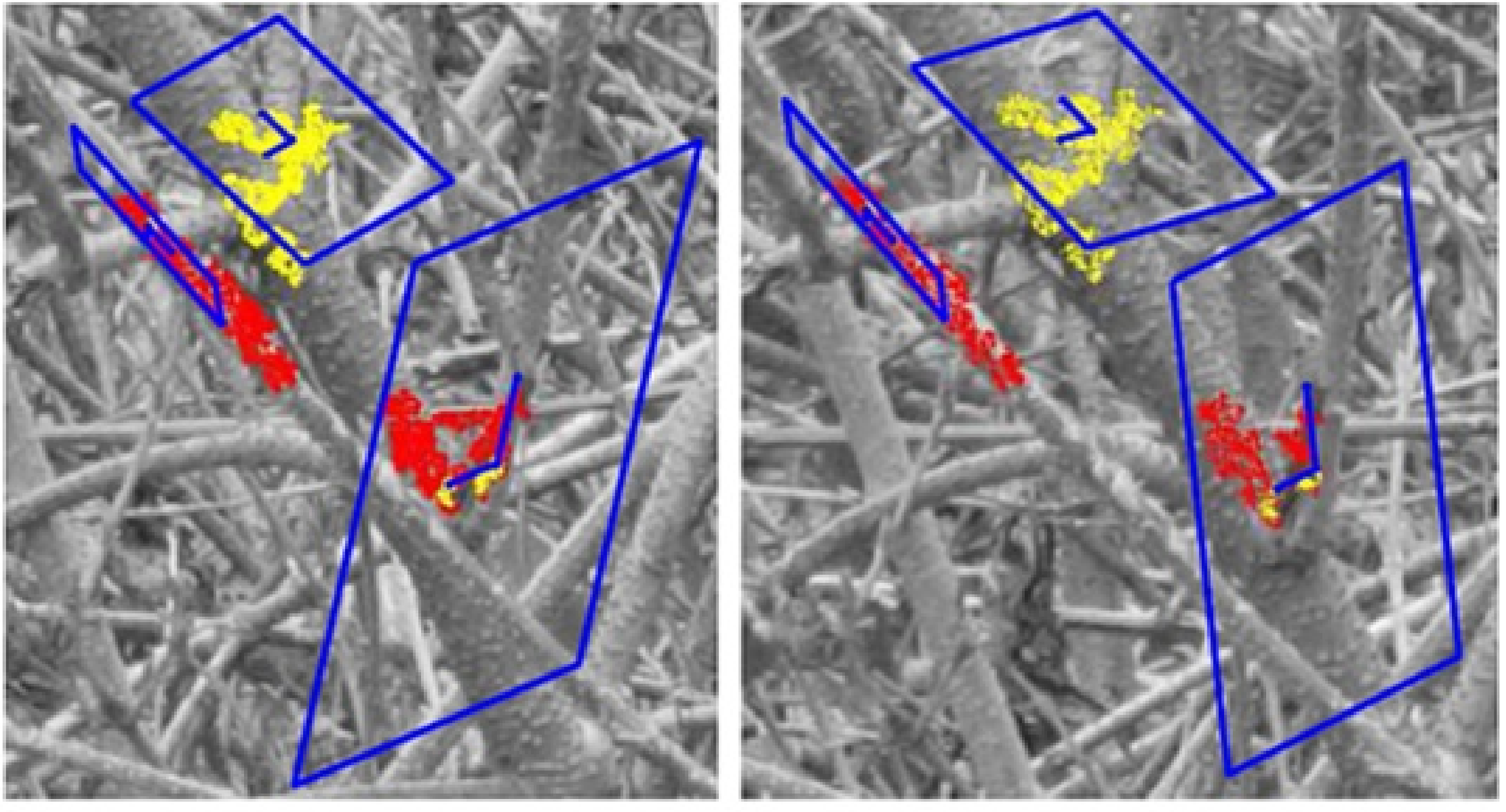

WaldBoost
Kadir-Brady
repeatability %
scale change
100
90
80
70
60
50
40
30
20
10
0
1
1.5
2
2.5

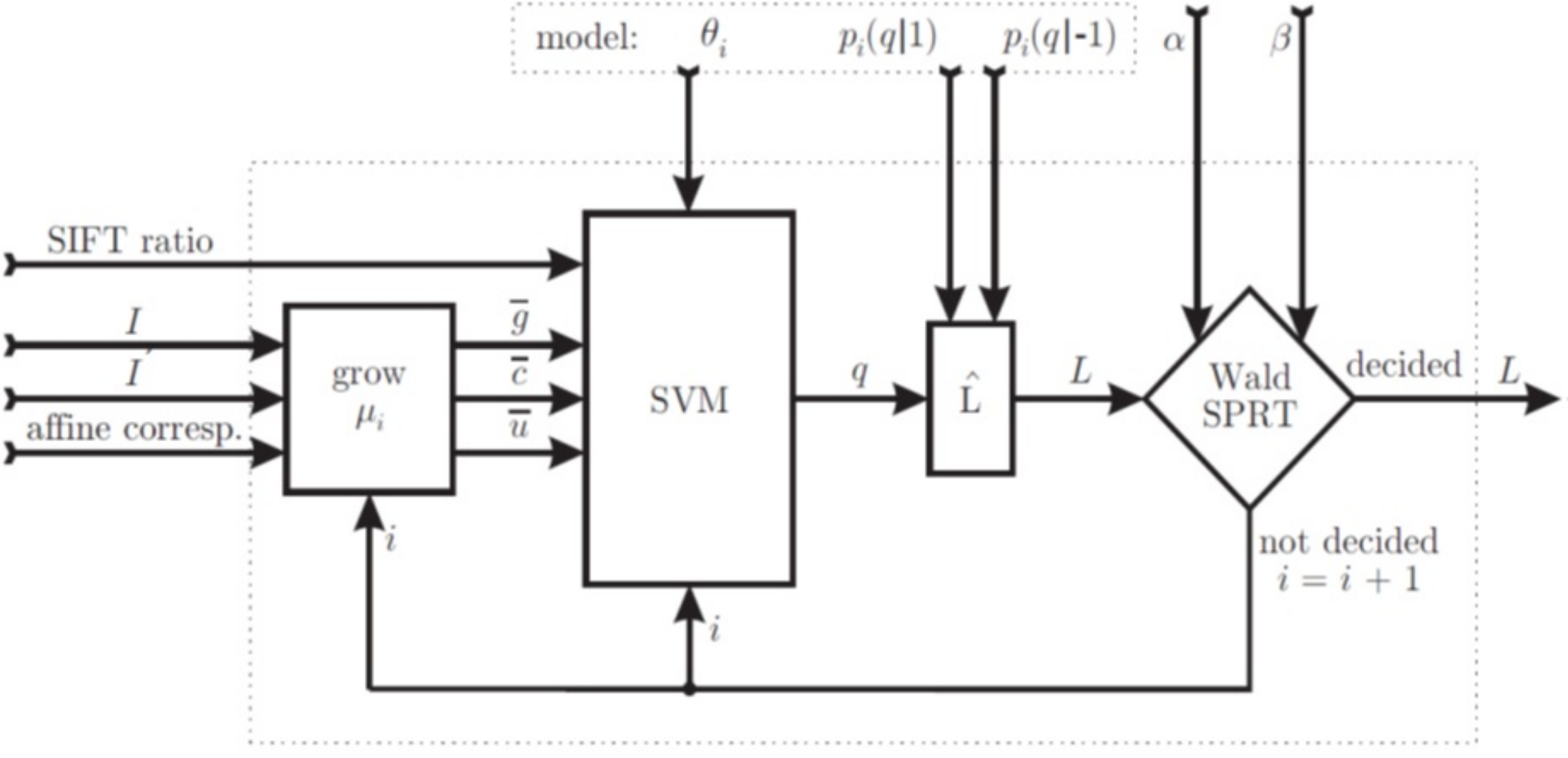

WaldBoost
Kadir-Brady
matching score
scale change
100
90
80
70
60
50
40
30
20
10
0
1
1.5
2
2.5
3



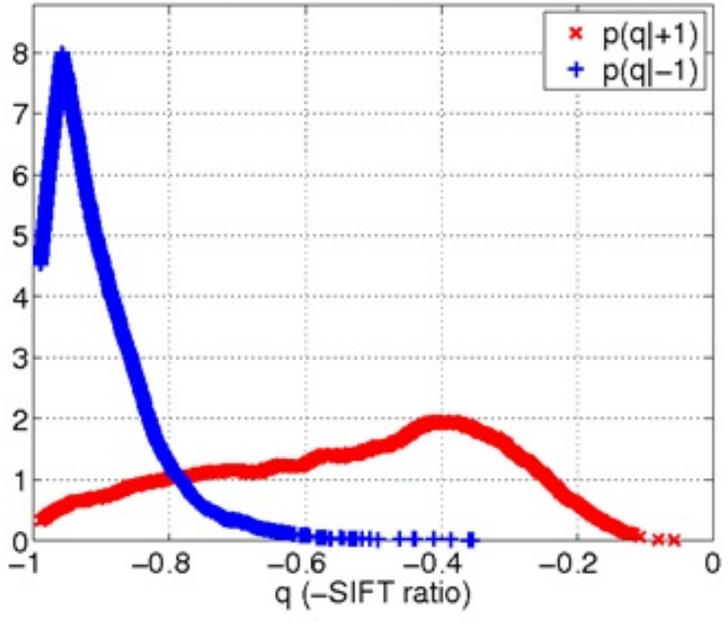



Local Descriptors (LD)
Is the correspondence correct?
NO
YES
NOT CLEAR
grow by cosegmentation
LD + growth statistics
Is the correspondence correct?
NO
YES
NOT CLEAR
grow by cosegmentation
LD + growth statistics
Is the correspondence correct?
NO
YES
NOT CLEAR
grow by cosegmentation
LD + growth statistics
(a)
(b)
(c)
(d)

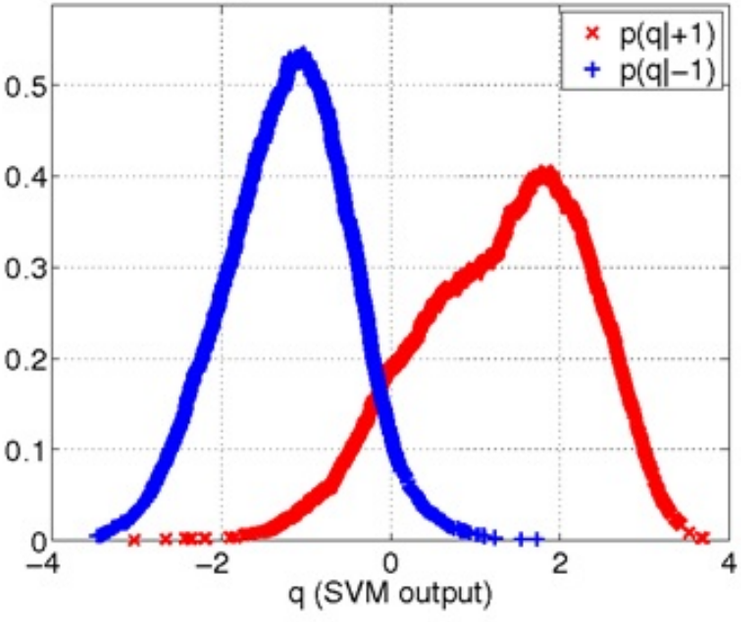



model:
θ_i
$p_i(q|1)$
$p_i(q|\text{-}1)$
α
β
SIFT ratio
I
I'
affine corresp.
grow
μ_i
$\overline{g}$
$\overline{c}$
$\overline{u}$
SVM
q
$\hat{\text{L}}$
L
Wald
SPRT
decided
L
not decided
$i = i + 1$
i
i

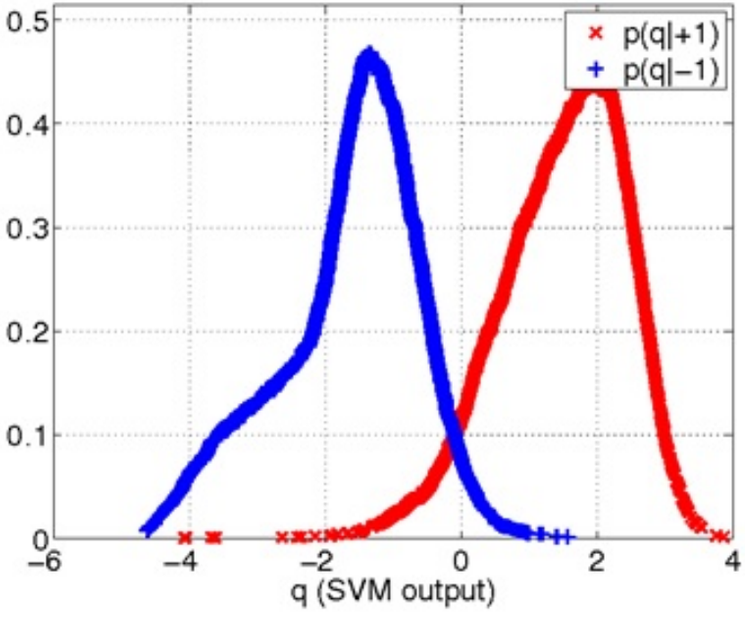
$i = 1 : \mu = 0$ (SIFT ratio only)



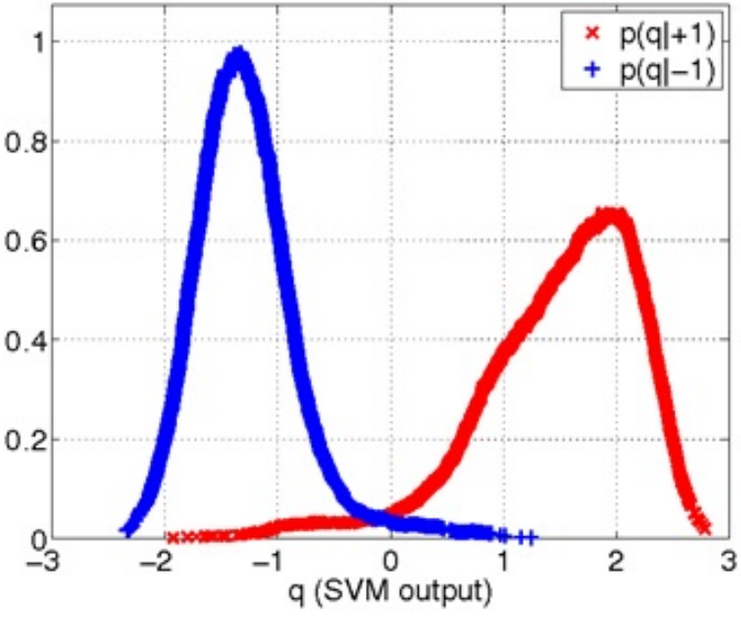
$i = 2 : \mu = 10$

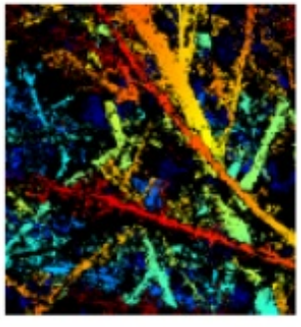


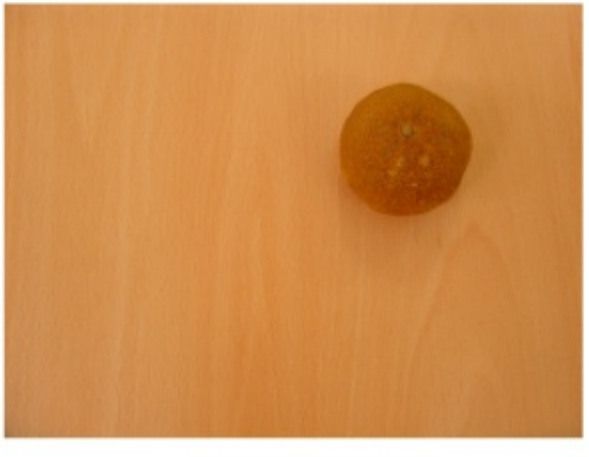
$i = 3 : \mu = 100$

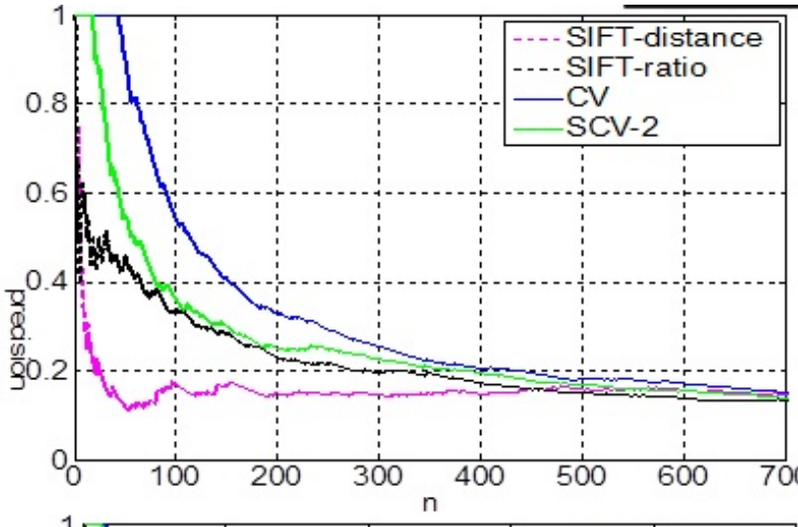


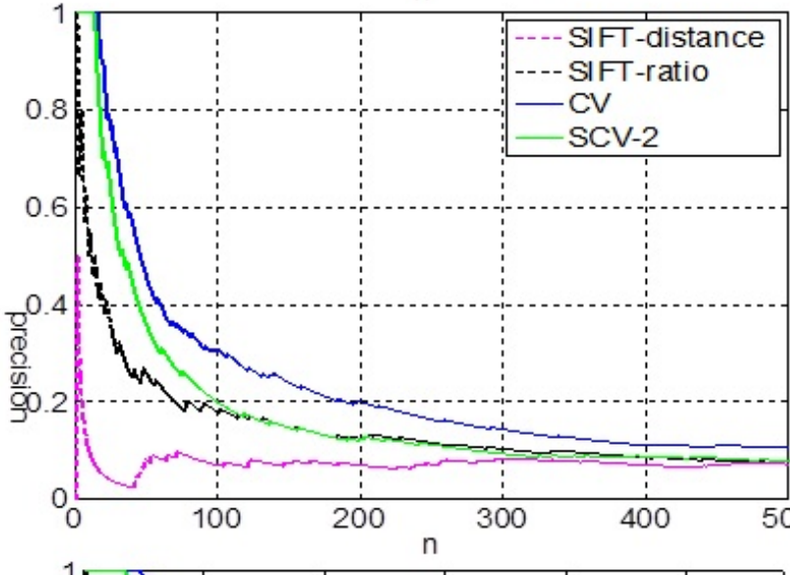
$i = 4 : \mu = 1000$

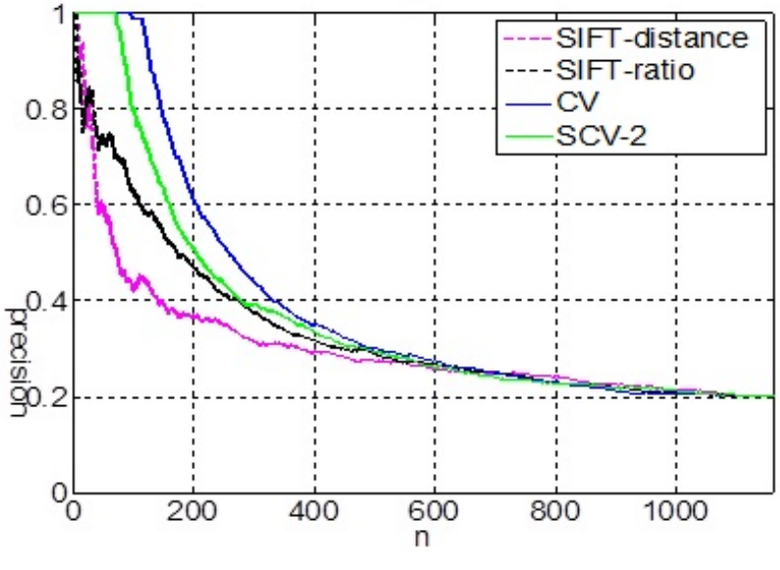


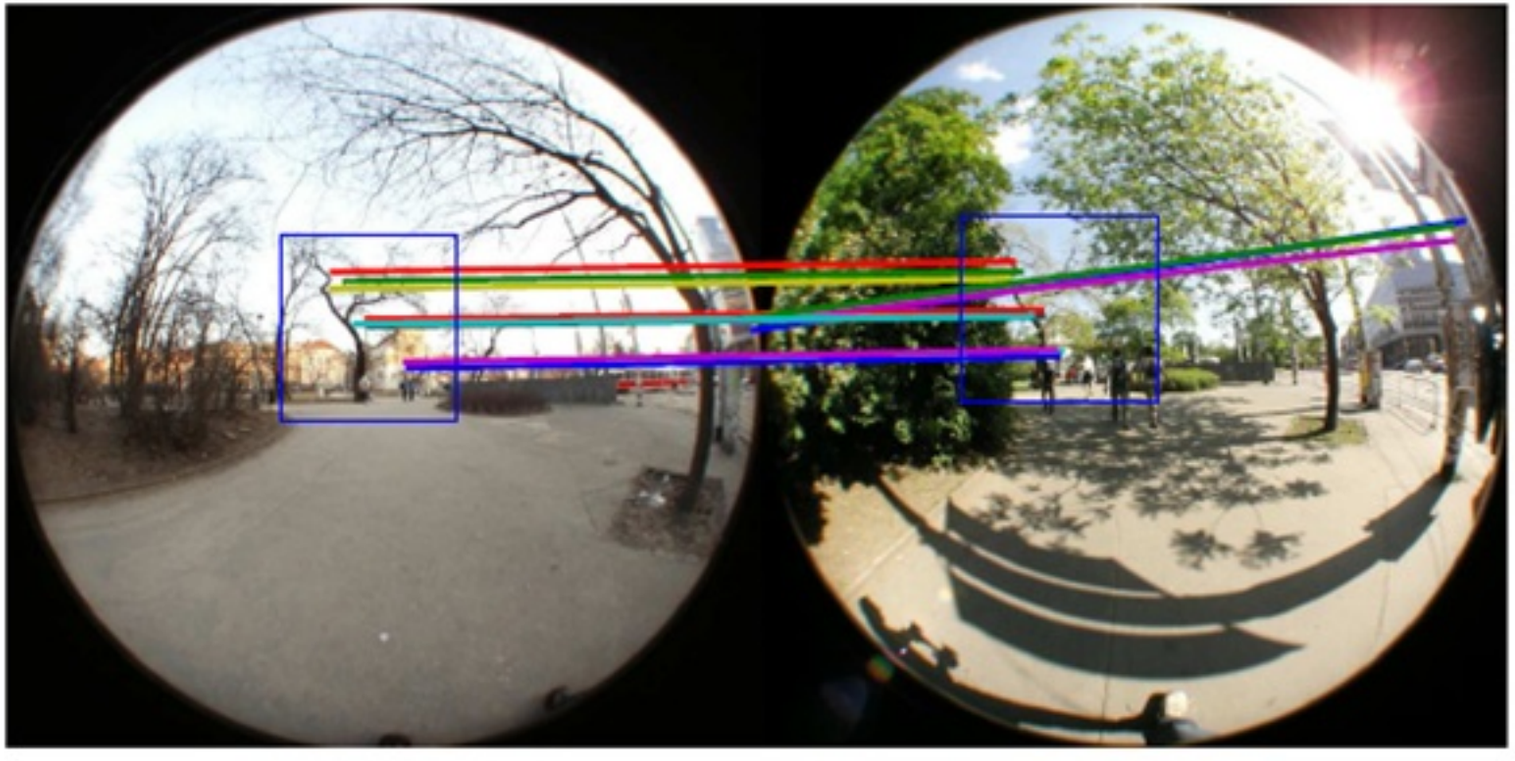




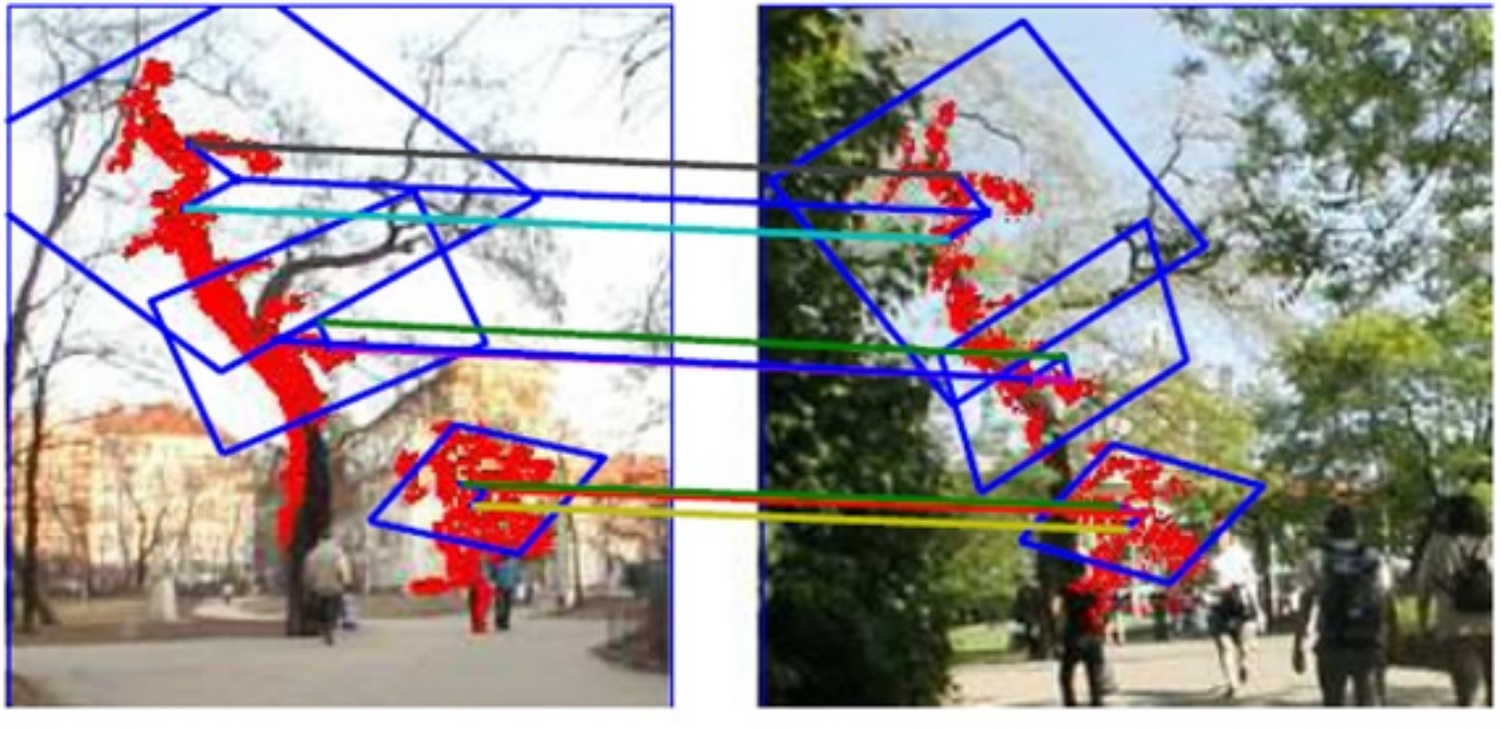

SIFT-distance
SIFT-ratio
CV
SCV-2
precision
n
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600


SIFT-distance
SIFT-ratio
CV
SCV-2
precision
n
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400


SIFT-distance
SIFT-ratio
CV
SCV-2
precision
n
0
0.2
0.4
0.6
0.8
1
0
200
400
600
800
1000



zoom (grown regions)



Randomised RANSAC [Matas, Chum 02]

Center for Machine
Perception Prague

Repeat $k/(1\text{-}\alpha)$ times

1. Hypothesis generation
2. Model pre-verification $T_{d,d}$ test
   - Verify $d << N$ data points, reject
   - the model if not all d data points
   - are consistent with the model
3. Model verification

   Verify the rest of the data points

Time
t_M

$\overline{m}_s \cdot V$

V – average number of data points verified

α – probability that a good model is rejected by $T_{d,d}$ test

$$t = \frac{k}{1-\alpha}(t_M + \overline{m}_s V)$$

WaldSAC

Center for Machine
Perception Prague

Repeat $k/(1\text{-}1/A)$ times
1. Hypothesis generation
2. Model verification
   use SPRT

Time
t_M

$\overline{m}_s \cdot C \log A$

$C \approx ((1-\delta)\log\frac{1-\delta}{1-\varepsilon} + \delta \log\frac{\delta}{\varepsilon})^{-1}$

$$t(A) = \frac{k}{(1-1/A)}(t_M + \overline{m}_S C \log A)$$

In sequential statistical decision problem decision errors are traded off for time. These are two incomparable quantities, hence the constrained optimization.

In WaldSAC, decision errors cost time (more samples) and there is a single minimised quantity, time t(A), a function of a single parameter A.