Rozpoznávání tváří I

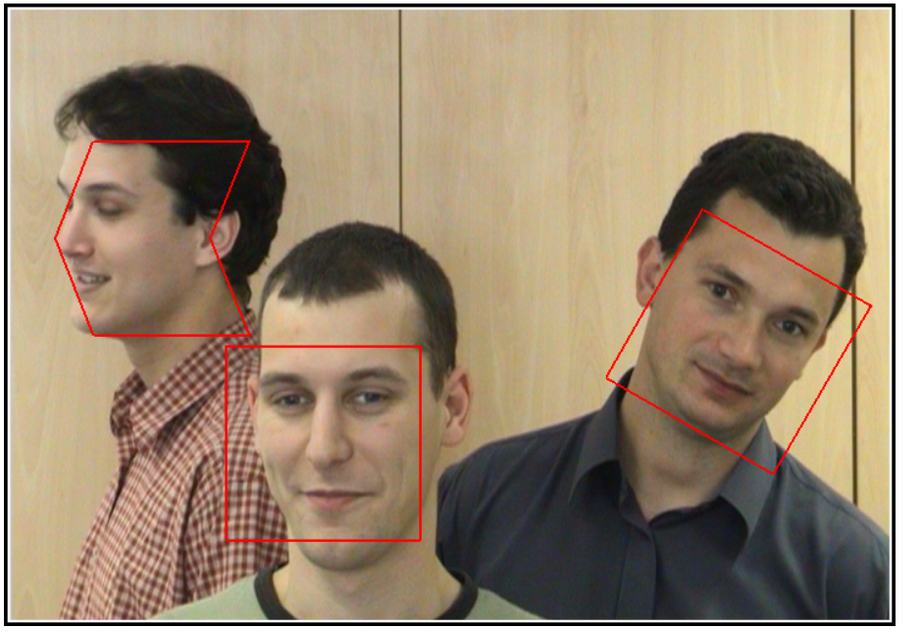
Vojtěch Franc
Centrum strojového vnímání, ČVUT FEL Praha

Biometrie ZS 2011

Poděkování Janu Šochmanovi za slajdy vysvětlující AdaBoost

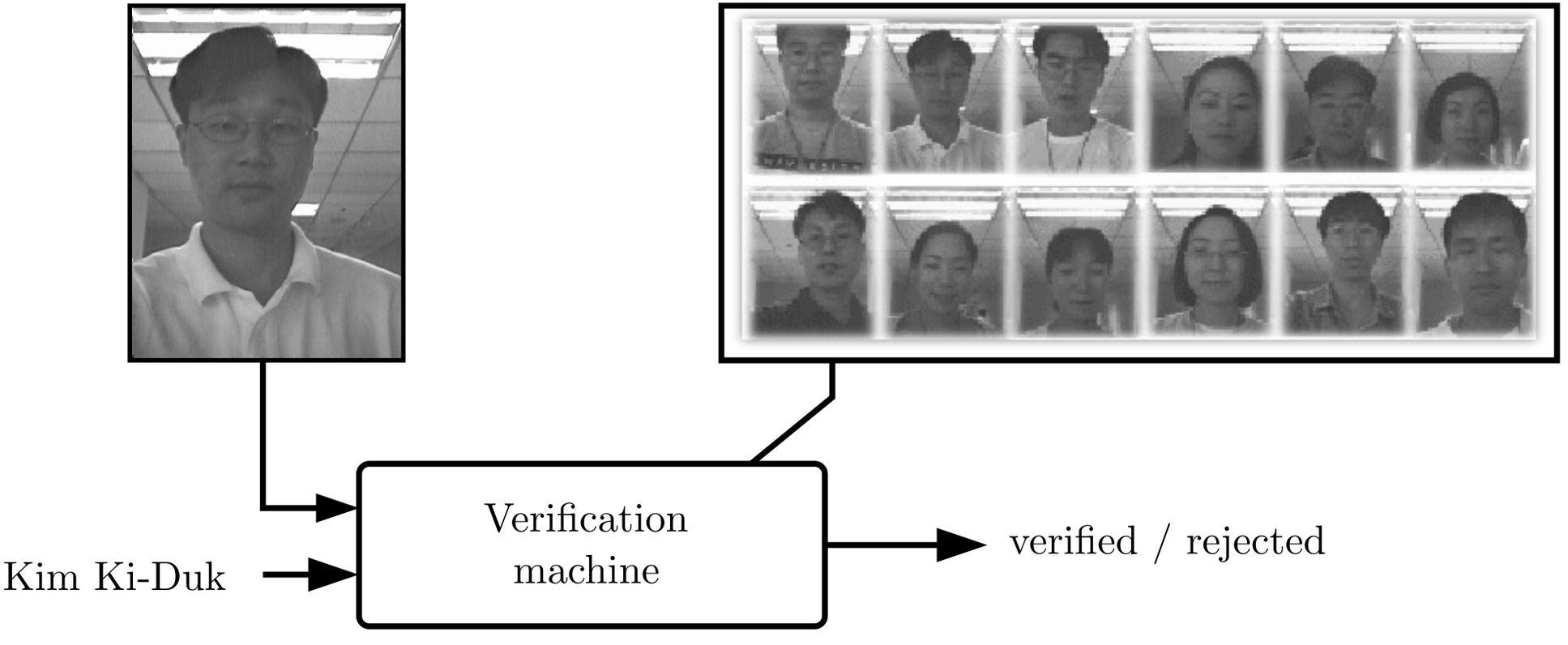
Úlohy rozpoznávání tváří: Detekce

- Cíl: lokalizovat tvář v obraze
- Vstup: obrázek
- Výstup: pozice tváře (souřadnice, velikost, popřípadě orientace)



Úlohy rozpoznávání tváří: Verifikace

- Cíl: ověřit identitu tváře
- Vstup: obrázek (nebo video) tváře a jméno
- Výstup: binární rozhodnutí - idenita ověřena nebo zamítnuta

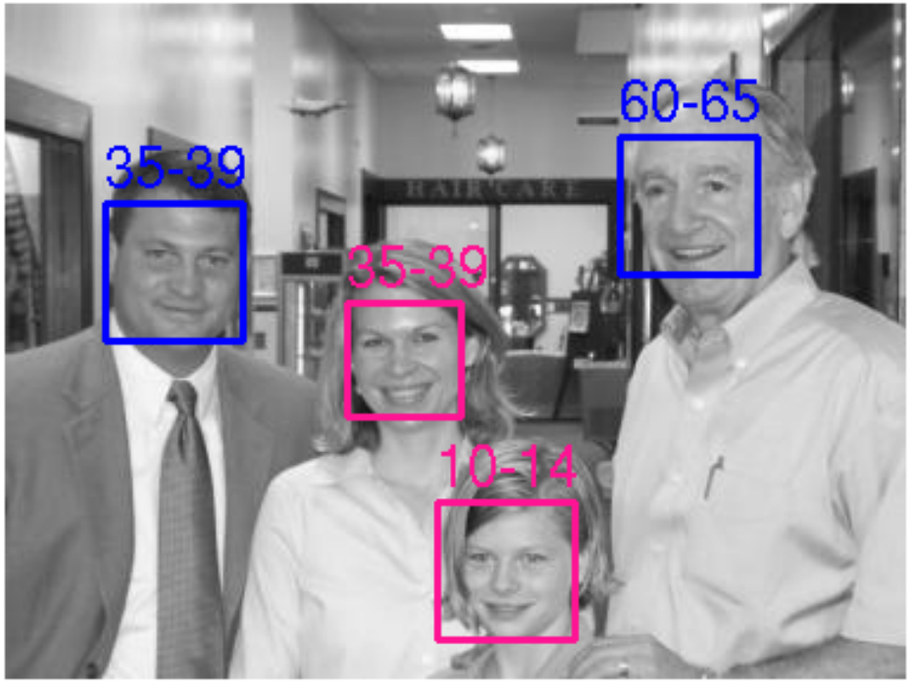


Úlohy rozpoznávání tváří: Identifikace

- Cíl: přiřadit tváři jednu z N identit
- Vstup: obrázek (nebo video)
- Výstup: jedna z N identit popřípadě odpověď není v databázi

Úlohy rozpoznávání tváří: Kategorizace

- Cíl: zařadit tvář do jedné z N skupin
- Vstup: obrázek (nebo video)
- Výstup: jedna z N skupin
- Příklady kategorií: muž/žena, věkové kategorie, nálada (úsměv/smutek/neutrální), rasa (běloch/černoch/Asiat)

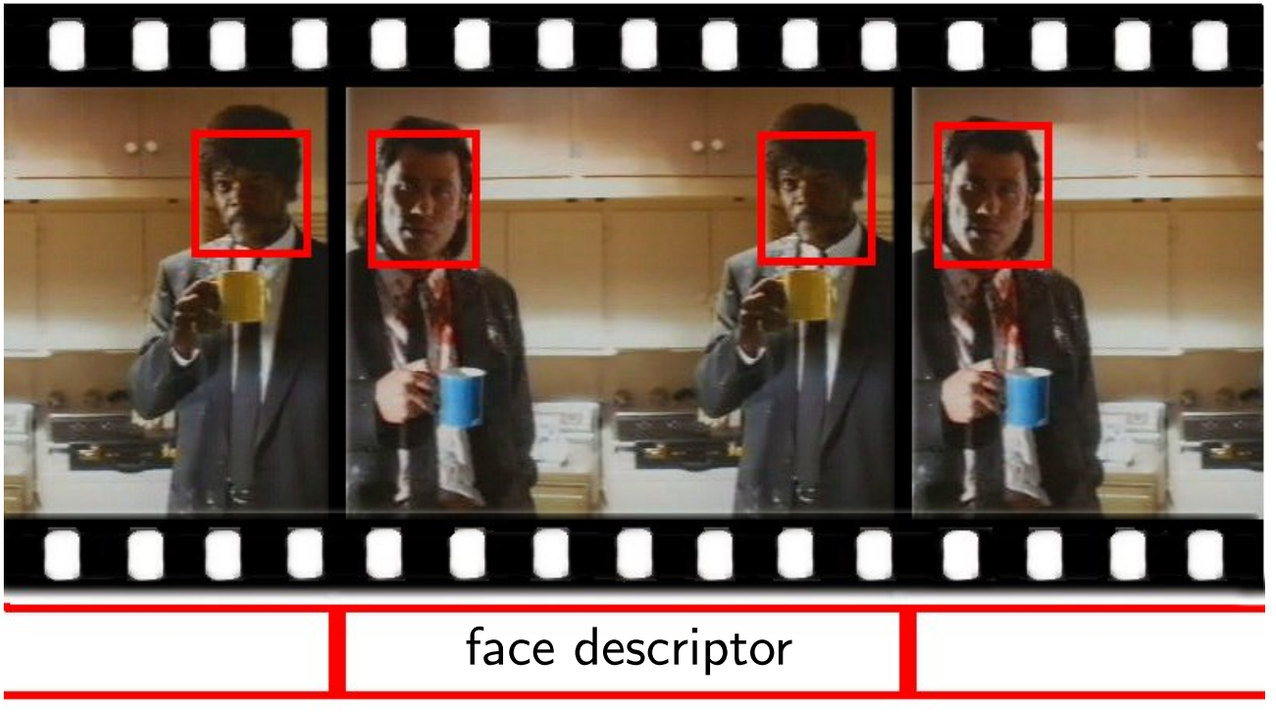


Úlohy rozpoznávání tváří: Detekce mluvčího

- Cíl: nalézt ve videu kdo mluví
- Vstup: video sekvence s lidmi
- Výstup: lokalizace mluvících tváří

Úlohy rozpoznávání tváří: Vyhledávání

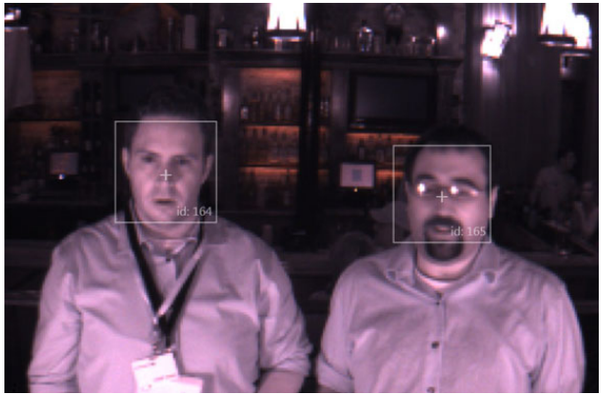
- Cíl: nalézt v databázi (videu) tváře podle zadaného popisu
- Vstup: textový popis / obrázek tváře
- Výstup: N nejbližších obrázků v databázi



Part of the ISO/IEC JTC 1SC 29WG 11 (MPEG7) standard.

Rozdělení úloh podle vstupu

- Senzor:
  - kamera pracující ve viditelném spektru
  - infračervená kamera (s IR přísvícením)
  - stereo kamera
  - 3D skener
- statický obrázek / video sekvence



Metody statistického rozpoznávání

- Formální popis úloh rozpoznávání:
  - Množina pozorování X (např. obrázek, video sekvence)
  - Množina skrytých stavů Y (např. identita, věková kategorie,...)
  - Statistický model $p(x, y)$ generuje dvojice $(x, y) \in X \times Y$.
  - Množina rozhodnutí D (např. tvář/netvář, identita/neznámá_tvář)
  - Rozhodovací funkce $h\colon X \to D$ (h nazývá klasifikátor pokud $D = Y$)
  - Hledání optimální rozhodovací funkce se vyjádřuje jako optimalizační úloha. Např. se minimalizuje pravděpodobnost chybného klasifikace

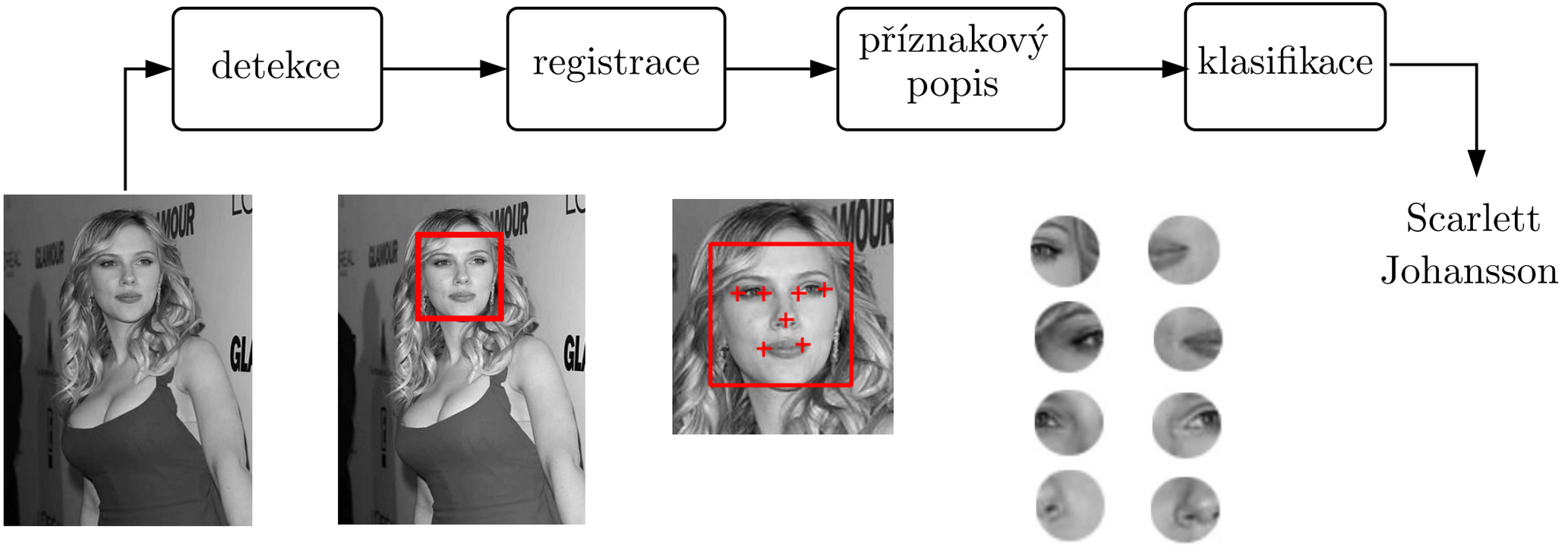
$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} E_{p(x,y)}[h(x) \neq y]$$

- Problém: statistický model $p(x, y)$ není téměř nikdy k dispozici.
- Metody učení hledají rozhodovací funkci h na základě množiny trénovacích příkladů $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ generovaných z $p(x, y)$.

Proč je rozpoznávání tváří těžké?

- Jednoduché metody jako template-matching nefungují dobře.
- Množina pozorování X je obrovská, např. existuje 256^{10000} šedotónových obrázků 100×100 pixelů, proto je těžké odhadnout statistiký model $p(x, y)$.
- Obázky tváří patřící do stejné třídy mají velkou variabilitu:
  - změna pozice, meřítka, rotace (roll, pan, tilt)
  - změna osvětlení
  - změna výrazu tváře (úsměv, smutek, neutralní, ...)
  - zákryty (brýle, pokrývka hlavy), změna účesu, make up, stárnutí ...

Stavební bloky typického rozpoznávacho systému

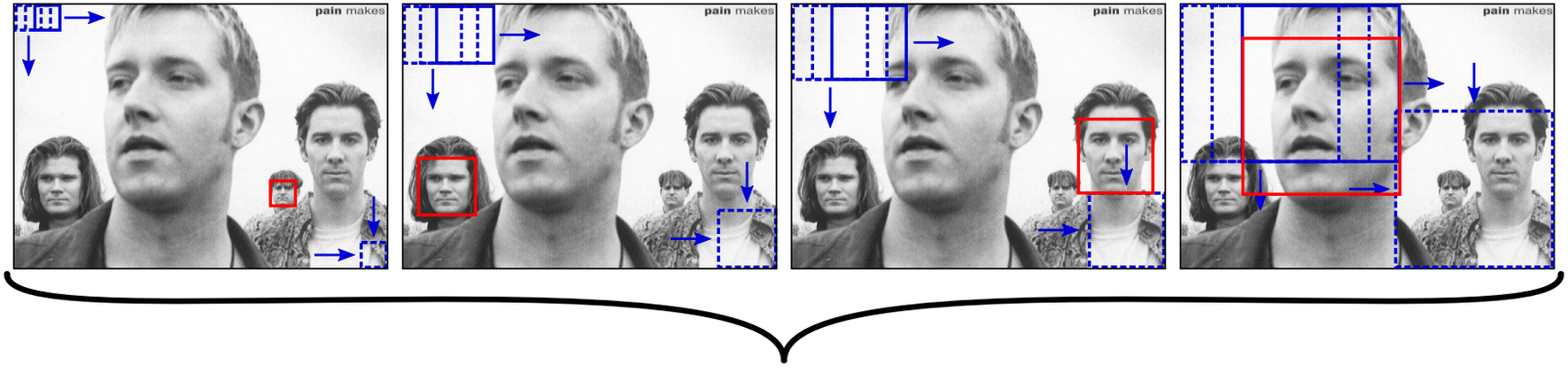


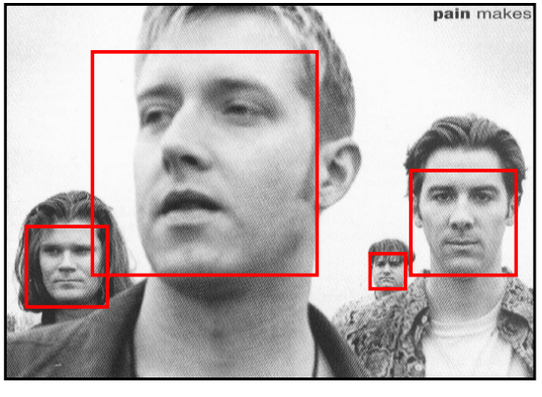
Detektor tváří

- Vstup: obrázek $x \in X$
- Výstup: pozice tváře (souřadnice, velikost, popřípadě orientace) $d \in D$.
  Množina všech možných pozic D je obrovská.
- Cíl: nalézt detektor $h\colon X \to D$, který je:
  1. Přesný: nízký počet i) chybných detekcí a ii) přehlídnutých tváří.
  2. Rychlý: krátký čas potřebný ke zpracování obrázku.
- Přelomová práce:
  Viola, Jones: Robust Real-time Object Detection, IJCV 2001
  - AdaBoost učící algoritmus.
  - Rychle spočítatelné příznaky z integrálního obrázku.
  - Sekvenční rozhodovací pravidlo (kaskádní klasifikátor).

Scanning window approach

Problém detekce tváří lze převést na sekvenci binárních klasifikačních problémů tvář/netvář.





AdaBoost klasifikátor

- AdaBoost klasifikátor $h\colon X \to \{+1, -1\}$ (tvář/netvář) rozhoduje podle znaménka skórovací funkce $f\colon X \to \mathcal{R}$,

$$h(x) = \begin{cases} +1 & \text{pokud} \quad f(x) \geq 0 \\ -1 & \text{pokud} \quad f(x) < 0 \end{cases}$$

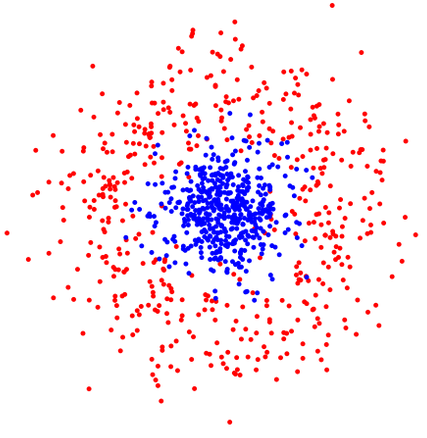
  která je lineární kombinací n slabých klasifikátorů

$$f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_n h_n(x)$$

- $h_i\colon X \to \{+1, -1\}$ ... i-tý slabý klasifikátor
- $\alpha_i \in \mathcal{R}$ ... váha i-tého slabého klasifikátoru
- AdaBoost učící algoritmus vybere z velké množiny slabých klasifikátorů $\mathcal{H}$ malou podmnožinu klasifikátorů a jejich váhy, tak aby výsledný (silný) klasifikátor byl přesný.

AdaBoost algoritmus

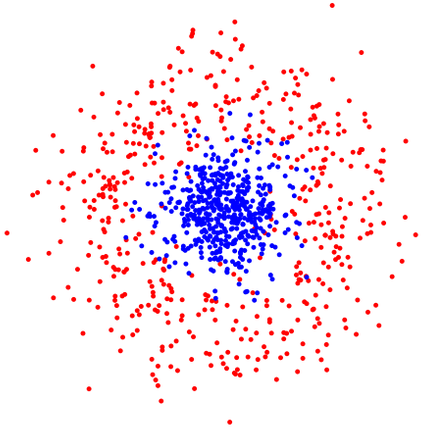
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$



AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

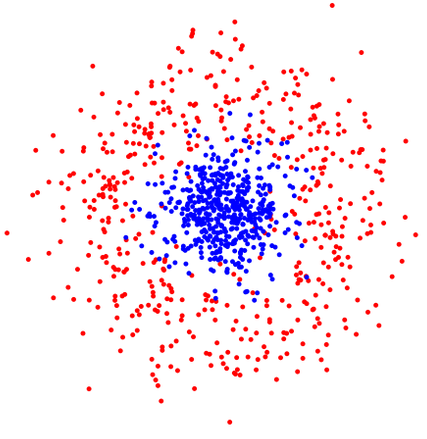


AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

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



AdaBoost algoritmus

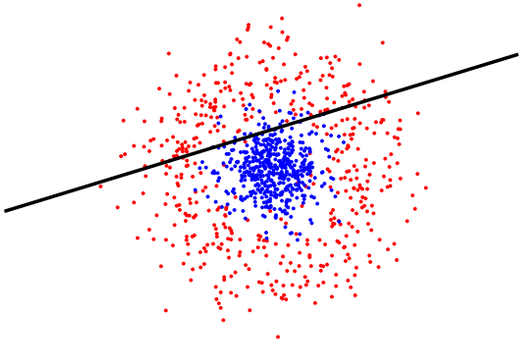
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

$t = 1$

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

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$



AdaBoost algoritmus

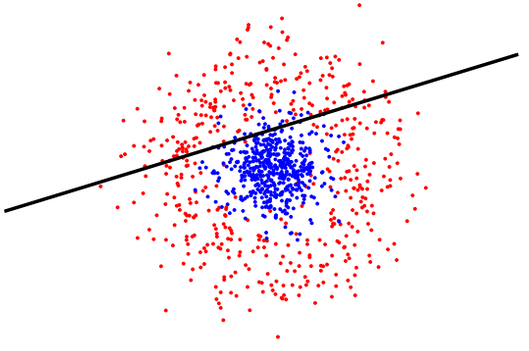
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

$t = 1$

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

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



AdaBoost algoritmus

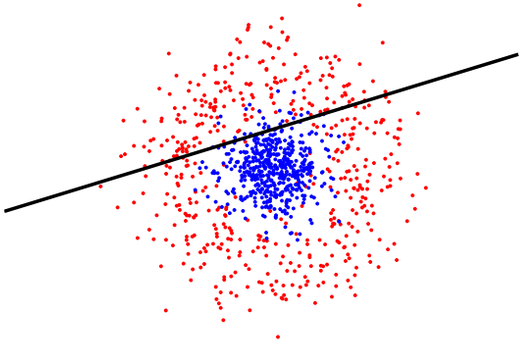
Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

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

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

$t = 1$



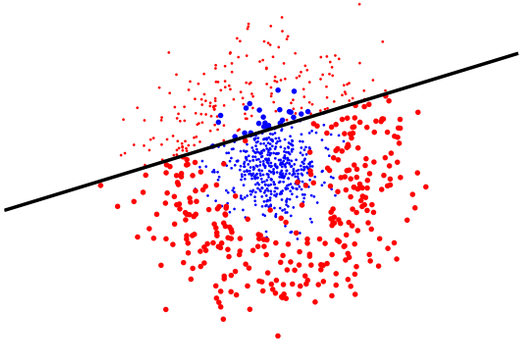
AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

$t = 1$

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



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

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

where Z_t is normalisation factor

AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

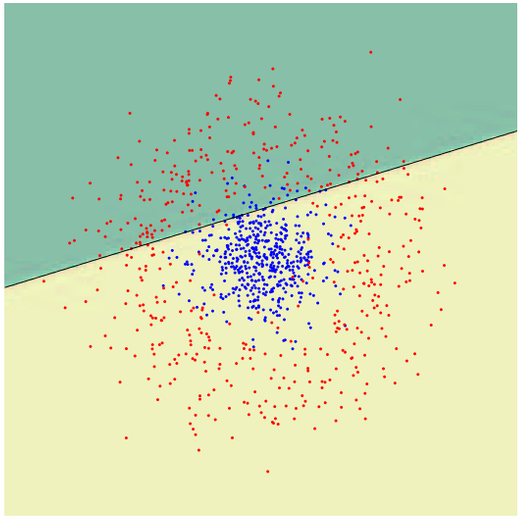
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

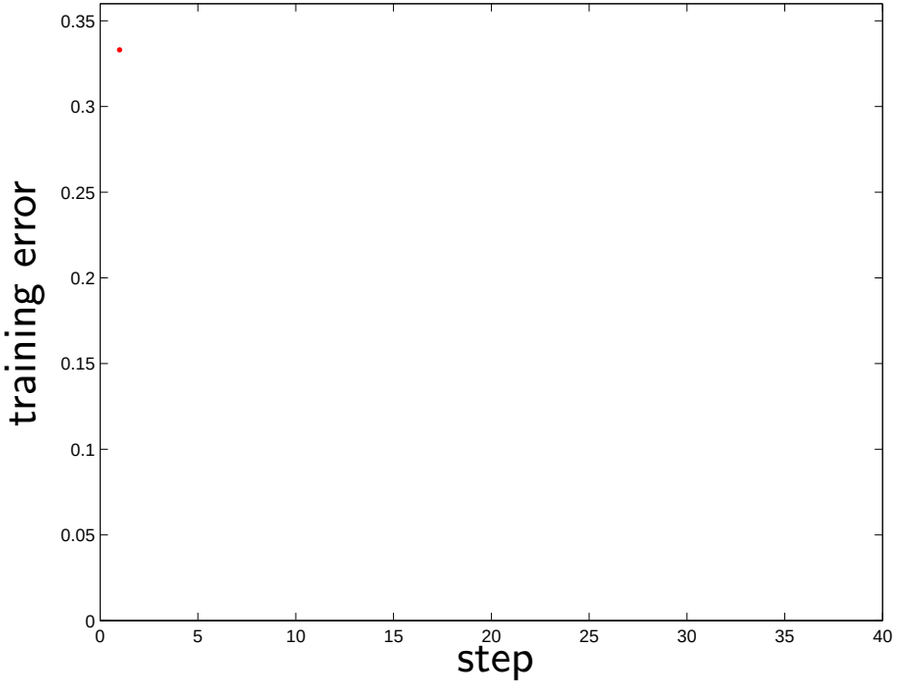
where Z_t is normalisation factor

Output the final classifier:

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

$t = 1$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

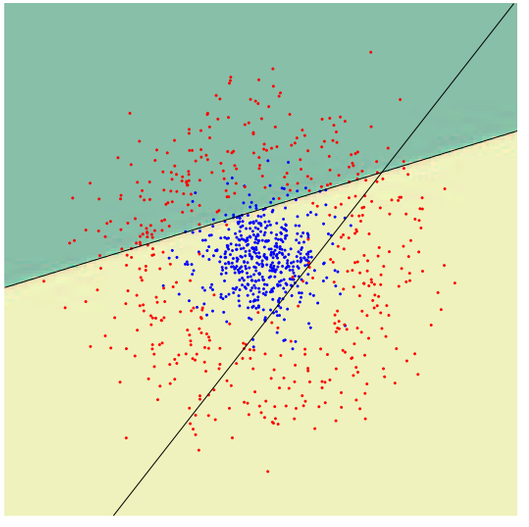
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

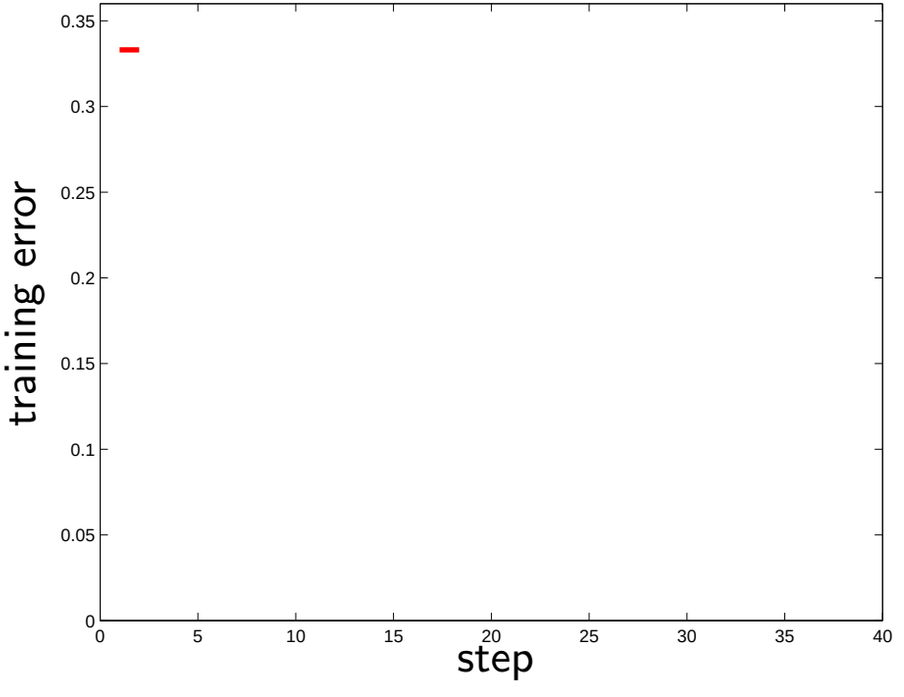
where Z_t is normalisation factor

Output the final classifier:

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

$t = 2$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

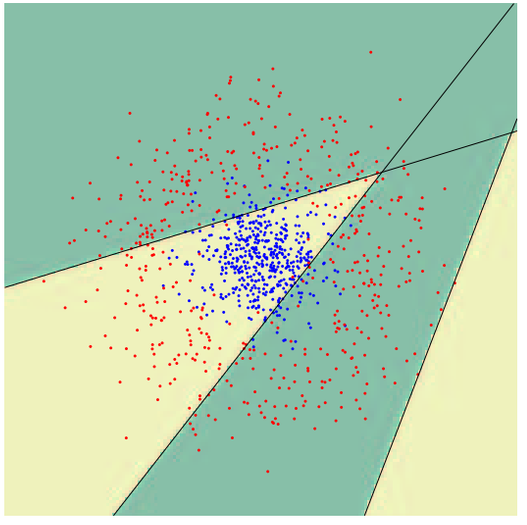
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

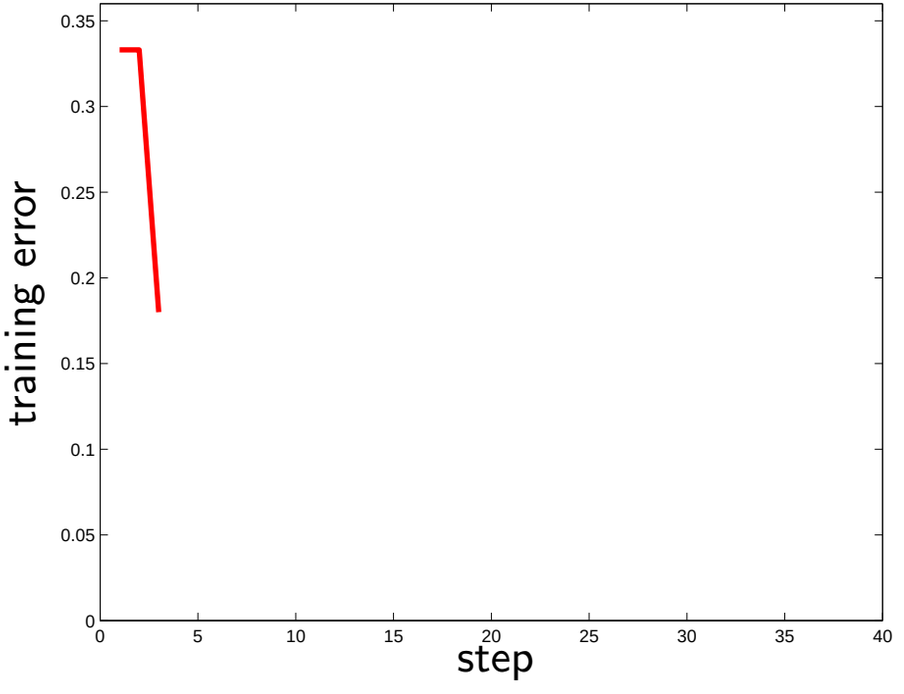
where Z_t is normalisation factor

Output the final classifier:

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

$t = 3$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

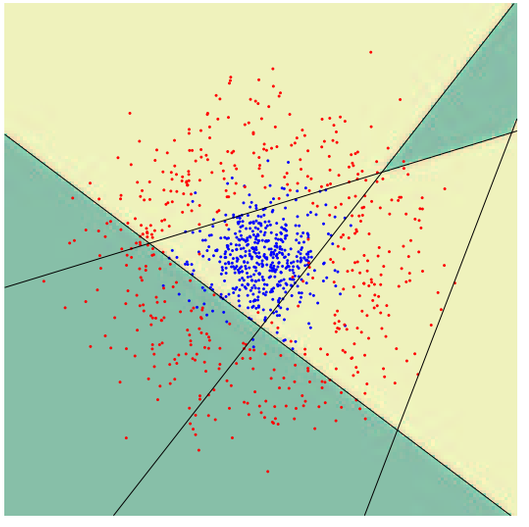
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

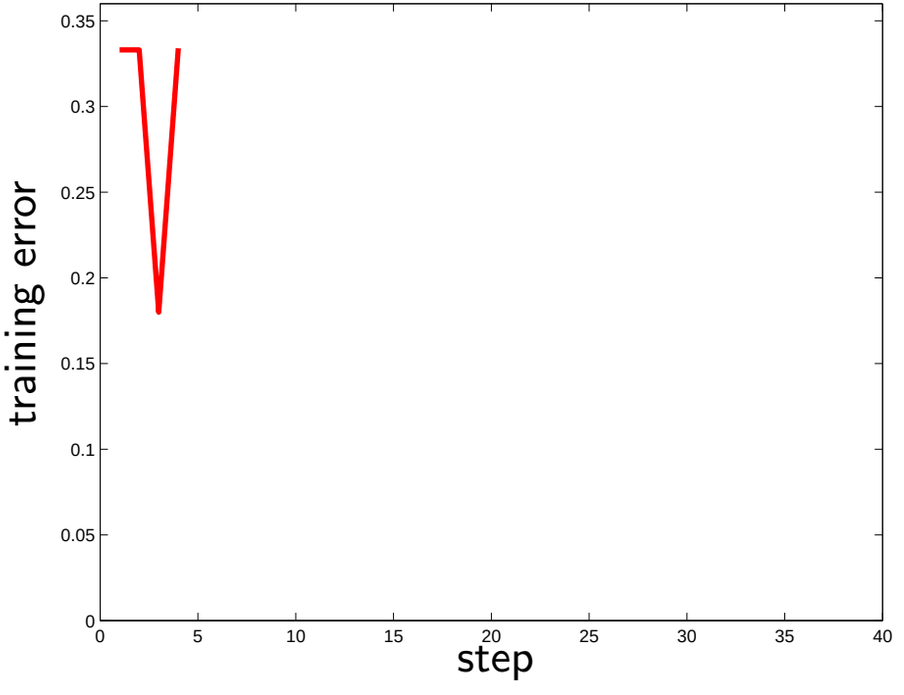
where Z_t is normalisation factor

Output the final classifier:

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

$t = 4$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

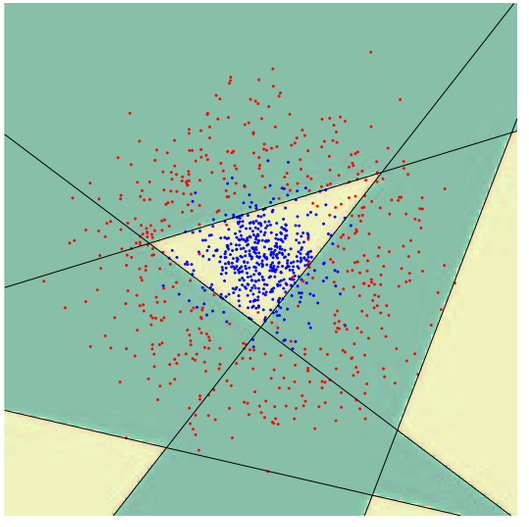
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

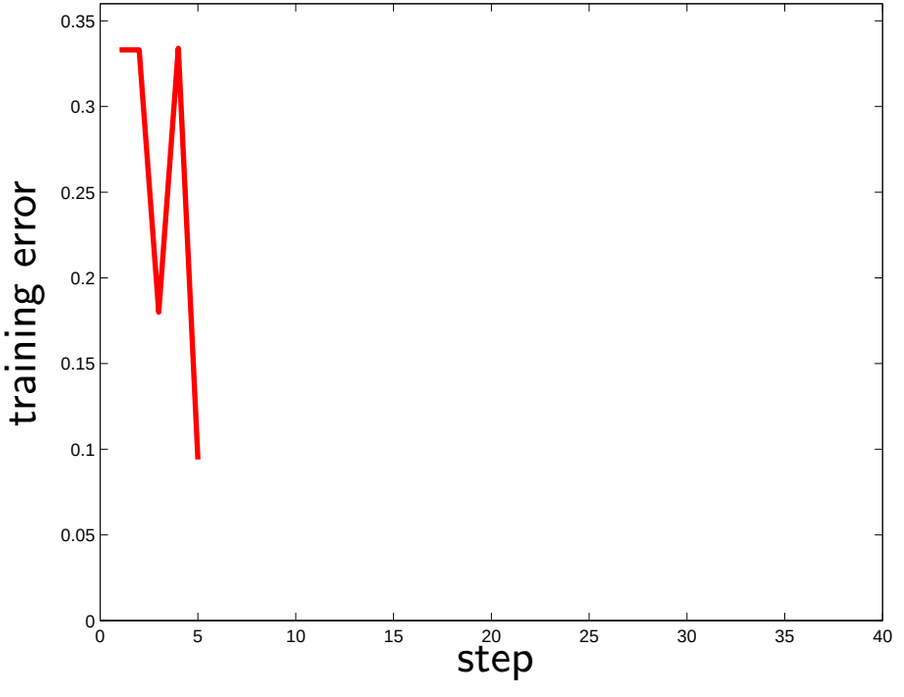
where Z_t is normalisation factor

Output the final classifier:

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

$t = 5$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

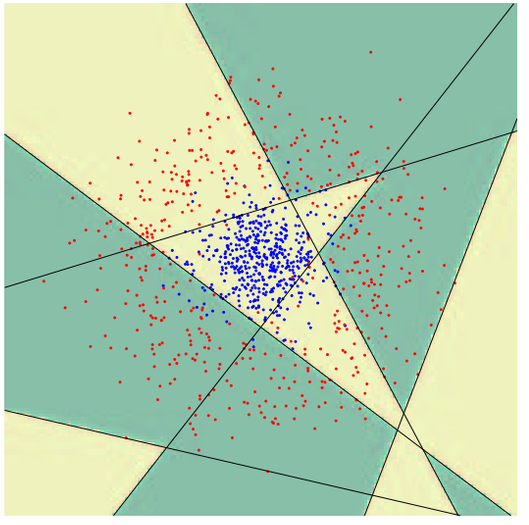
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

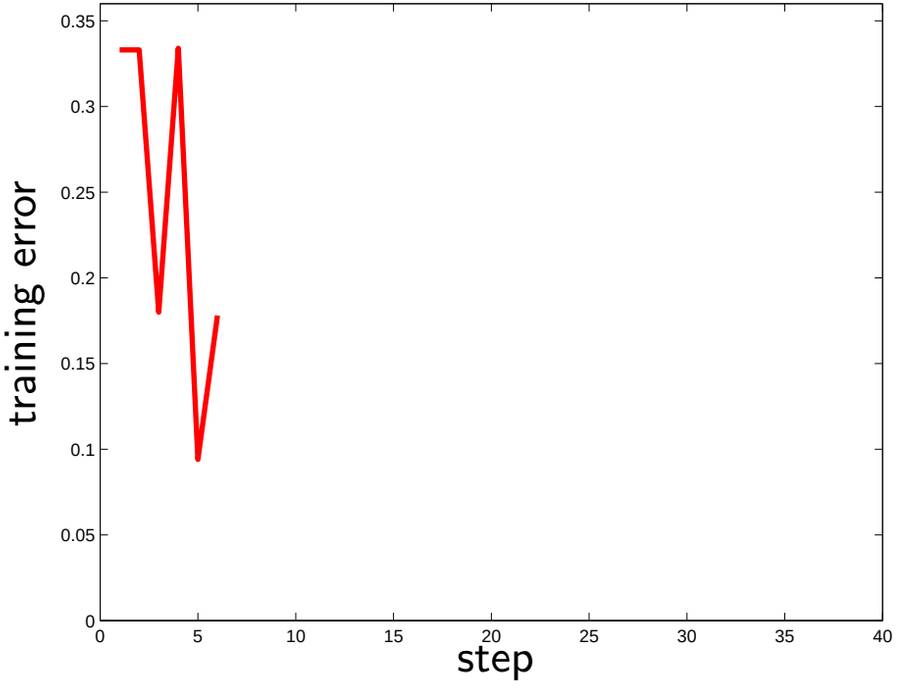
where Z_t is normalisation factor

Output the final classifier:

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

$t = 6$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

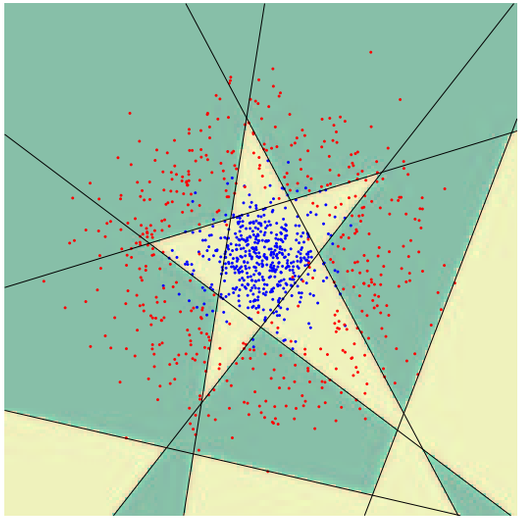
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

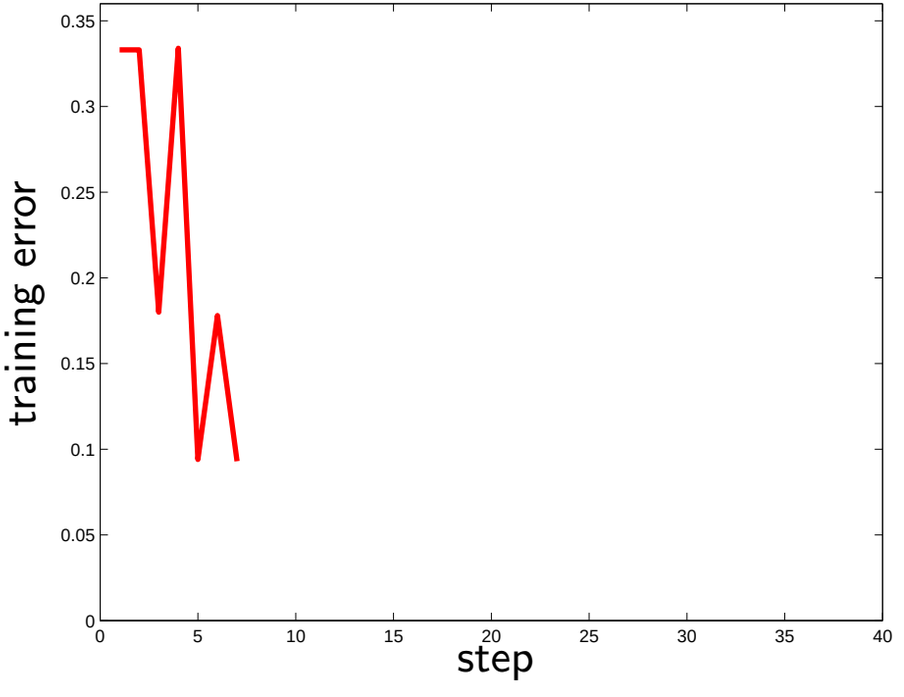
where Z_t is normalisation factor

Output the final classifier:

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

$t = 7$





AdaBoost algoritmus

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

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

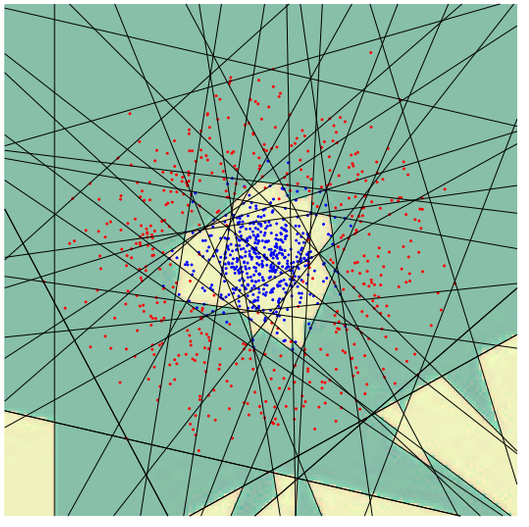
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

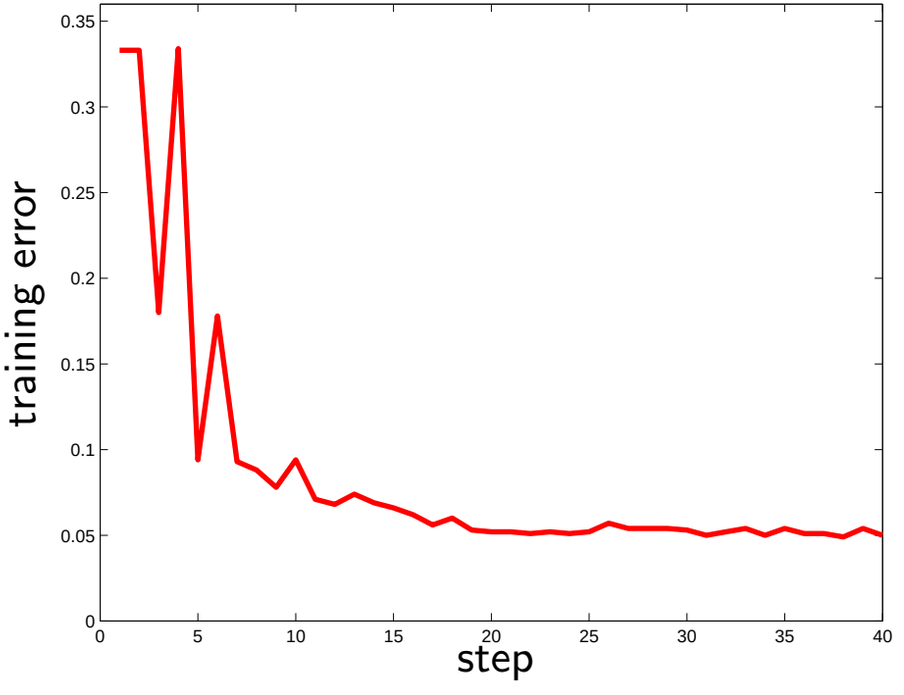
where Z_t is normalisation factor

Output the final classifier:

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

$t = 40$

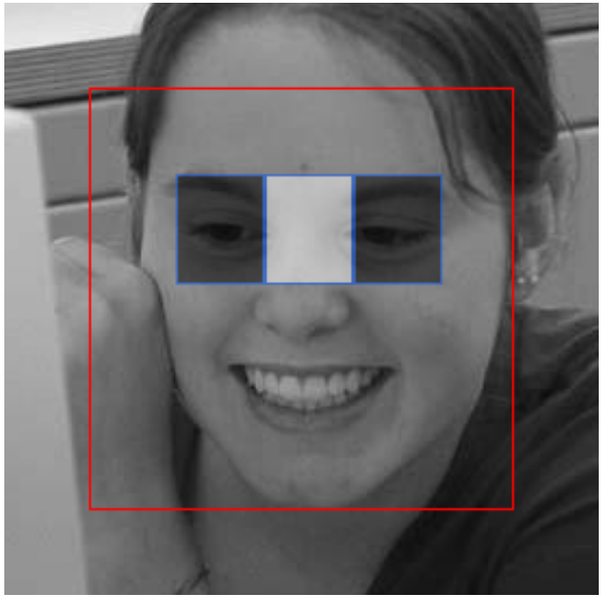




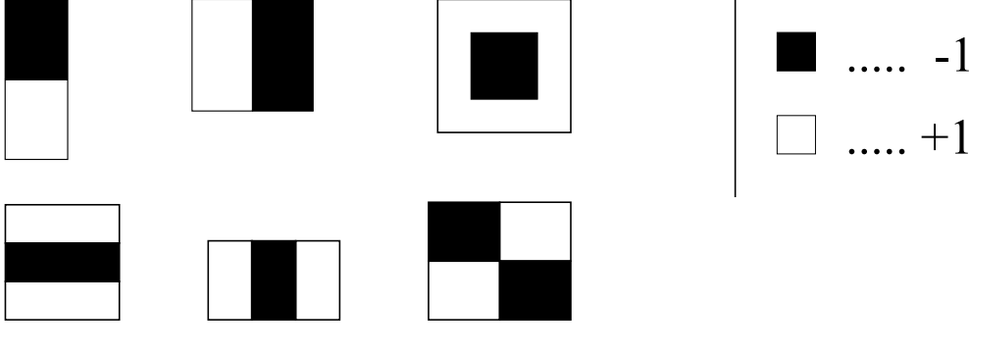
Slabý klasifikátor pro detektor tváří

- Slabý klasifikátor je oprahovaná odezva Haarova filtru

$$h_i(x) = \text{sign}\left(\sum_{(u,v)\in A_i^+(x)} x(u,v) - \sum_{(u,v)\in A_i^-(x)} x(u,v) + \theta\right)$$



Příklady filtrů

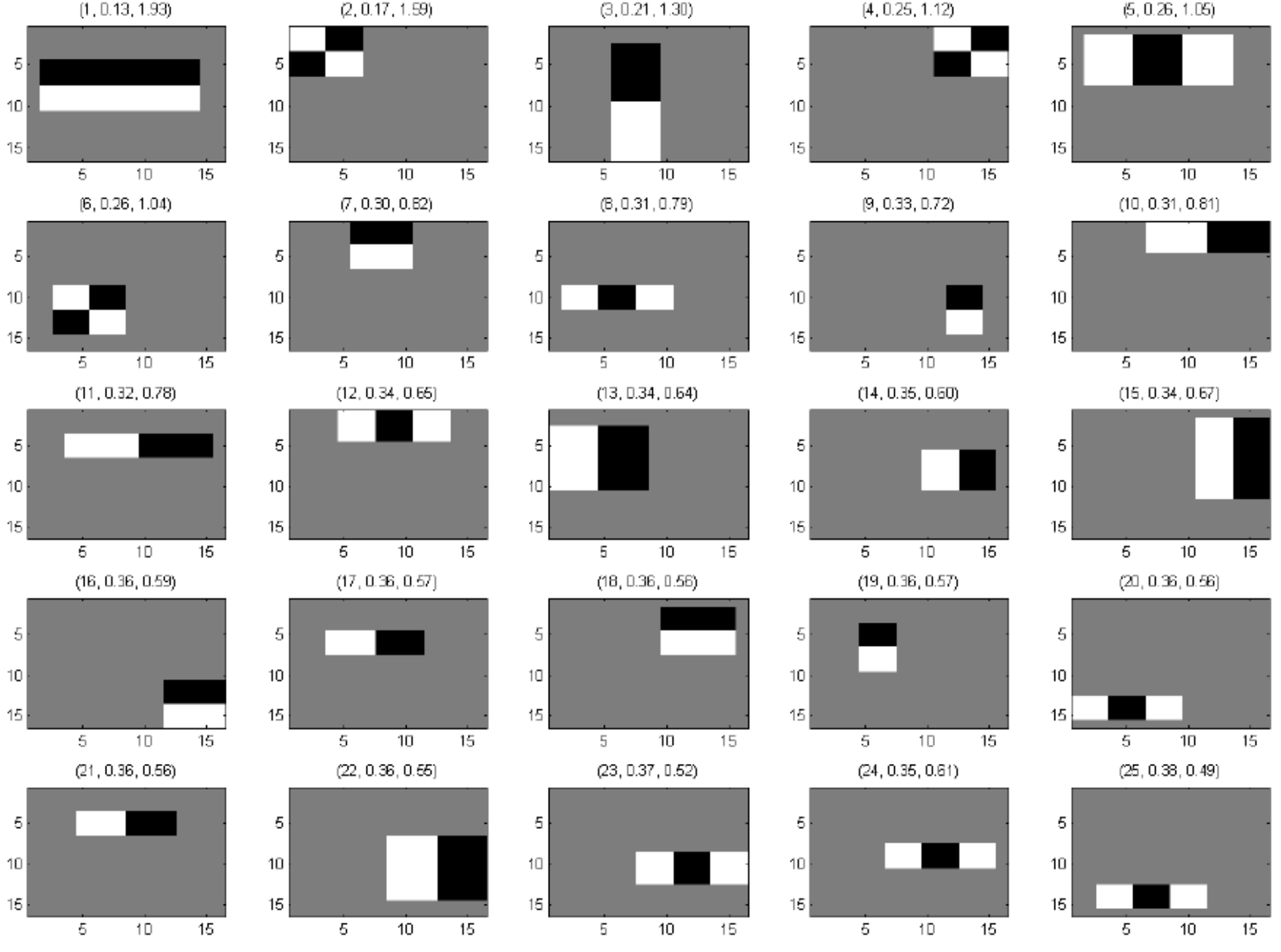


- Odezvu Haarova filtru lze spočítat velmi rychle pomocí integrálního obrázku

$$I(u,v) = \sum_{u'=1}^{u}\sum_{v'=1}^{v} x(u',v')$$

Příklady nalezených slabých klasifikátorů

Po prvních 25 iteracích.



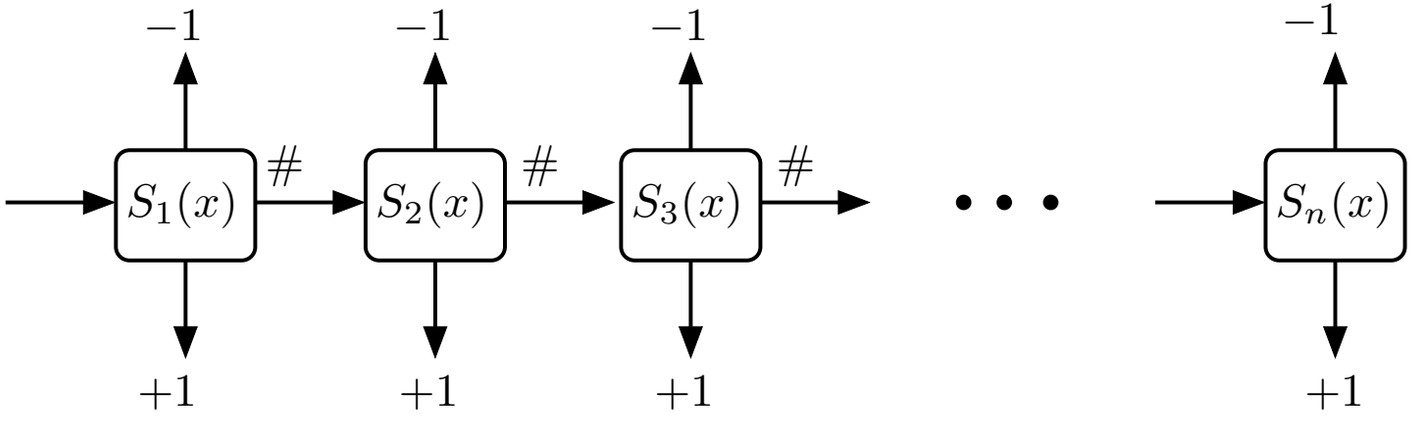
Urychlení detekce pomocí sekvenčního rozhodování

- Počet slabých klasifikátorů je $n \approx 1000$, což může být stále časově náročné pokud používáme celý klasifikátor

$$h(x) = \begin{cases} +1 & \text{pokud} \quad f(x) \geq 0 \\ -1 & \text{pokud} \quad f(x) < 0 \end{cases} \qquad f(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$$

- K rozhodnutí o jasných příkladech tváří/netváří stačí jednoduší pravidlo.
- Detektor lze urychlit použitím sekvenčního rozhodovacího pravidla

$$S_t(x) = \begin{cases} +1 & f_t(x) \geq \theta_t^A \\ -1 & f_t(x) \leq \theta_t^B \\ \# & \theta_t^B < f_t(x) < \theta_t^A \end{cases} \qquad f_t(x) = \sum_{i=1}^{t} \alpha_i h_i(x)$$

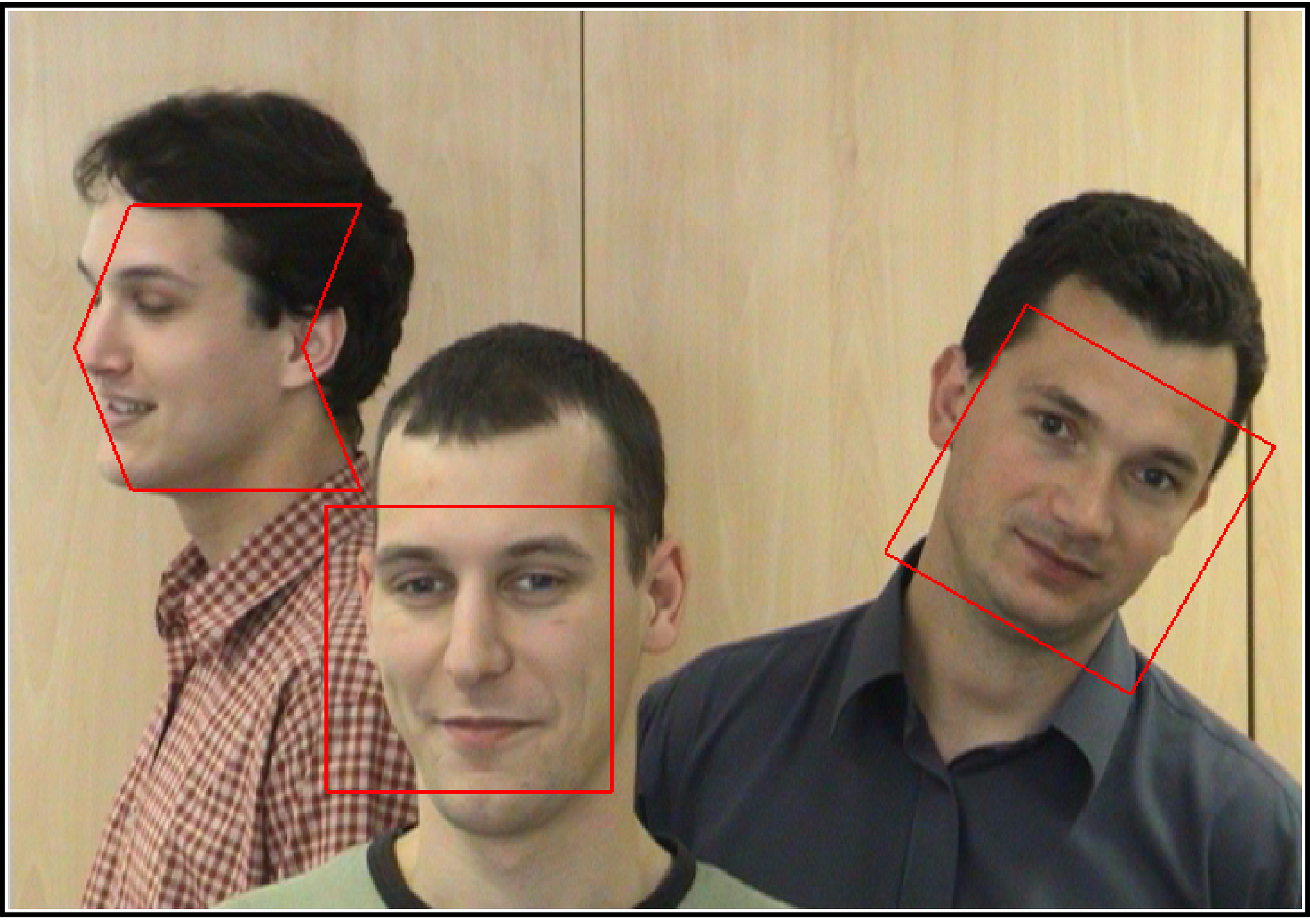


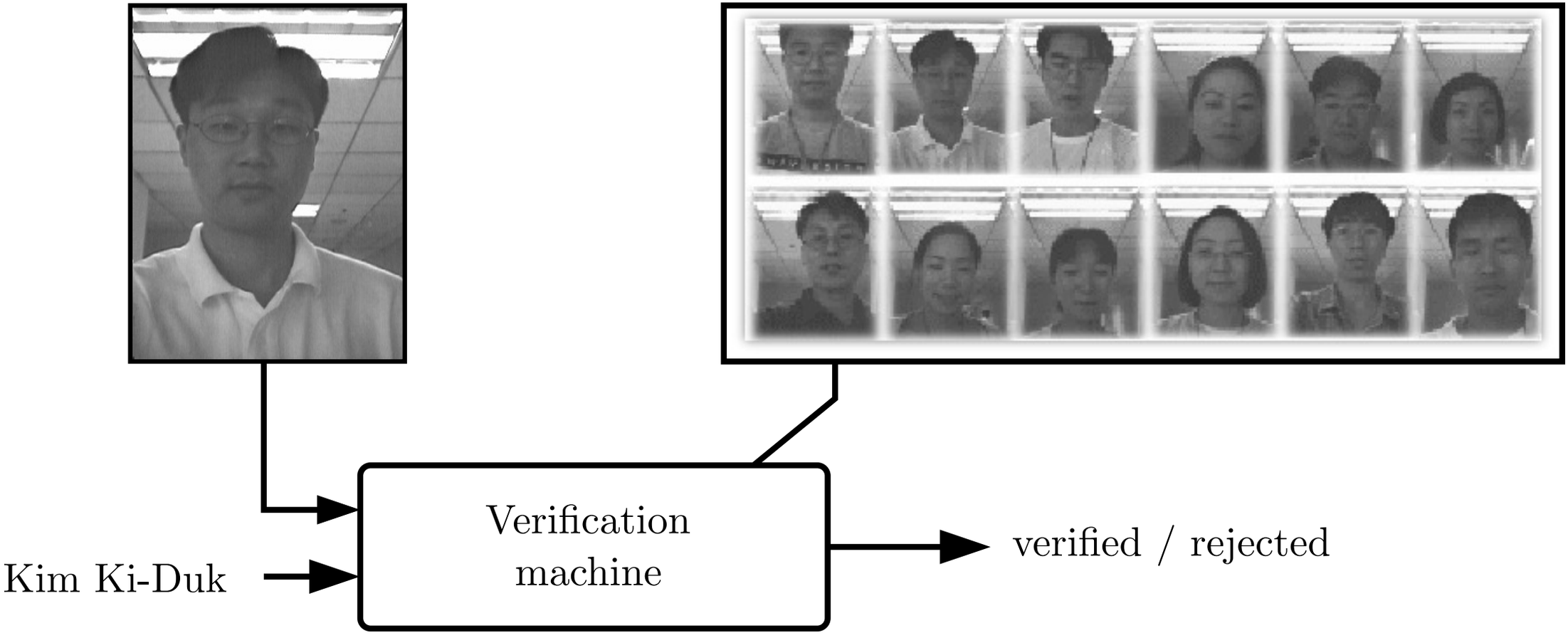
Příklad: Detektor tváří firmy Eyedea Recognition LtD.

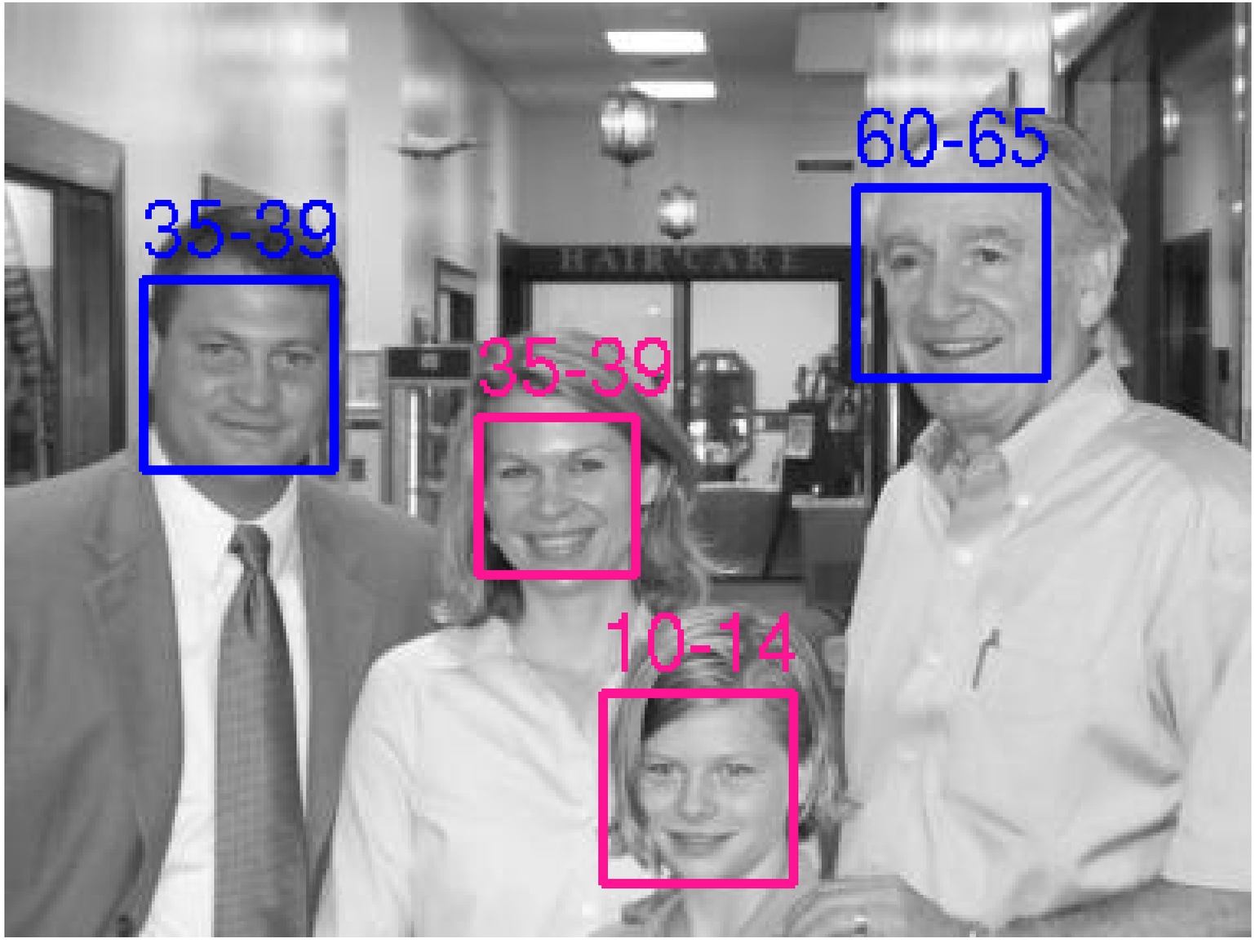
- Trénovací množina vytvořena tak, aby pokryla co nejvíce variance (rasa, osvětlení, výraz tváře, pozadí obrázku...).
  - Pozitivní příklady (tváří): více než $500,000$
    - Synteticky generované z cca $60,000$ tváří aplikováním transformací, které nemění třídu obrázku - malá změna rotace, změna měřítka, posunutí.
    - Pozitivní příklady vyžadují manuální anotaci.
  - Negativní příklady (netváře): přibližně 4×10^9
    - Negativní příklady se generují z množiny obrázků neobsahujících žádné tváře.
    - Není potřeba anotace.
- Rychlost detektoru závisí na mnoha parametrech (minimální velikost detekované tváře, velikost vstupního obrázku, krok posunutí, rotace ...)
  - Například pro rozlišení $640 \times 480\,\mathrm{px}$ a minimální velikosti tváře $24 \times 24\,\mathrm{px}$ zpracuje detektor cca $10 - 15$ smímků za vteřinu.

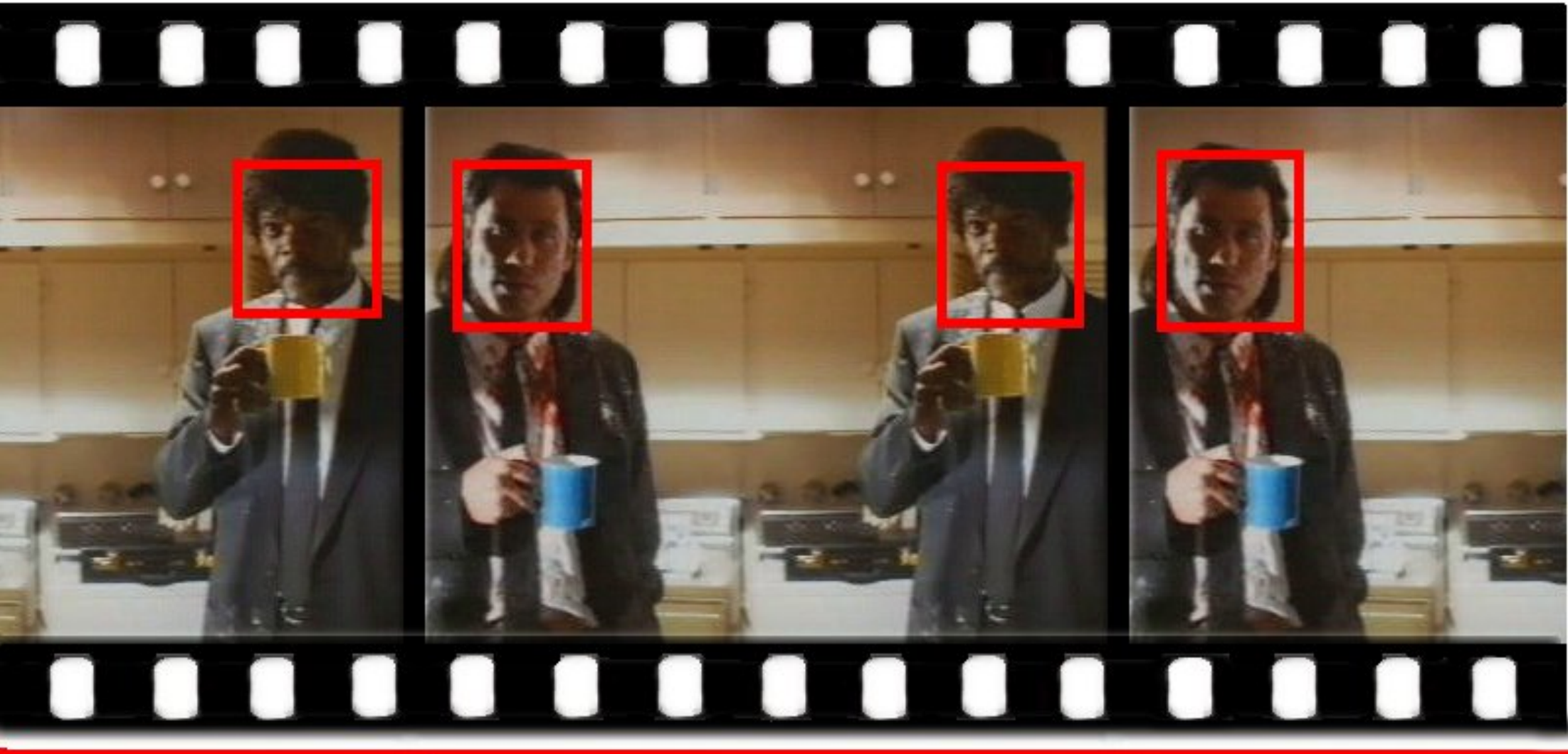
Konec

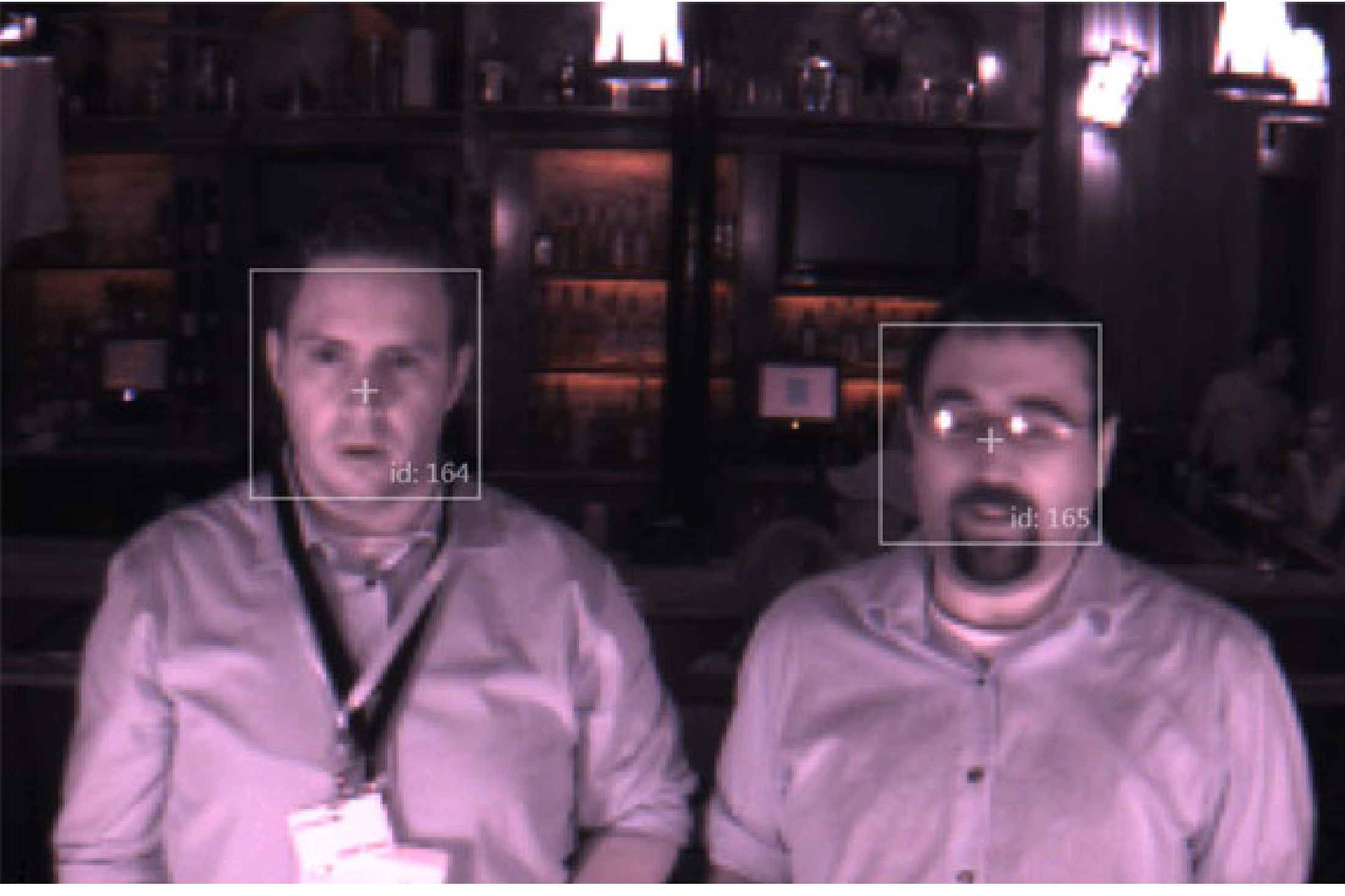

m p

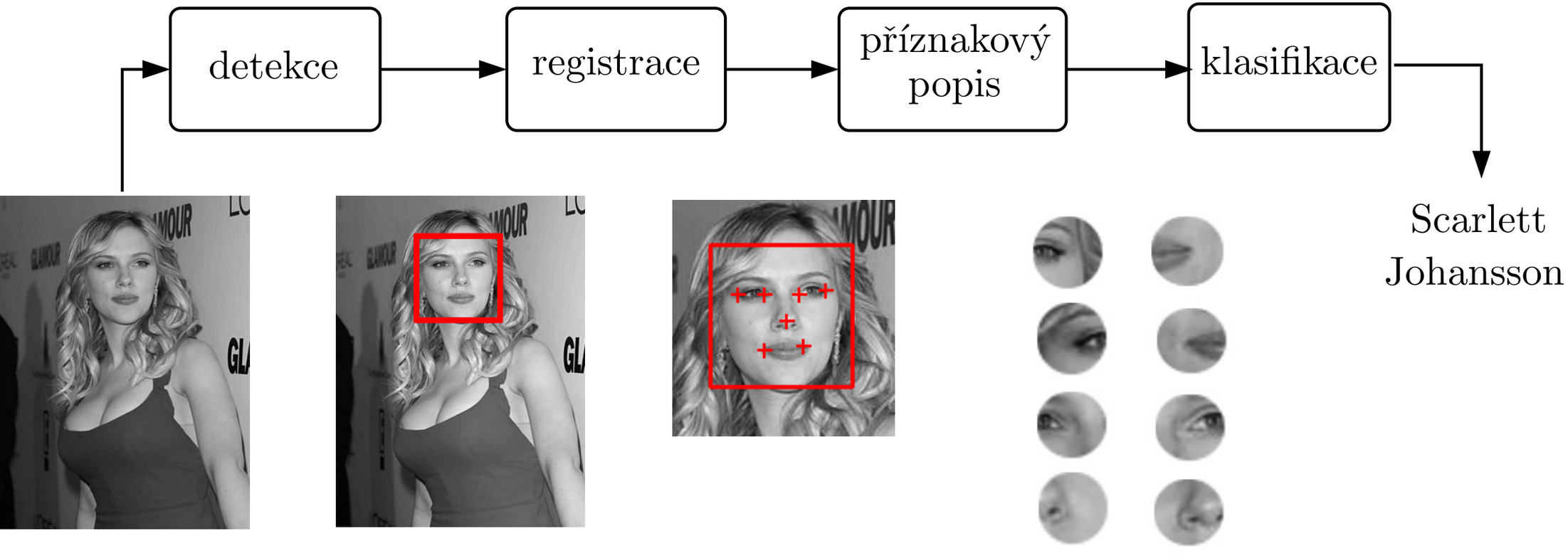


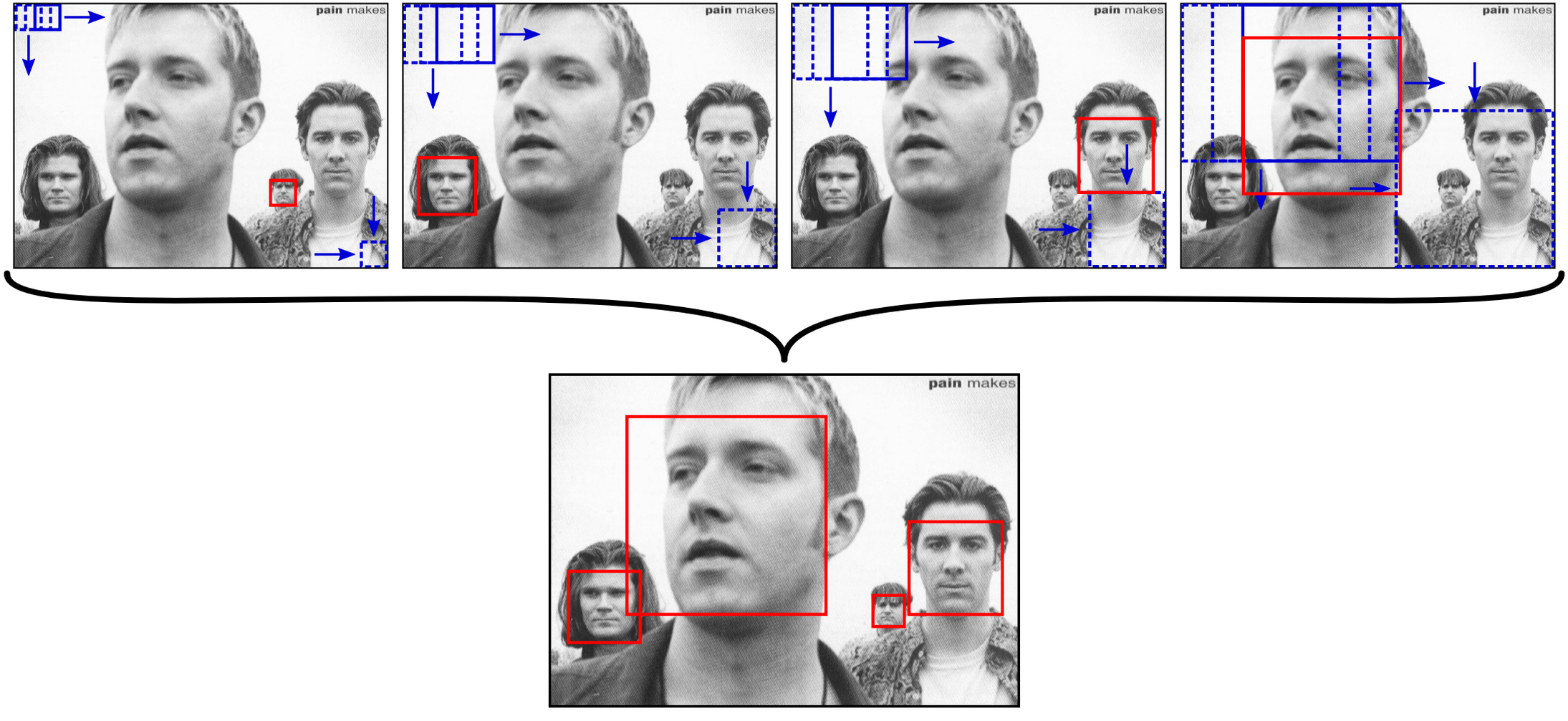

Verification
machine
Kim Ki-Duk
verified / rejected


60-65
35-39
35-39
10-14



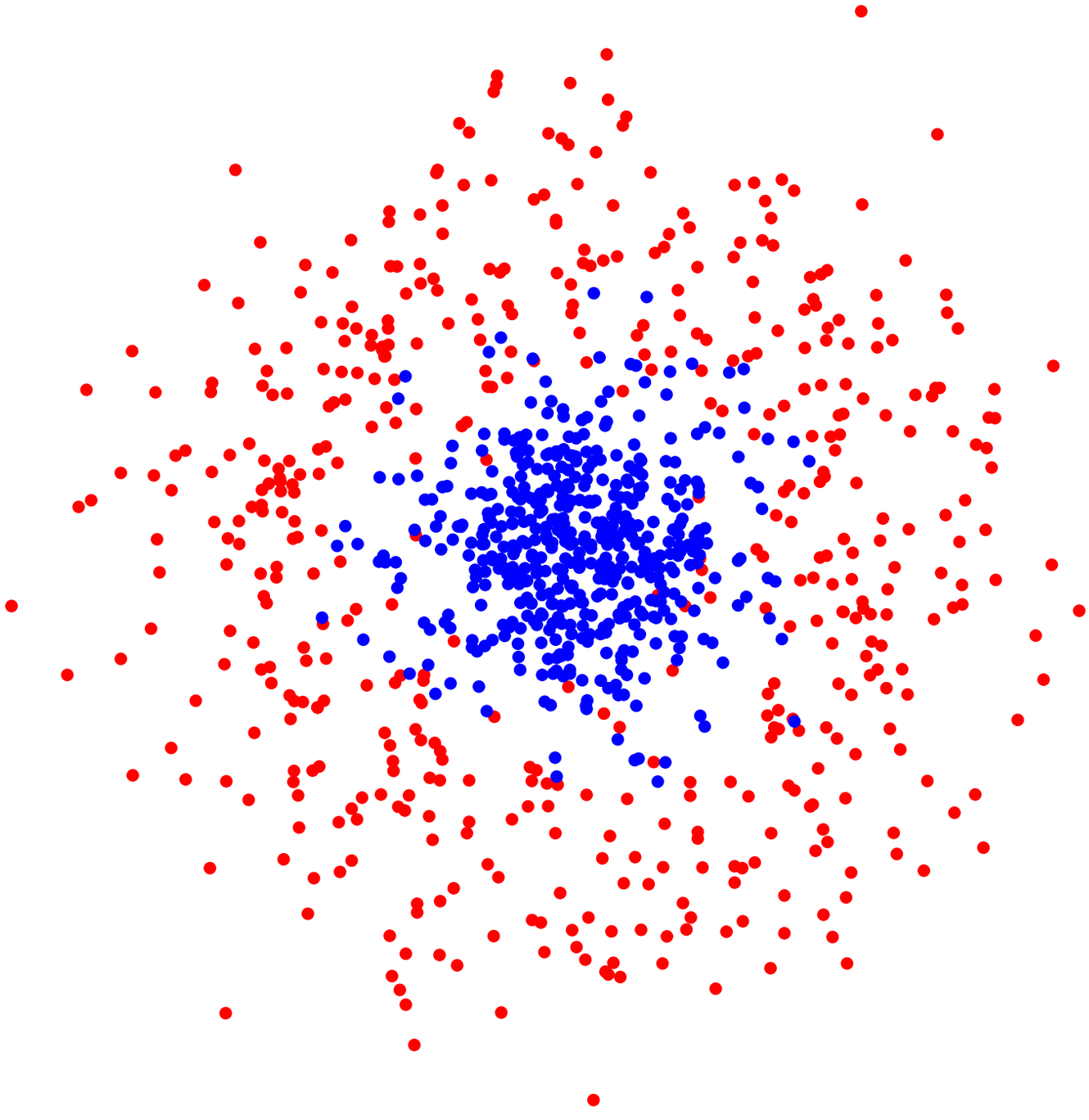

id: 164
id: 165

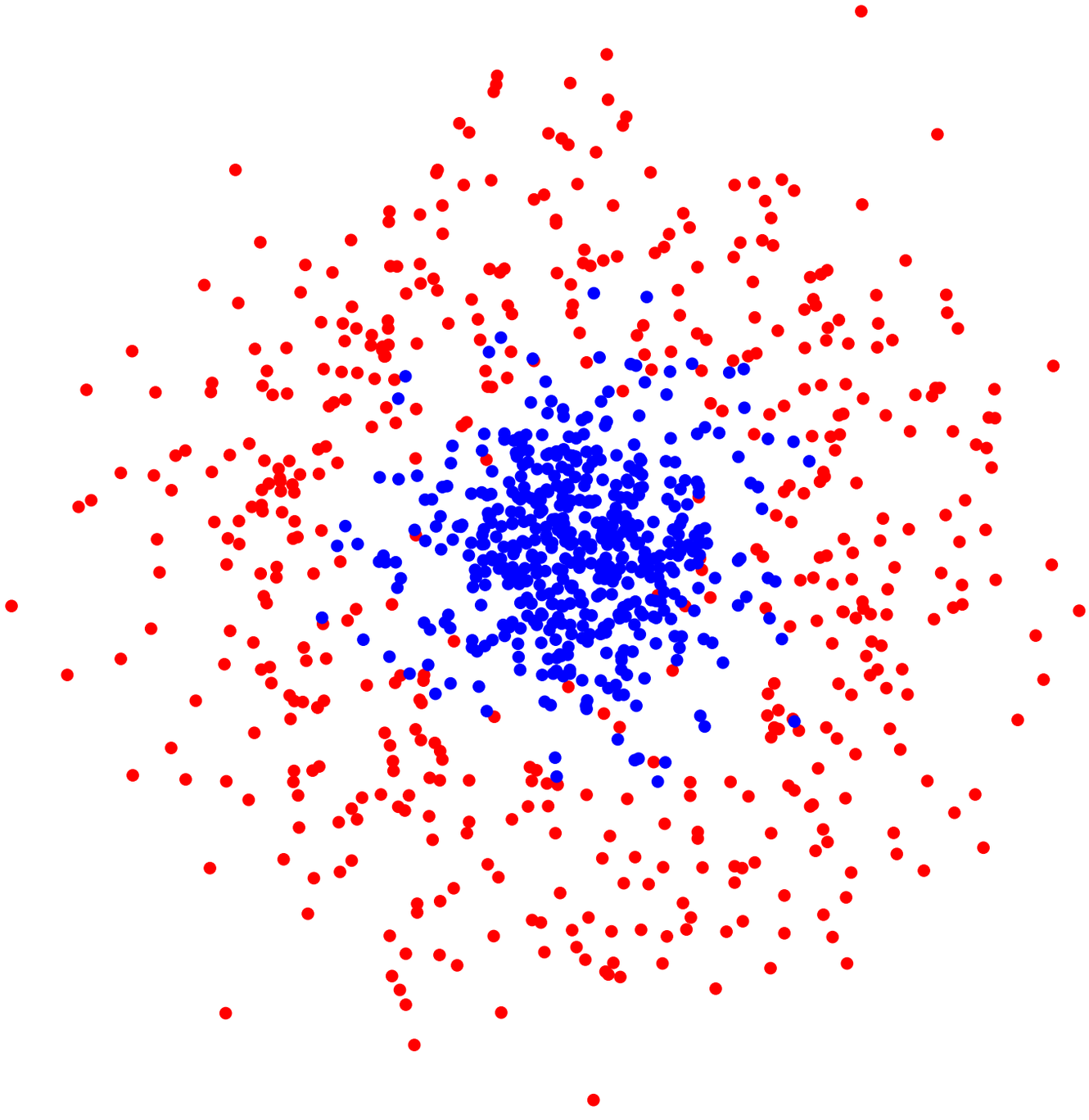

detekce
registrace
příznakový popis
klasifikace
Scarlett Johansson

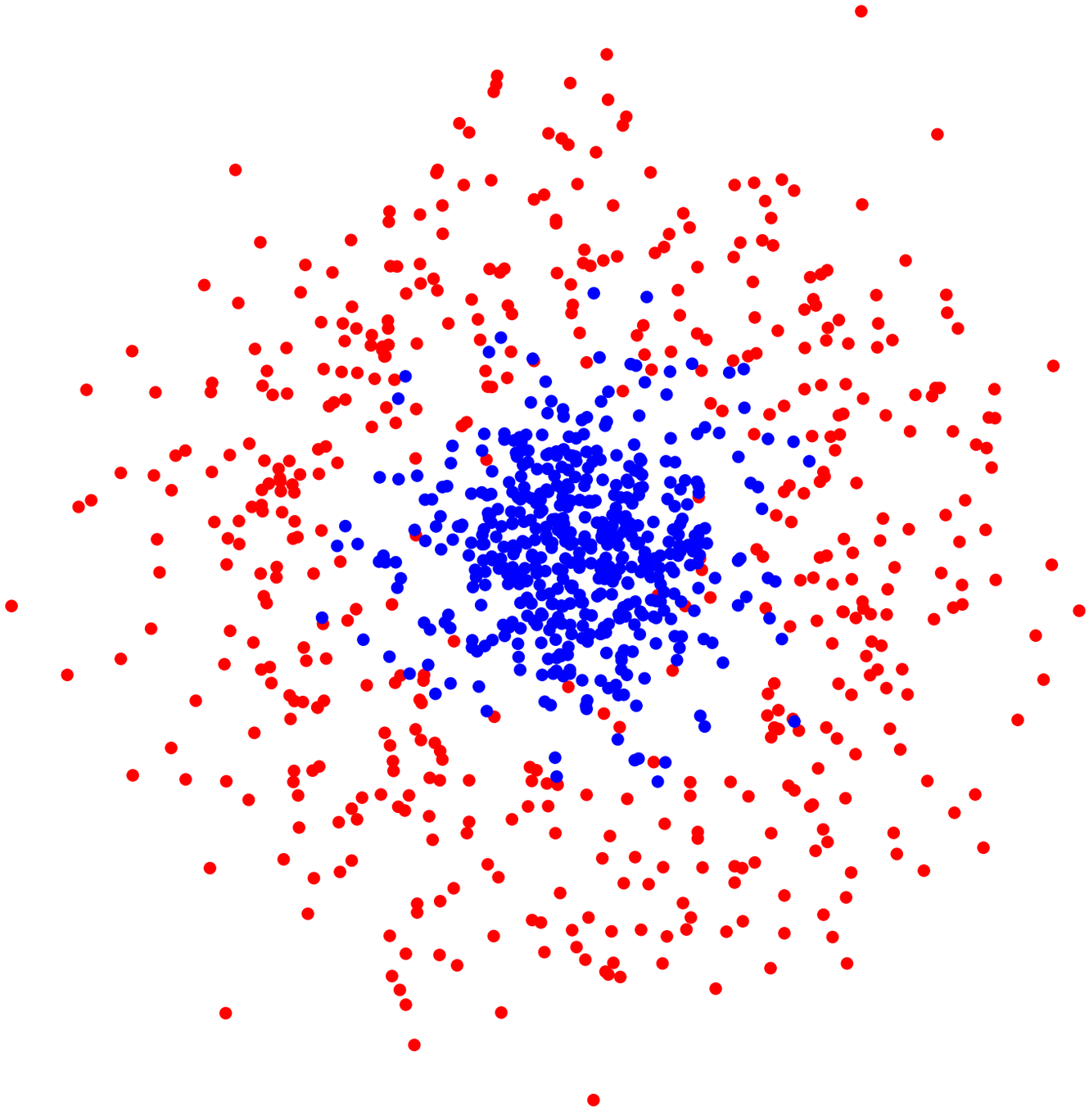

pain makes
pain makes
pain makes
pain makes
pain makes

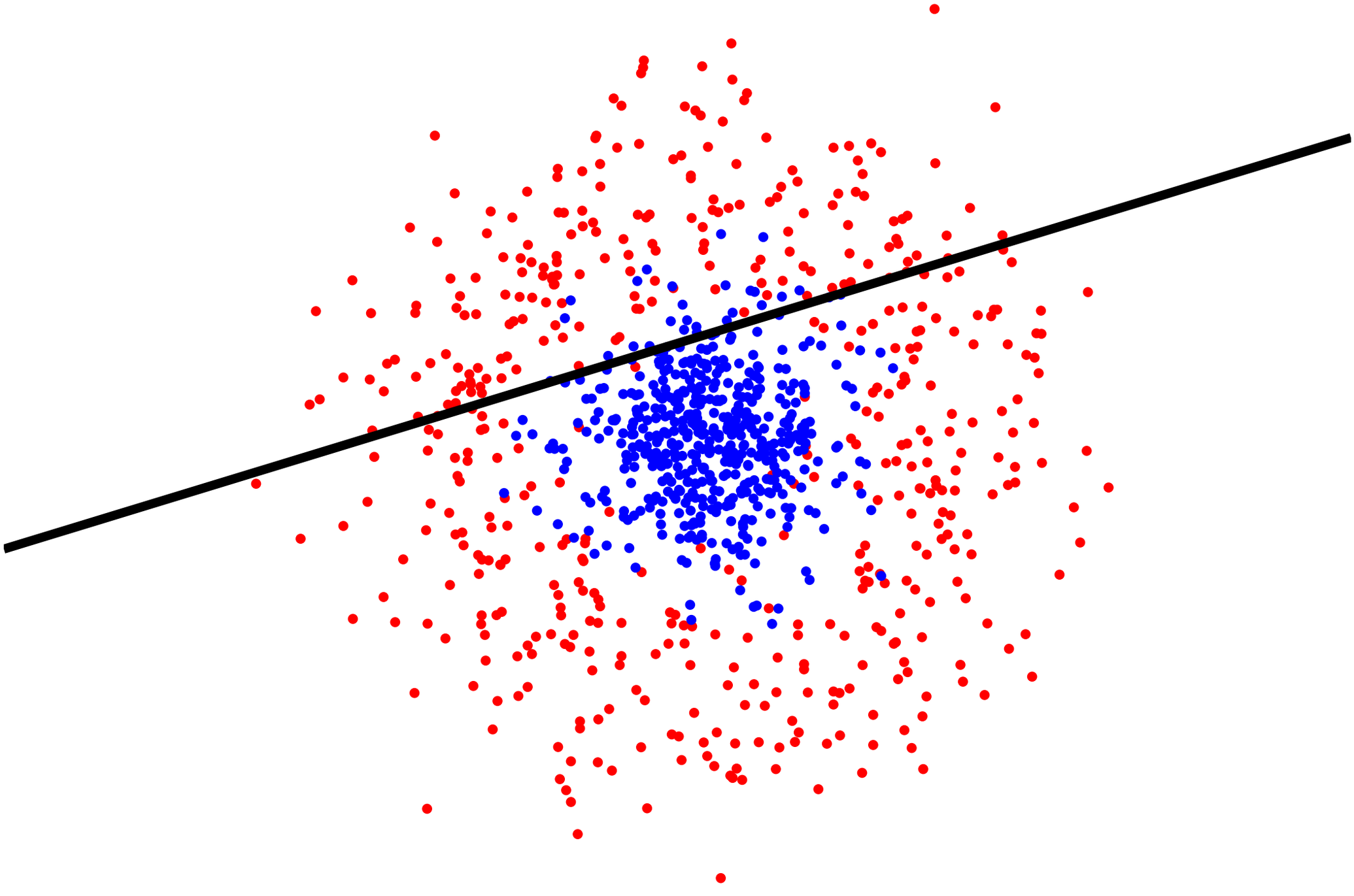
FACE

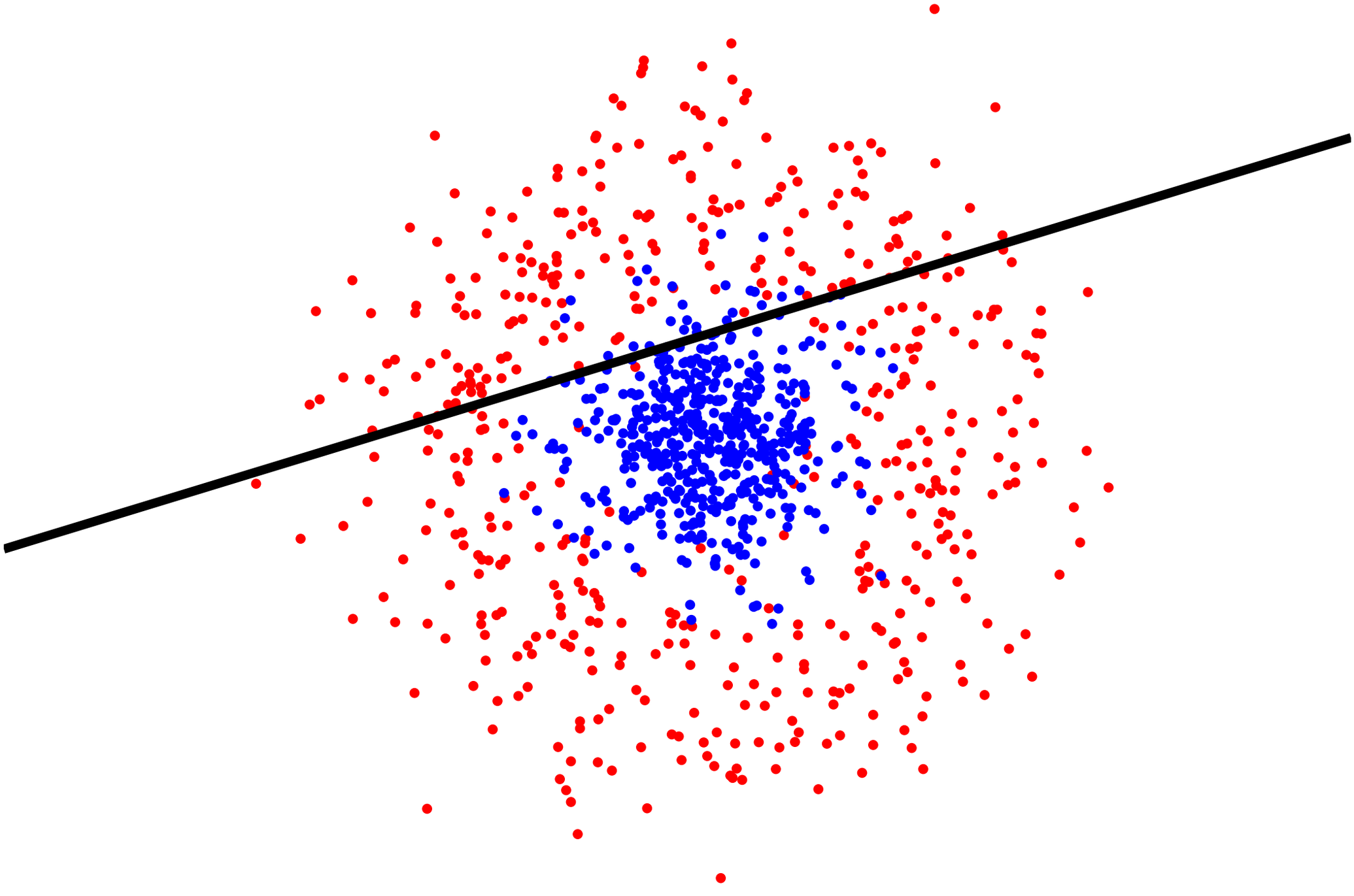
NONFACE

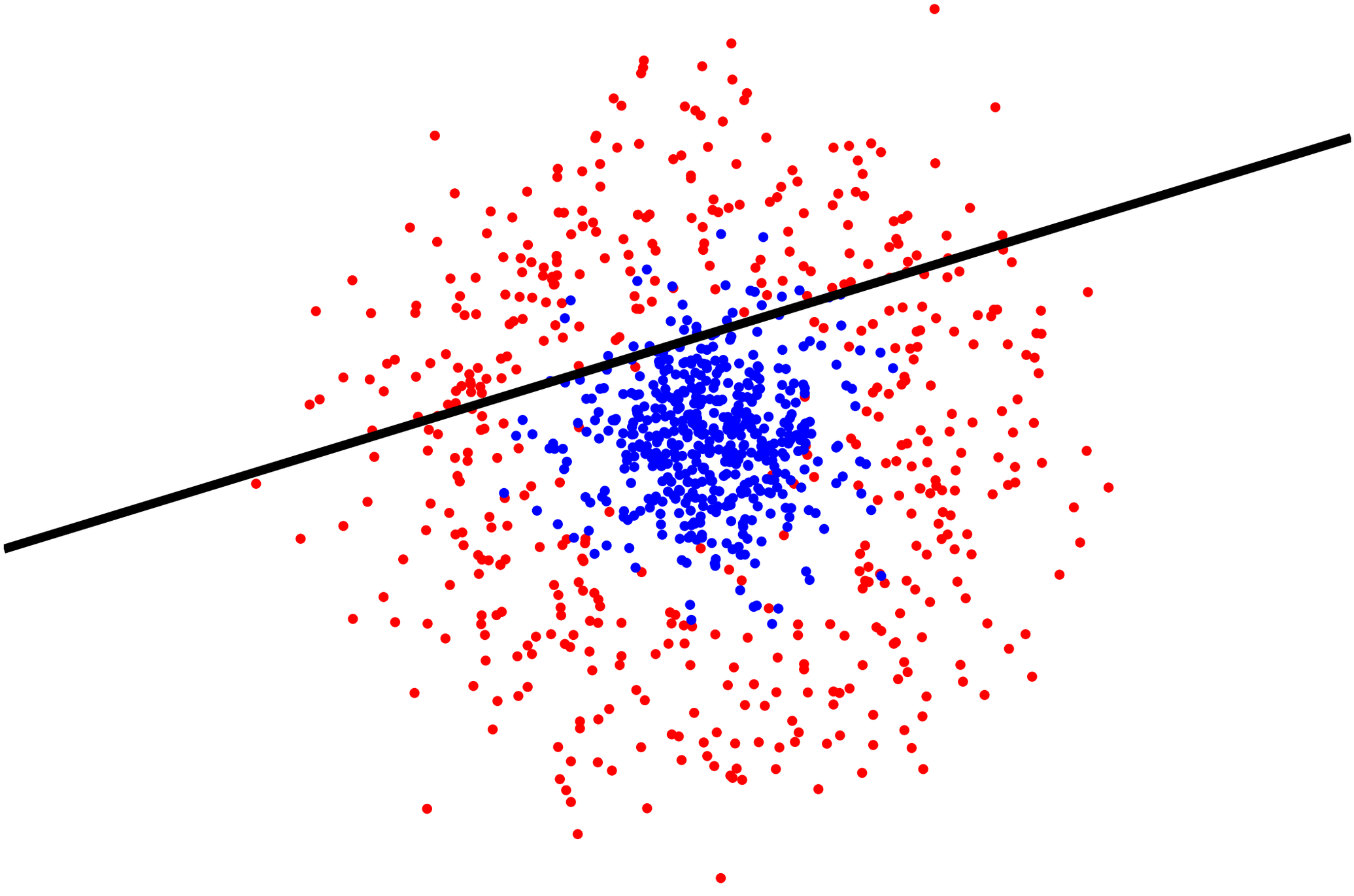


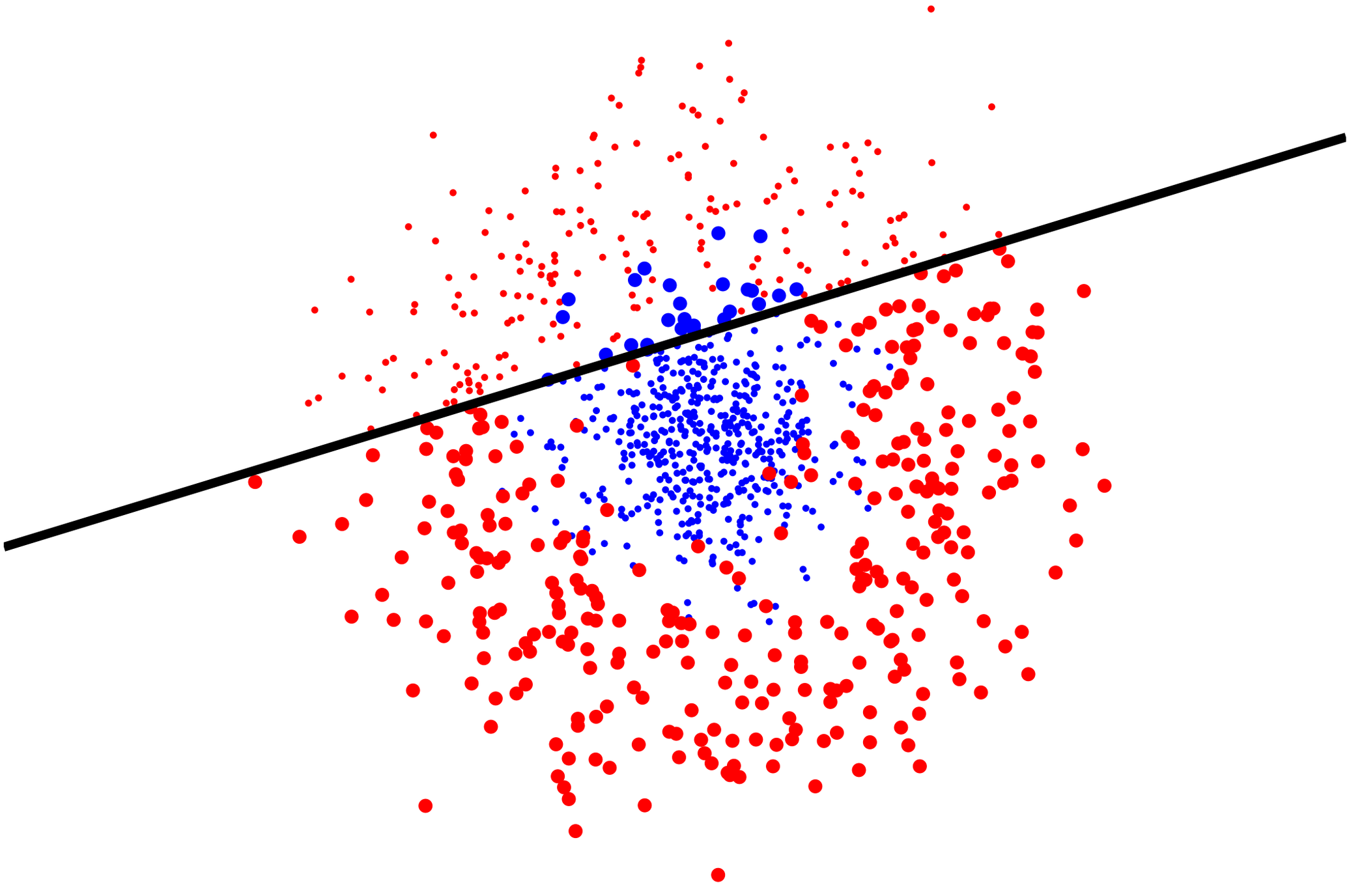


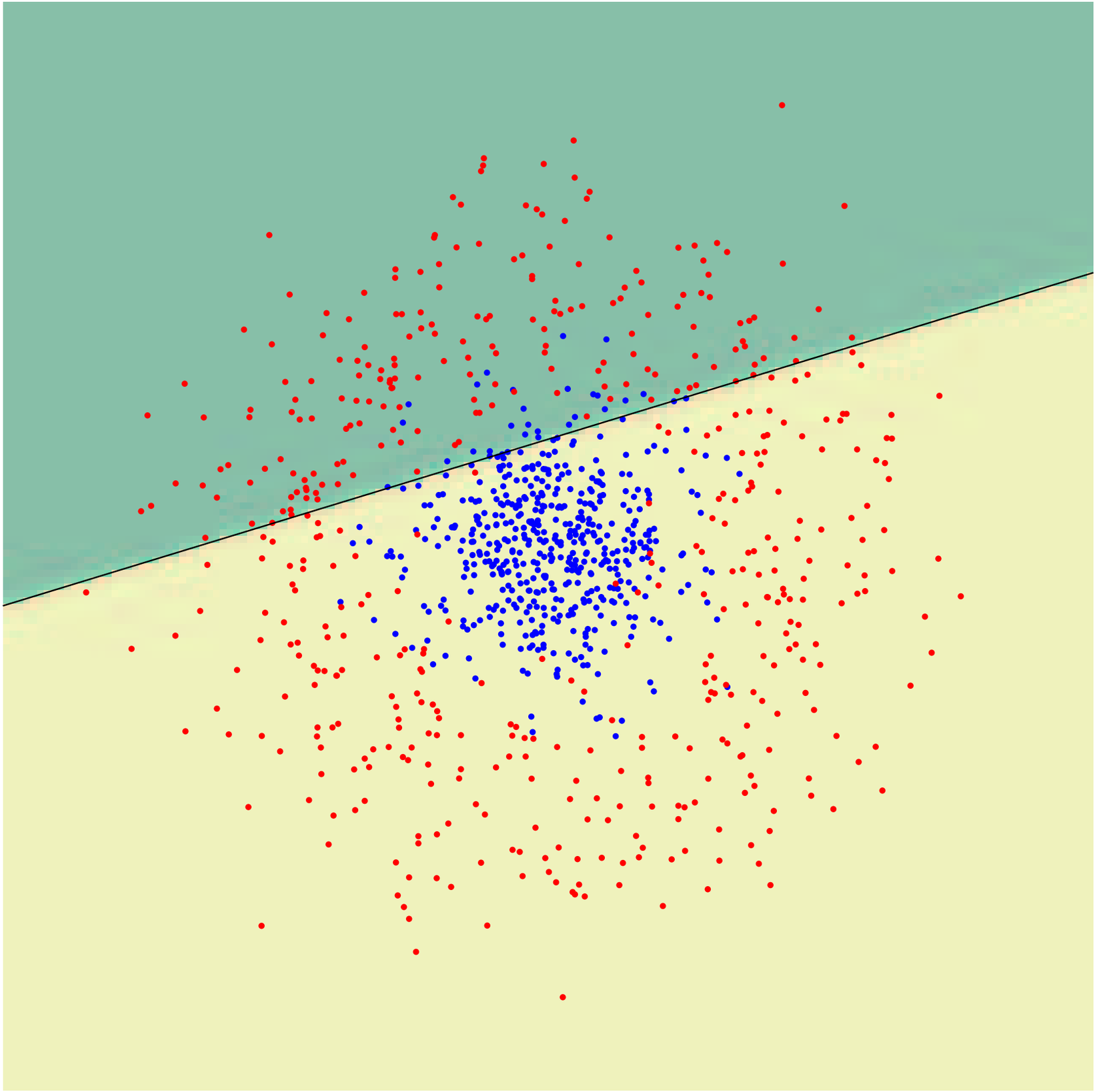


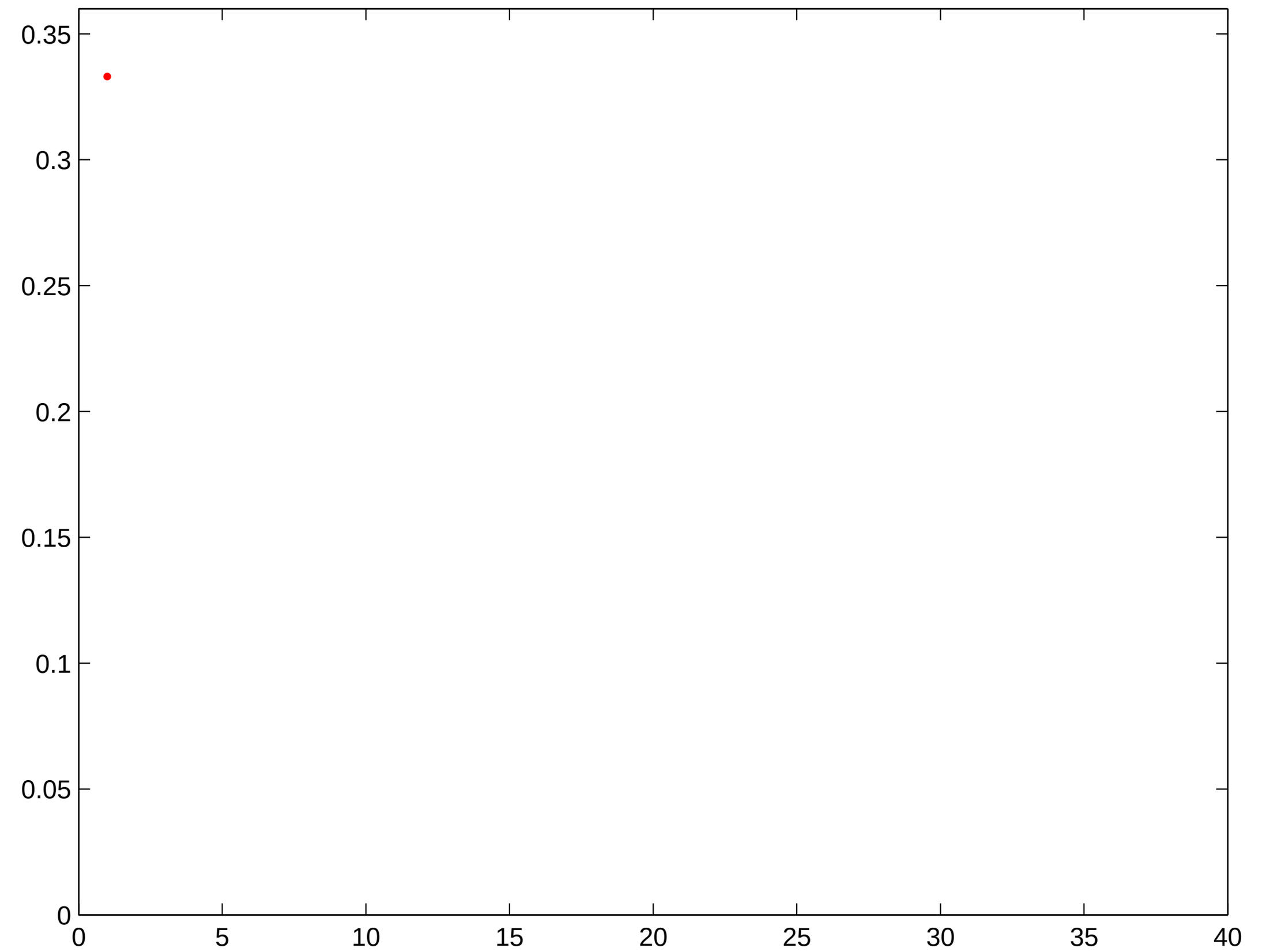


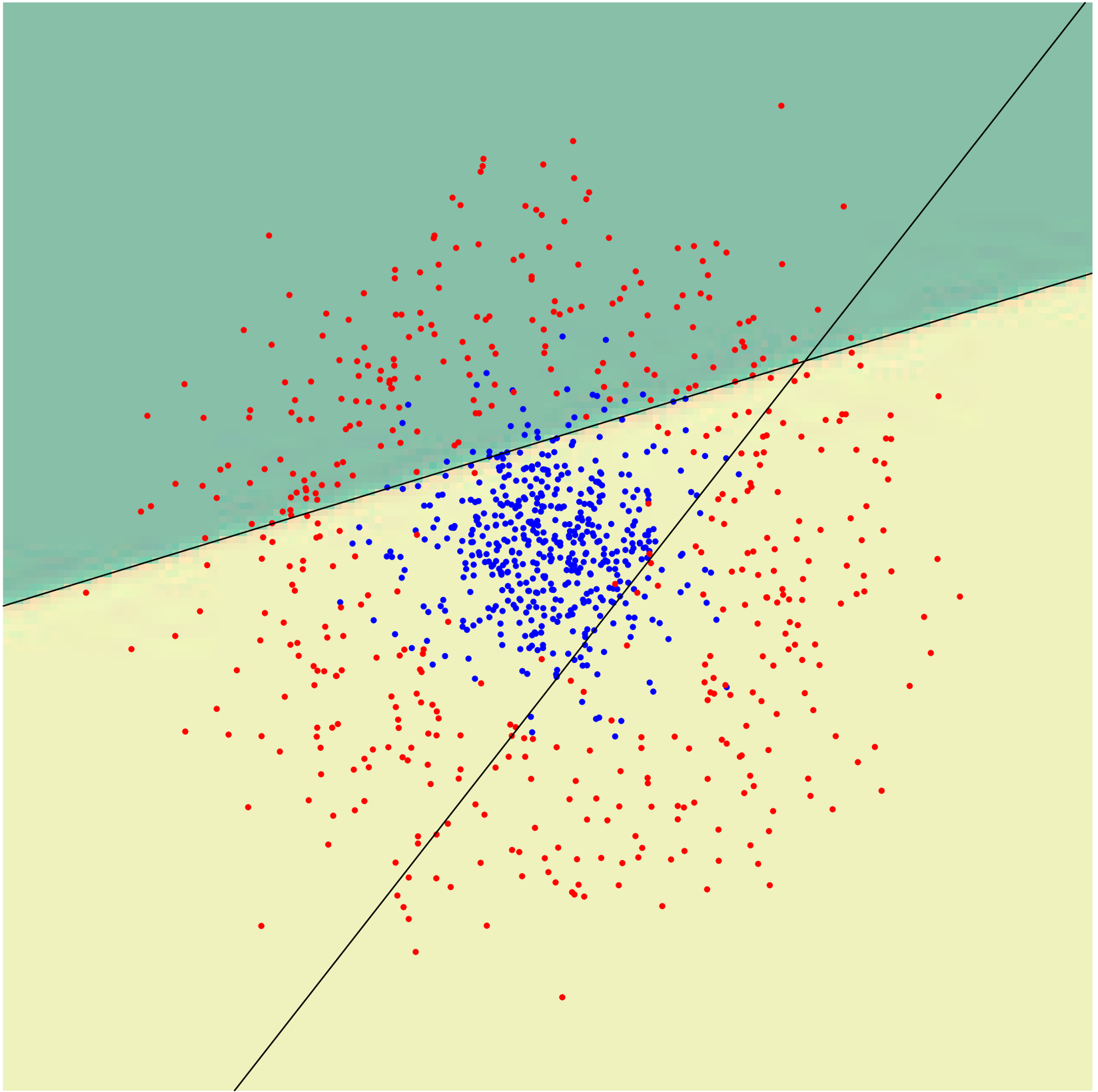


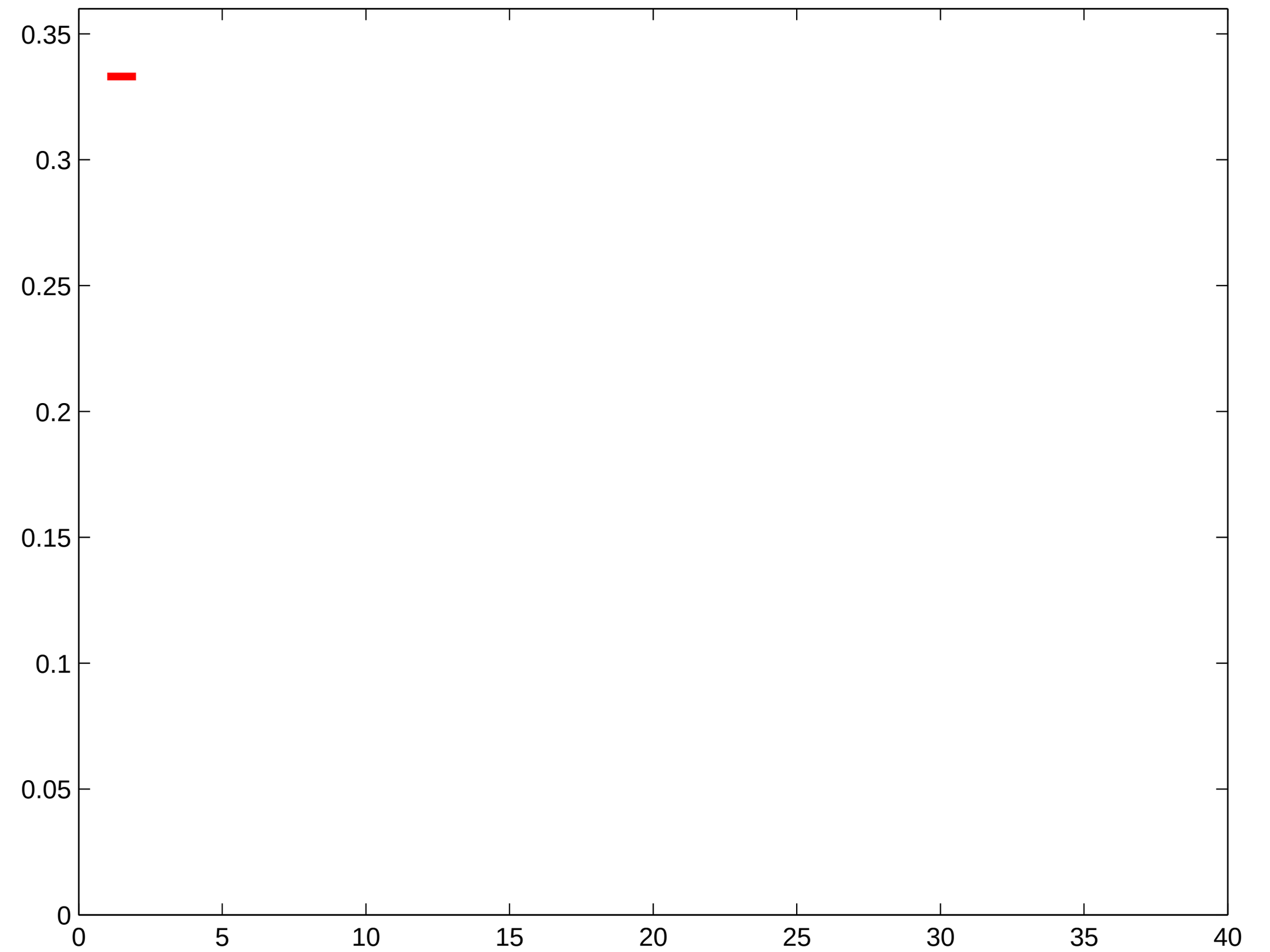


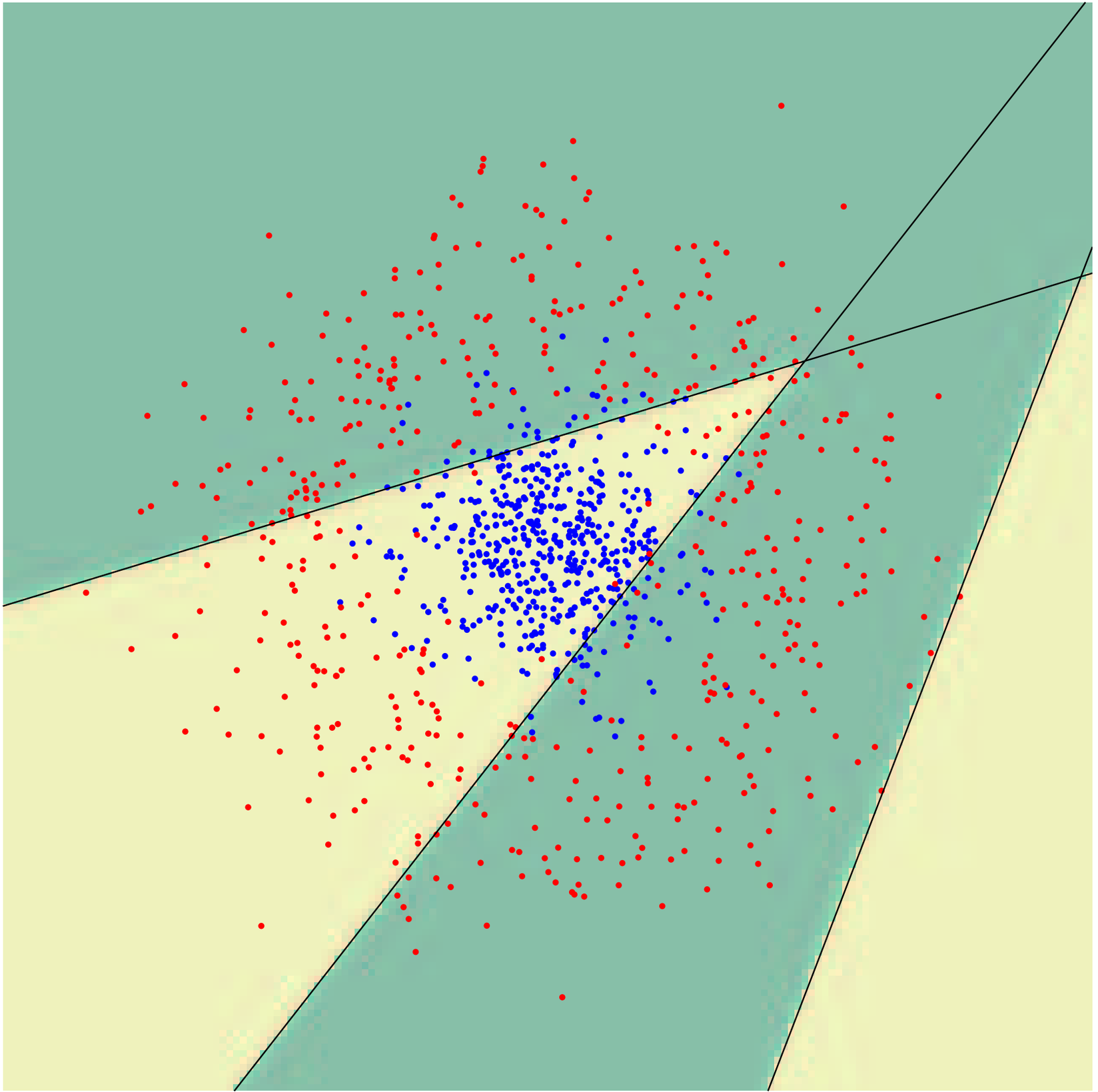


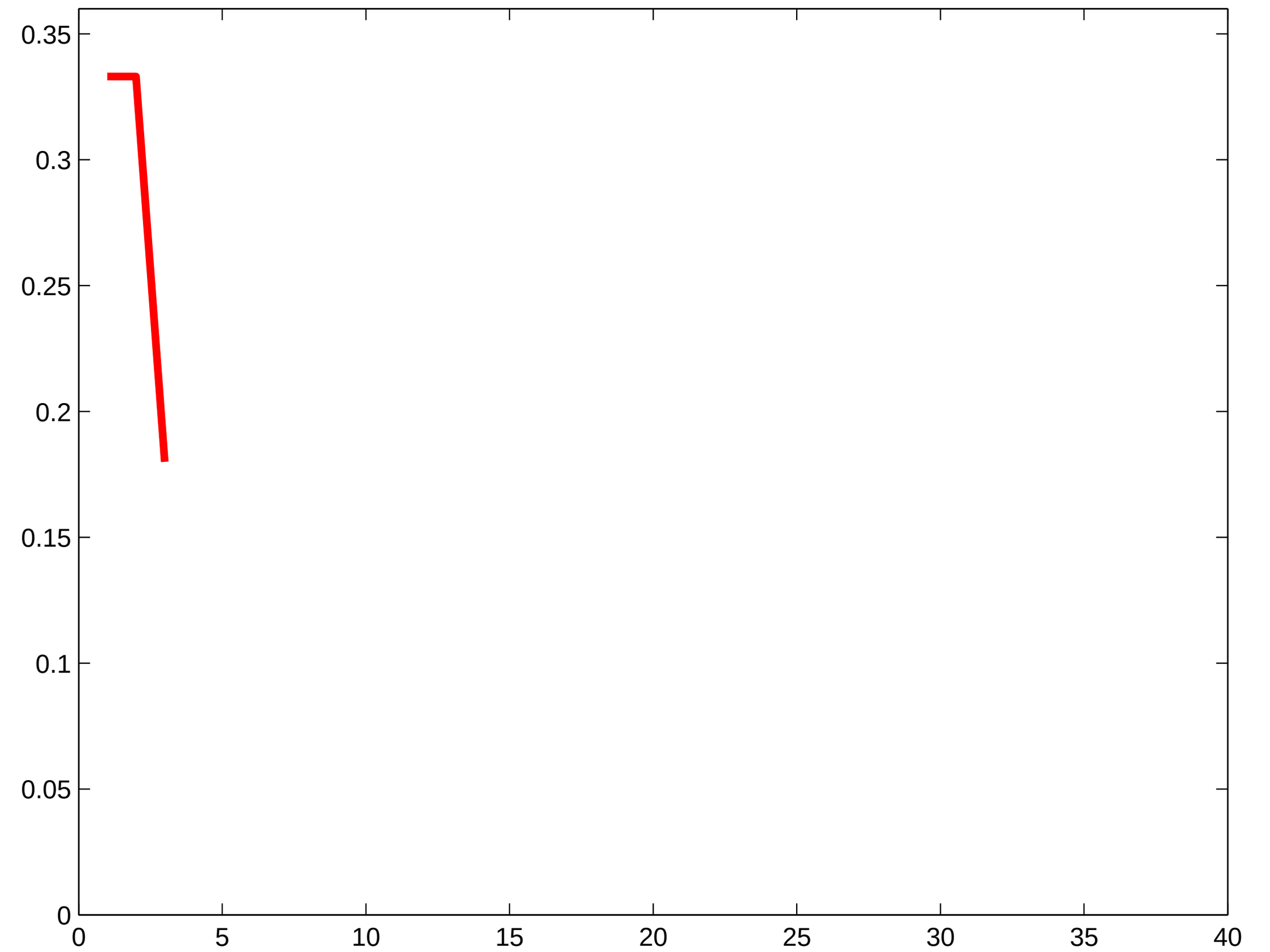


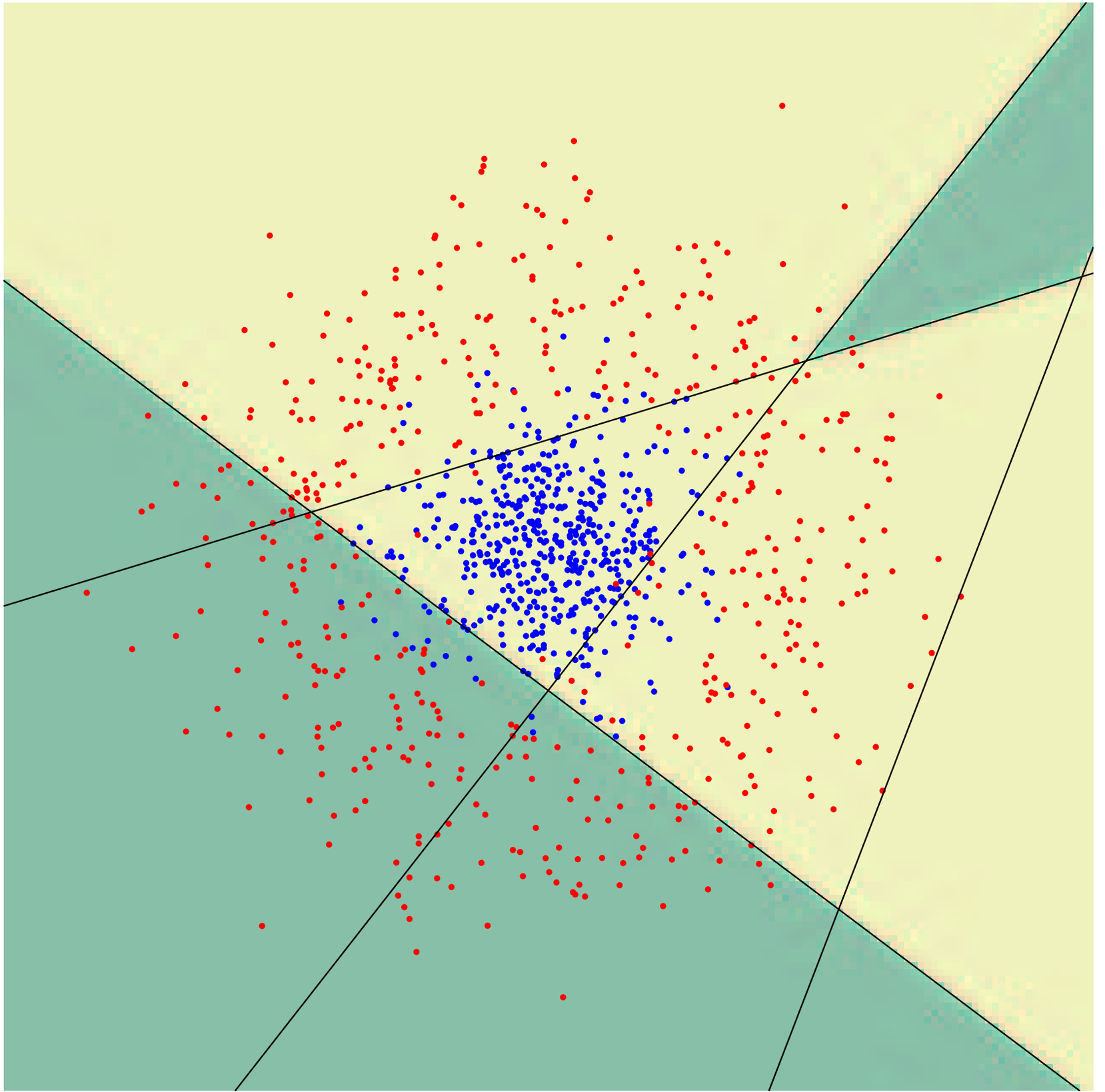

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

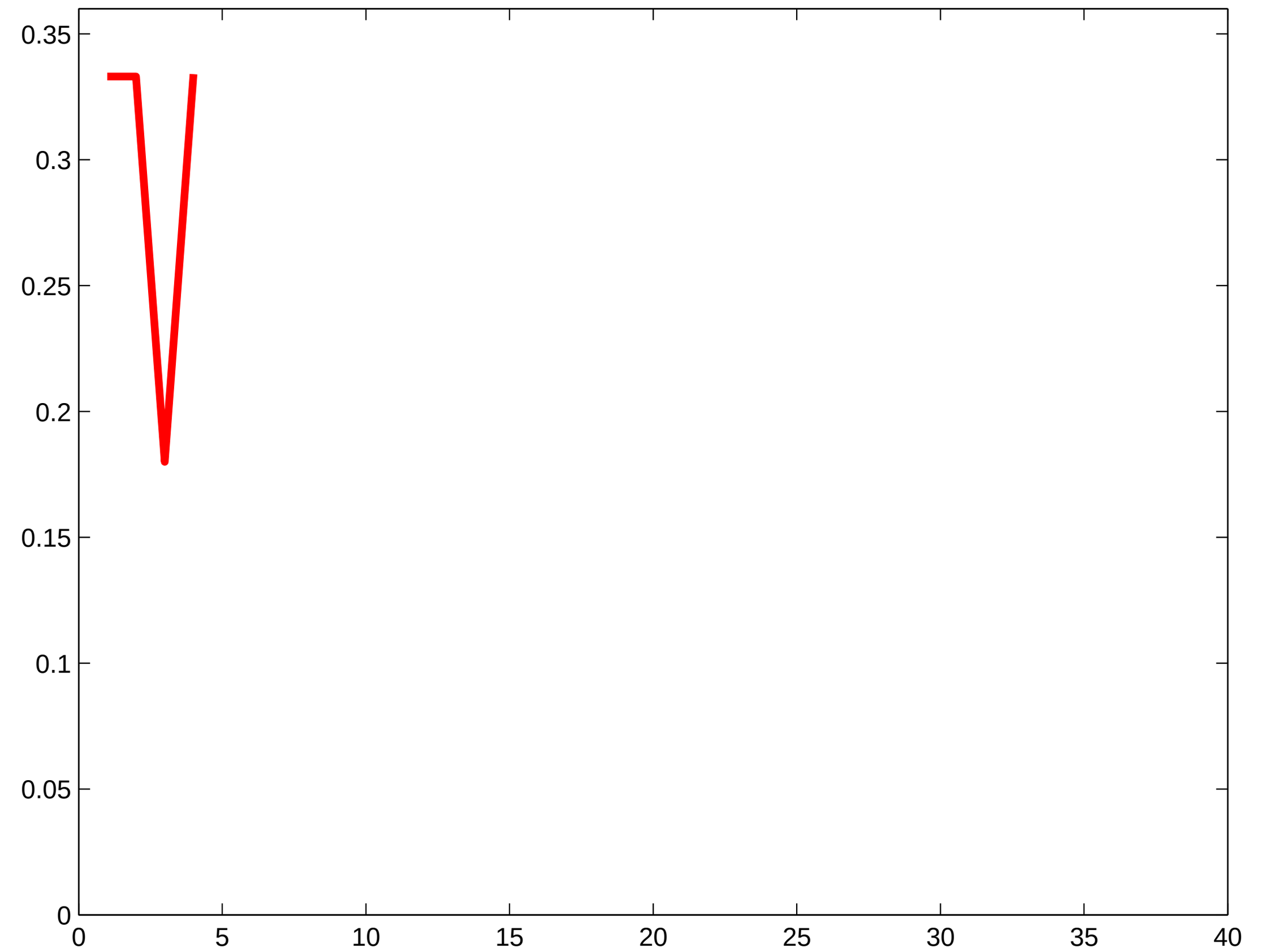


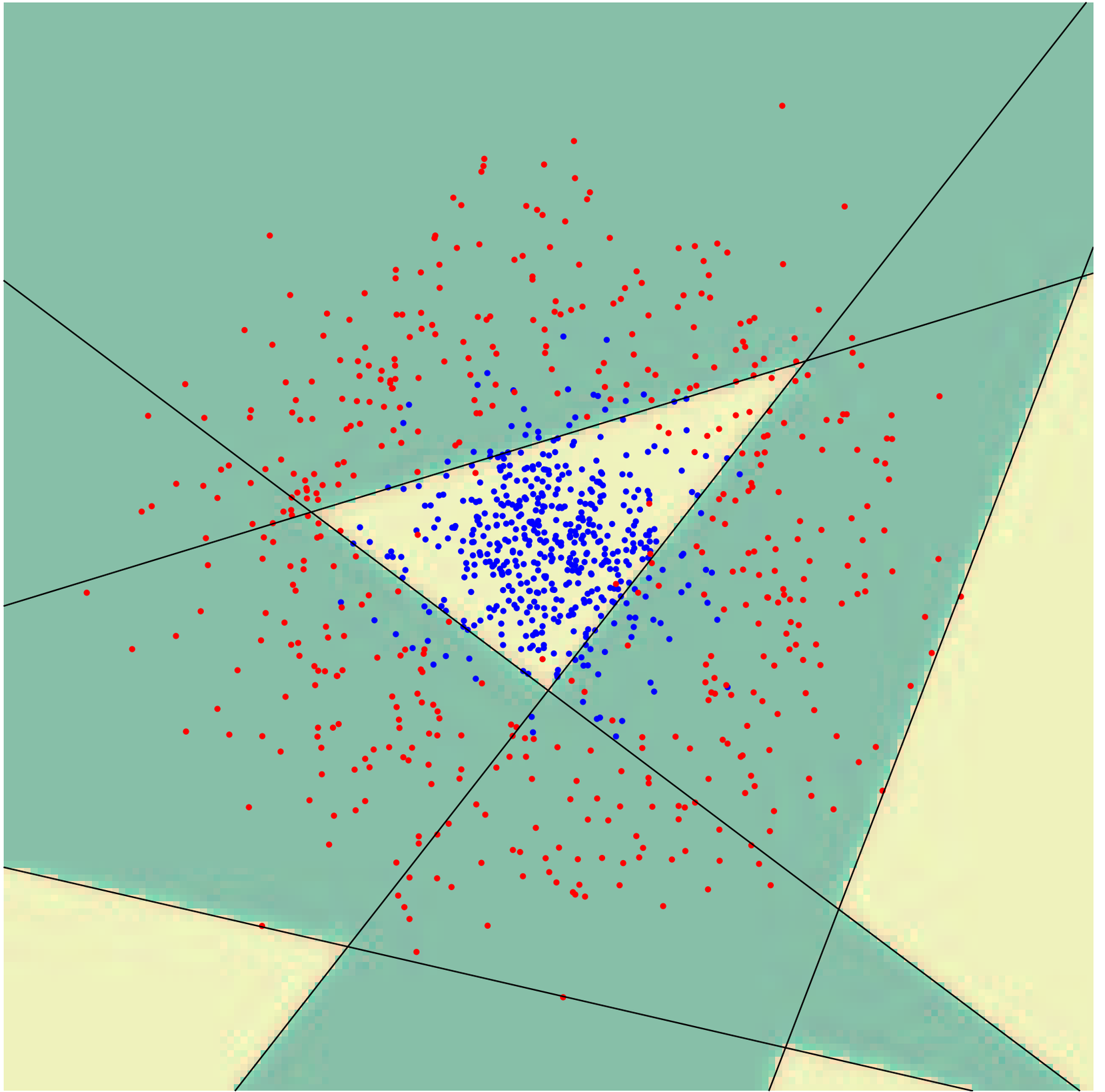

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

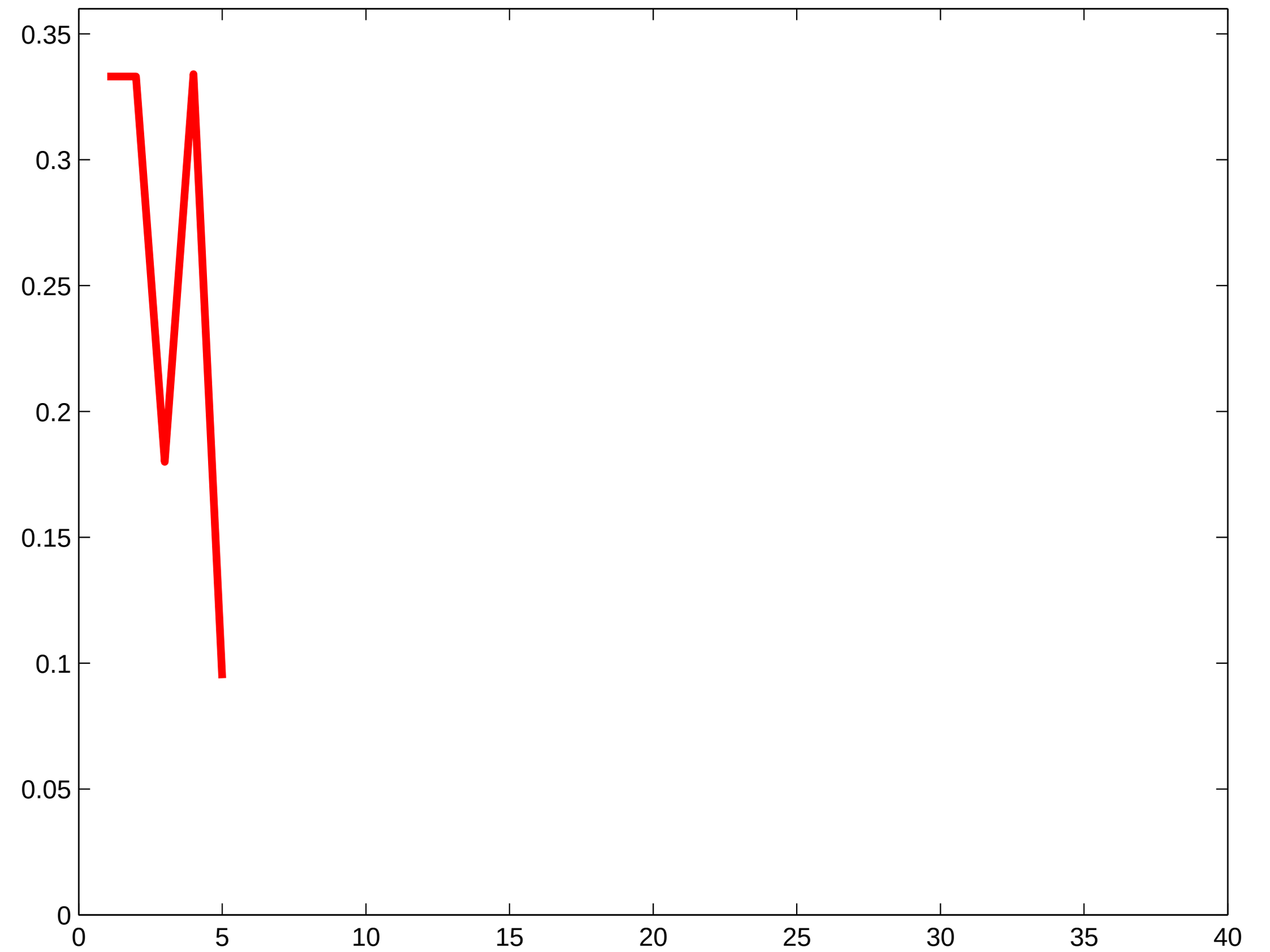


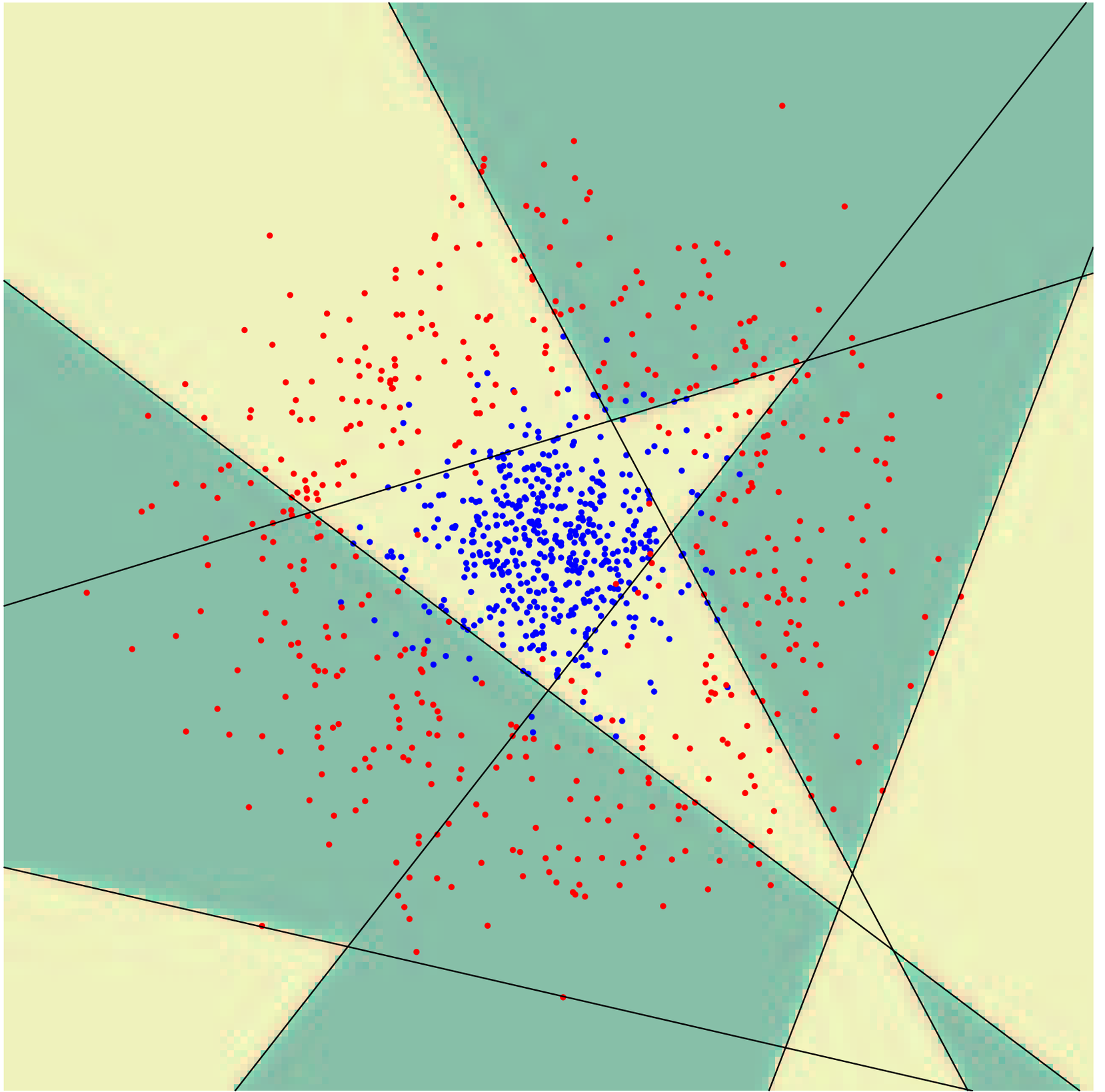

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

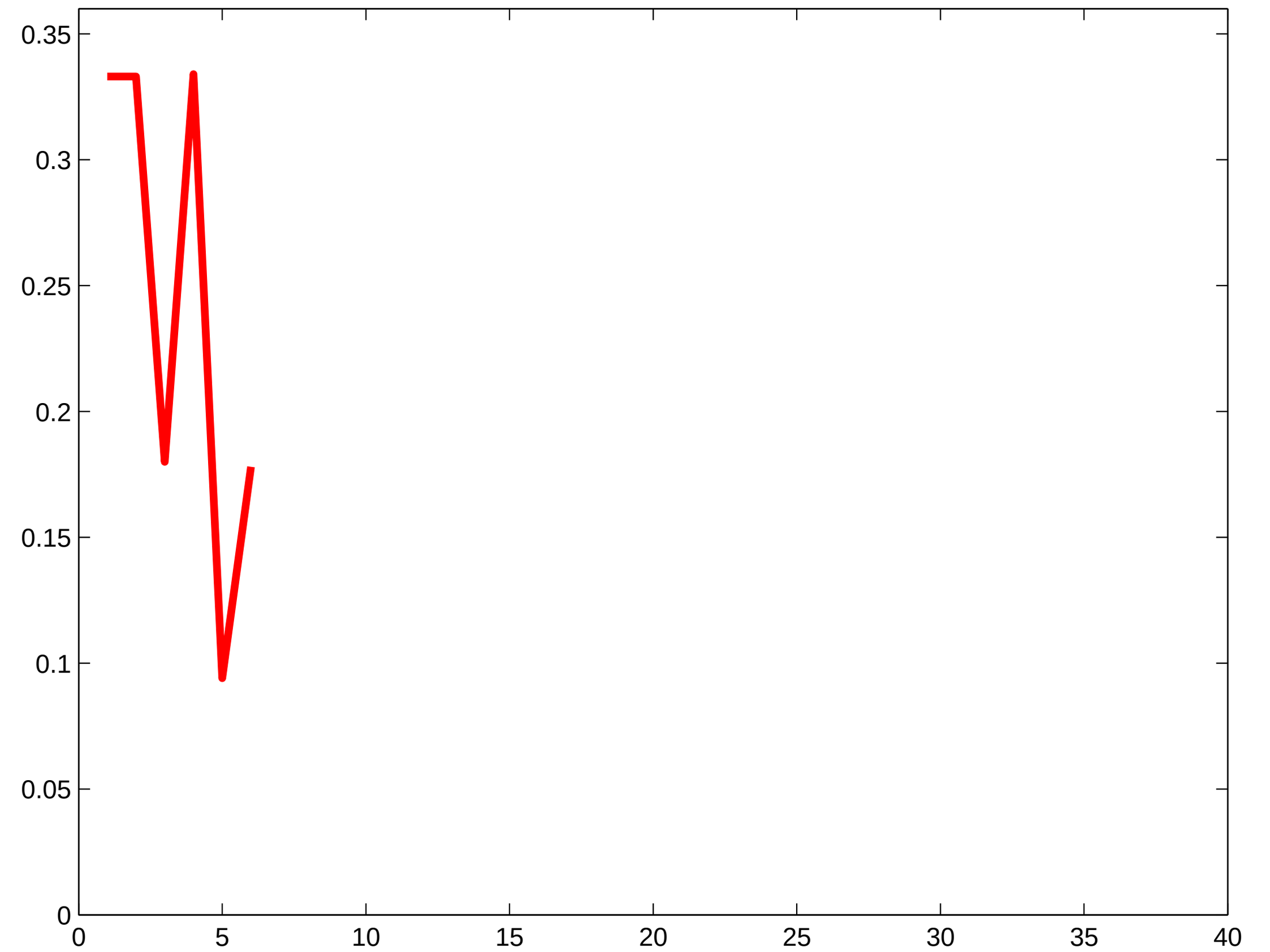


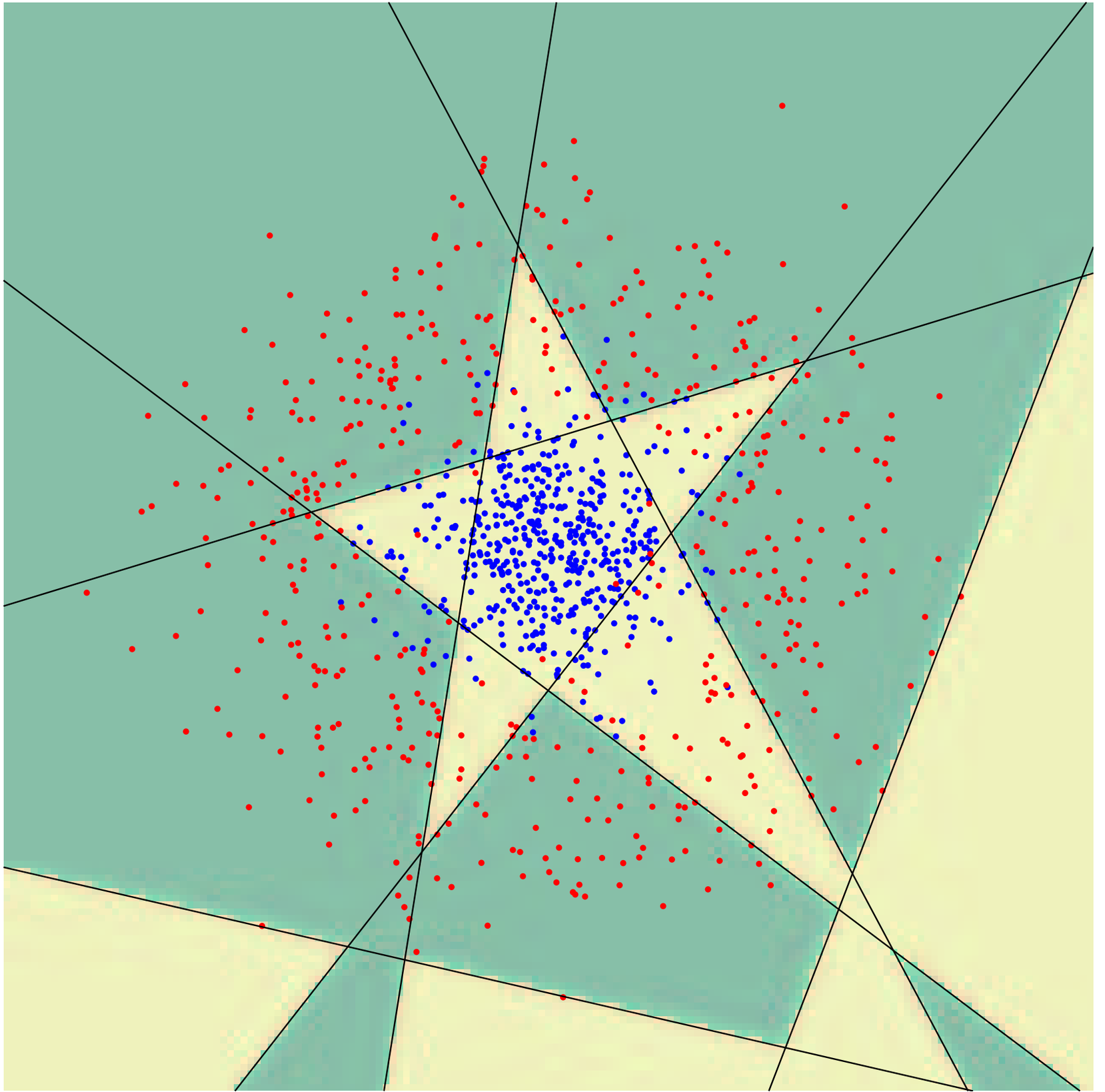

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

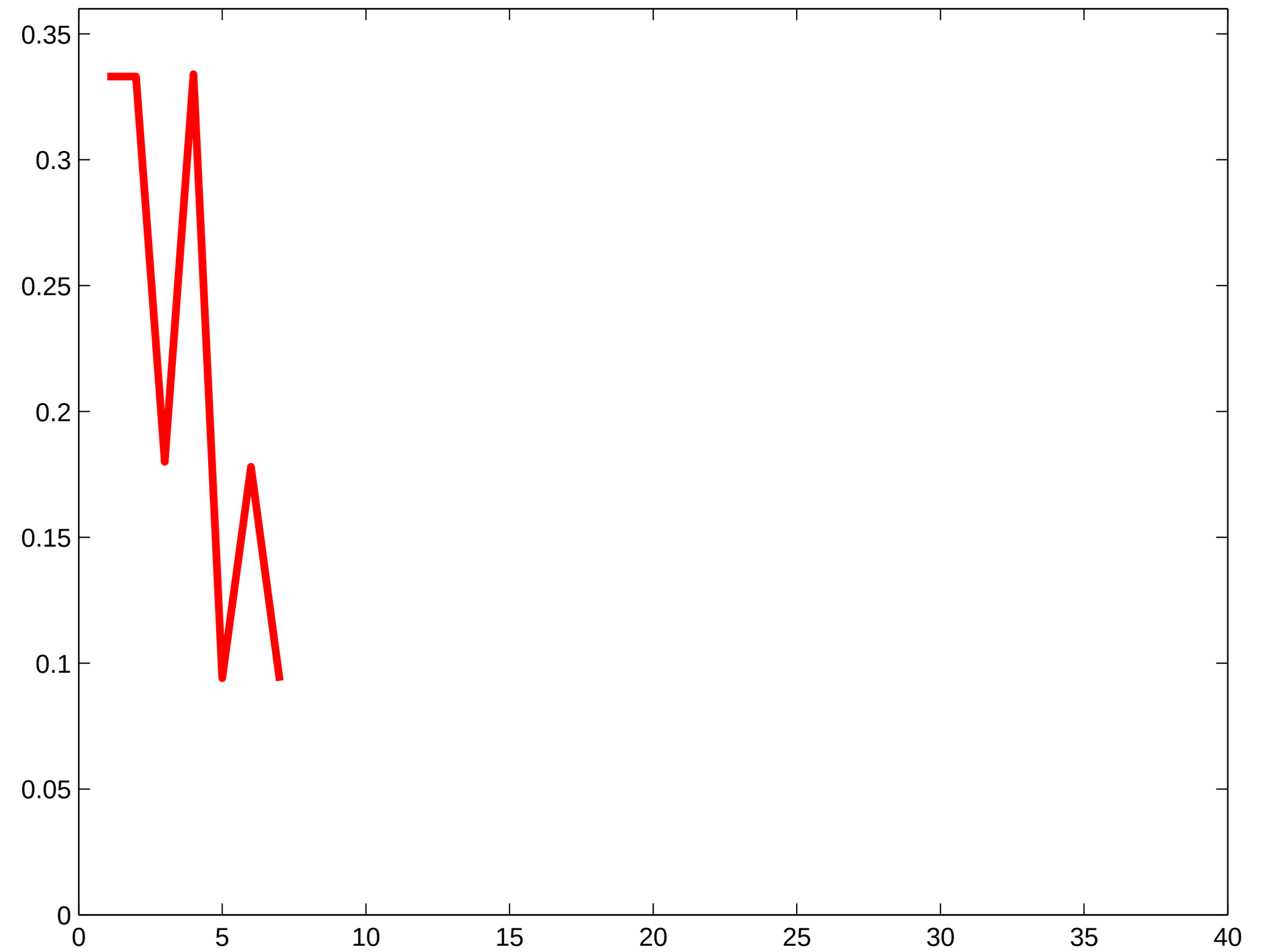


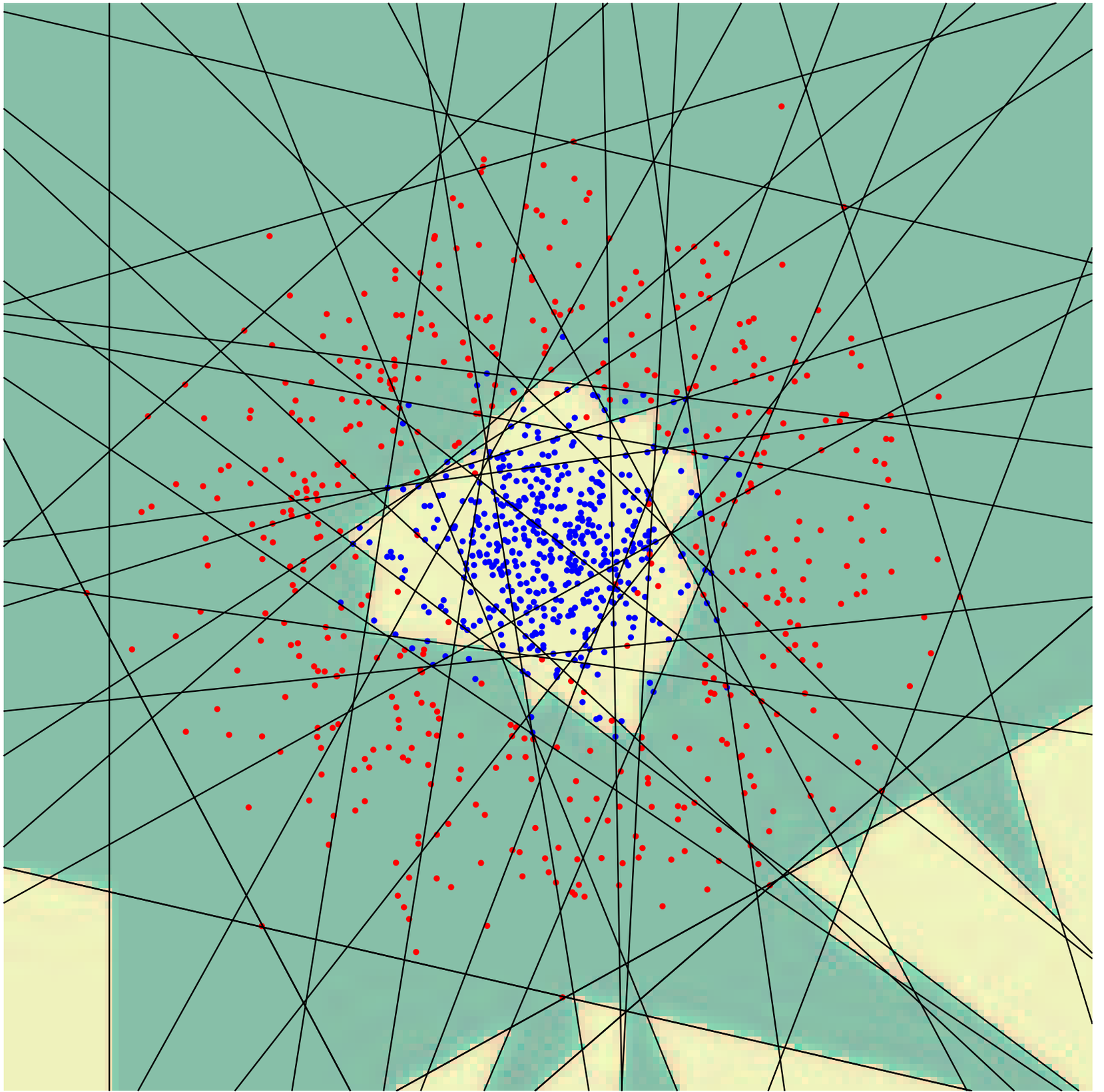

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

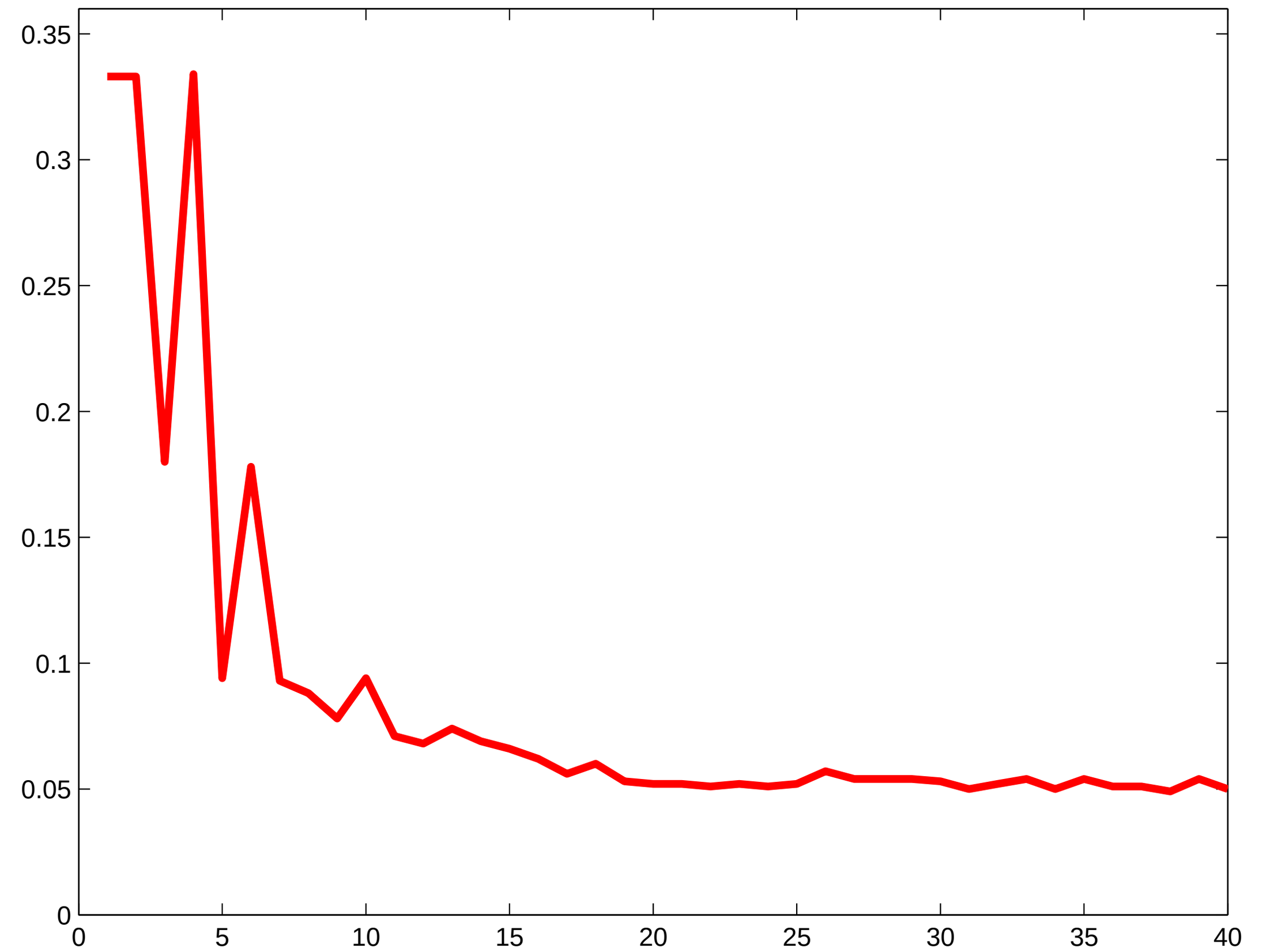


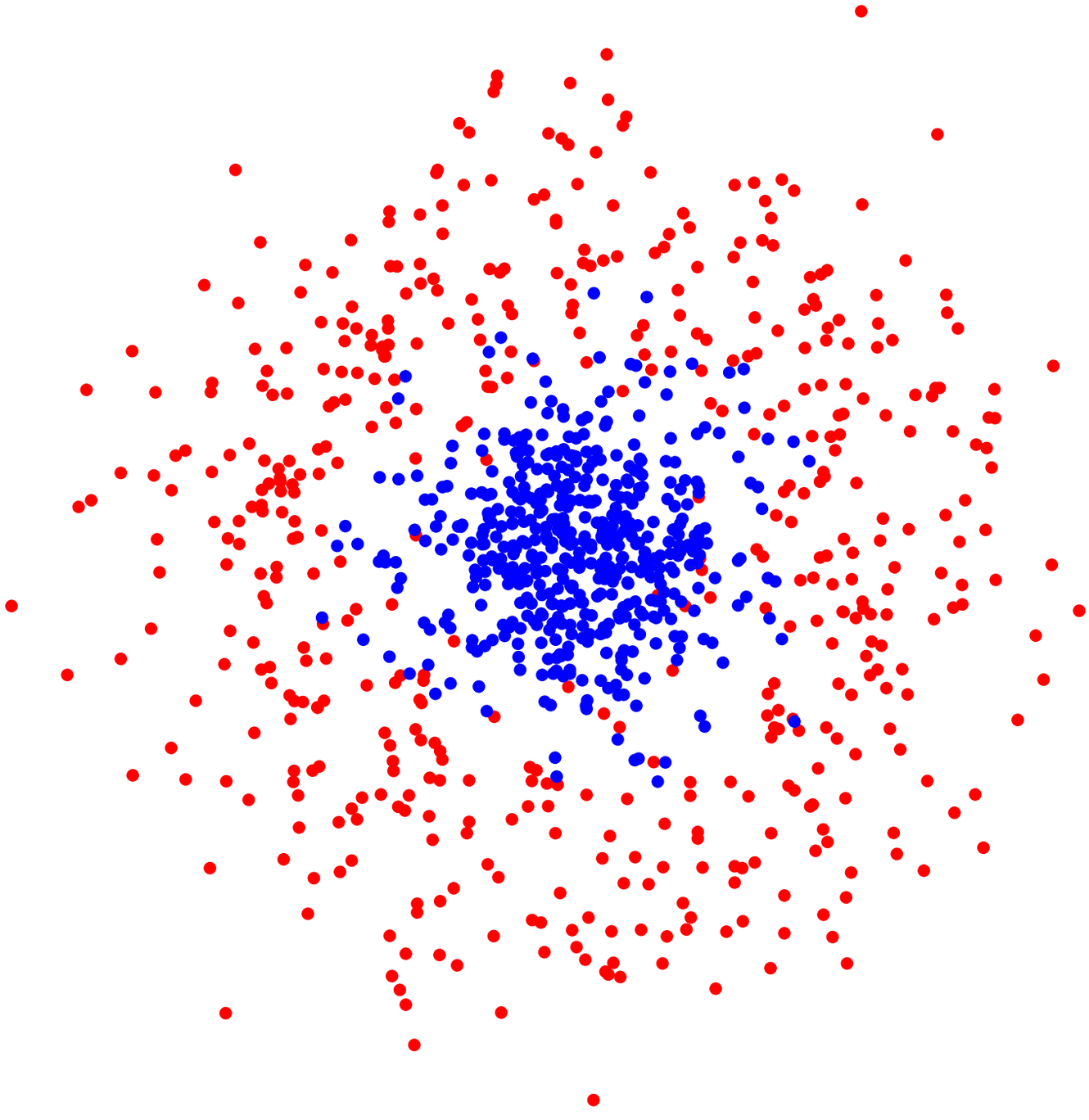

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

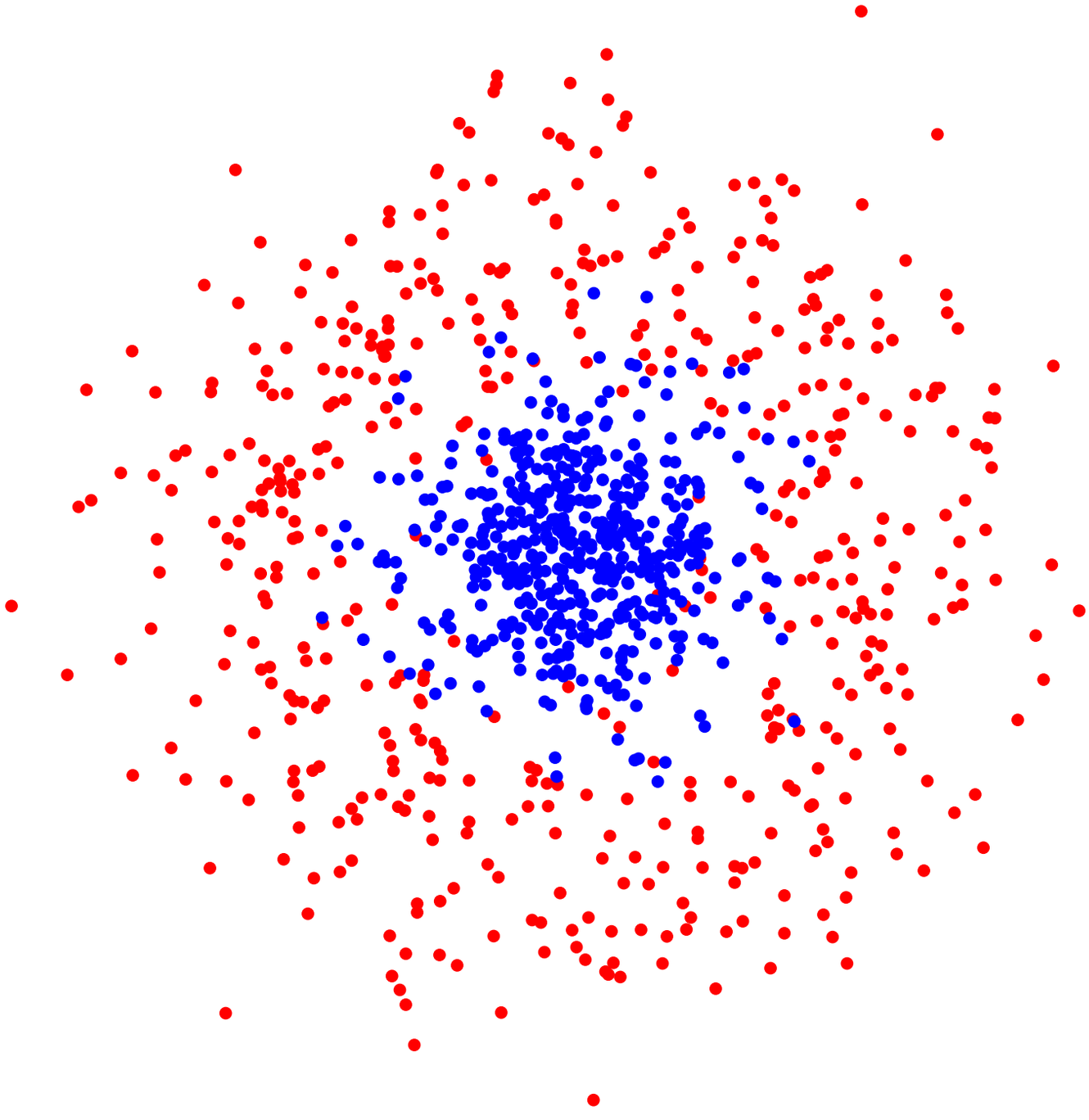


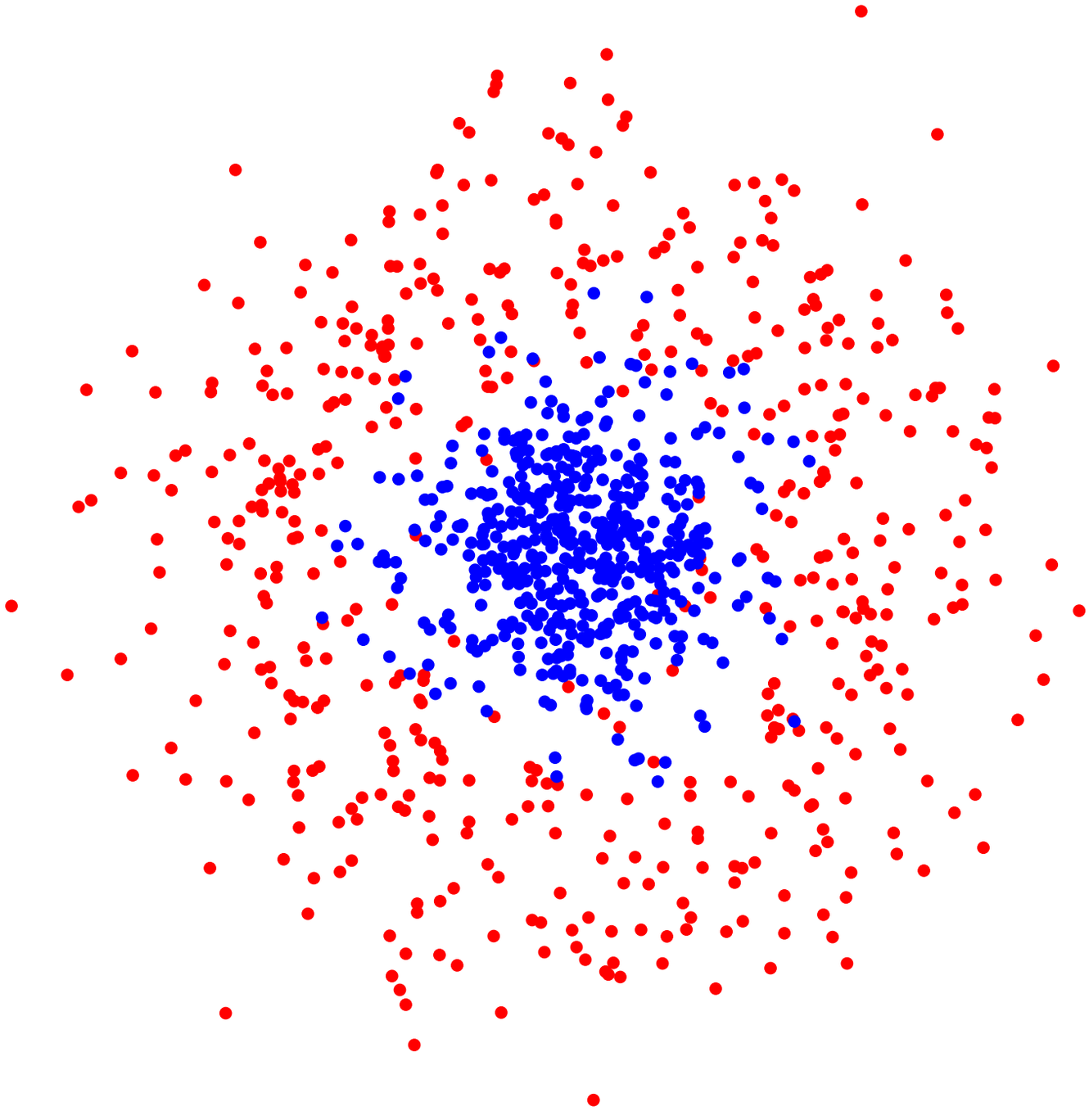

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

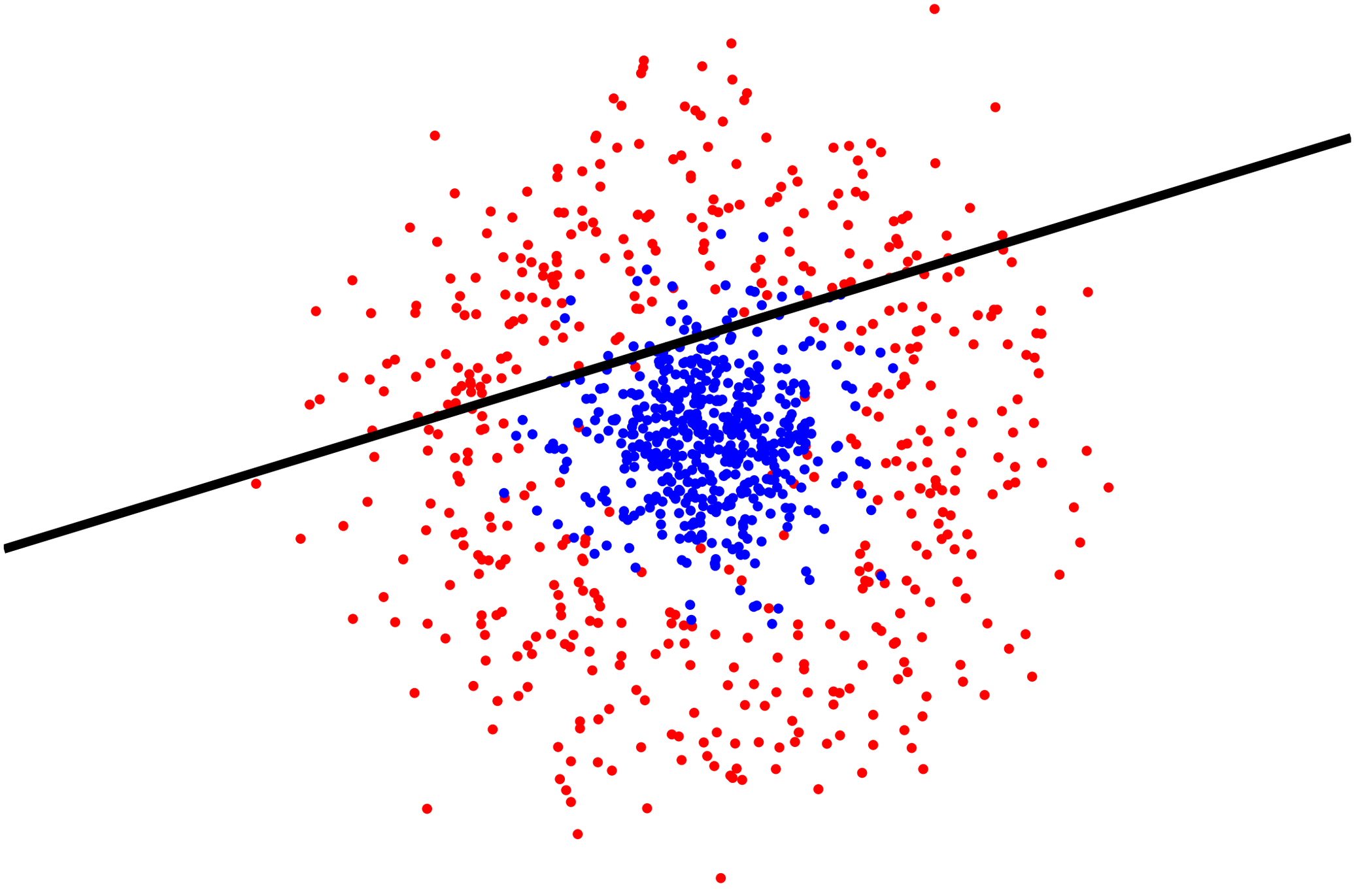


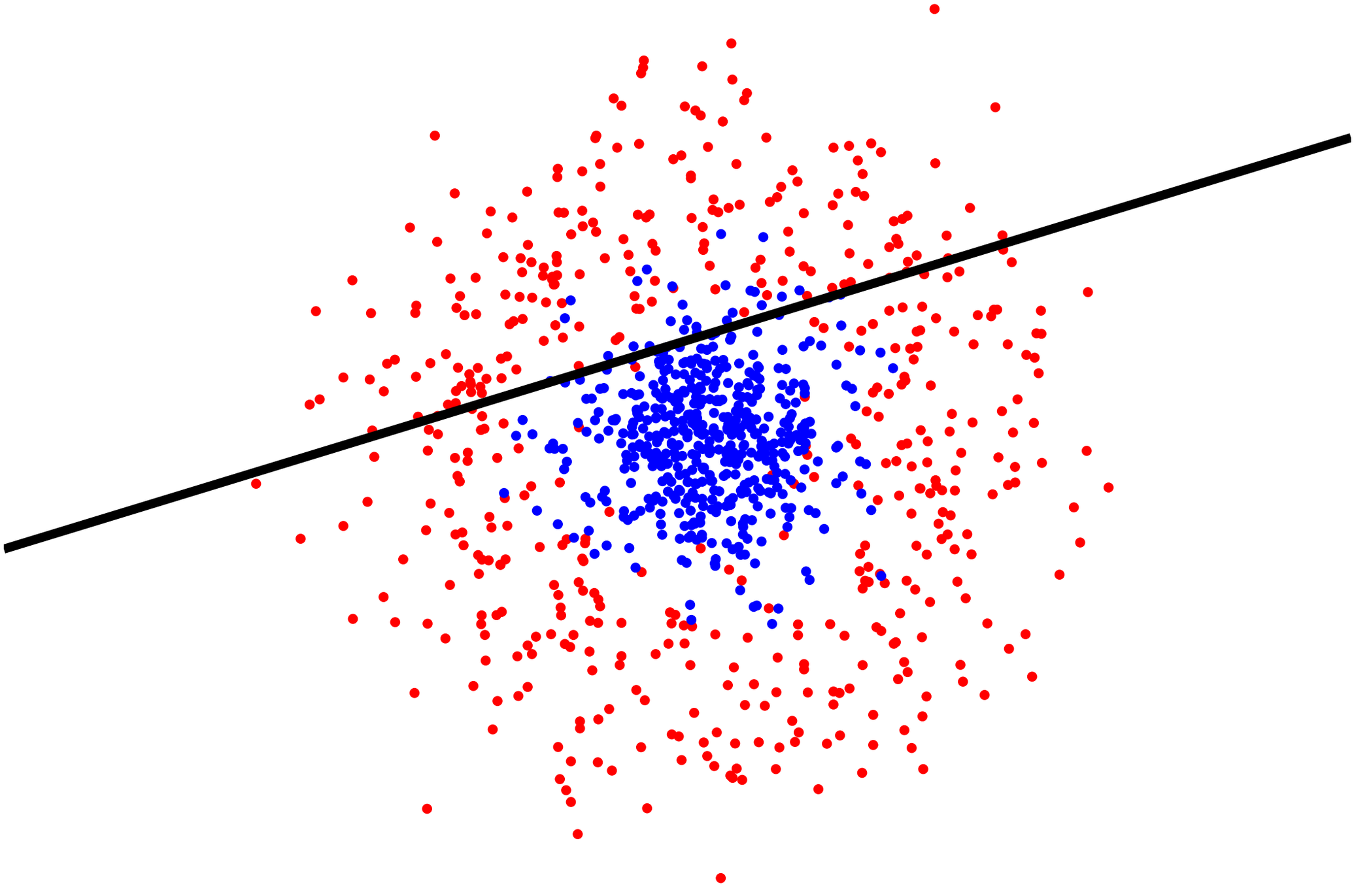

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

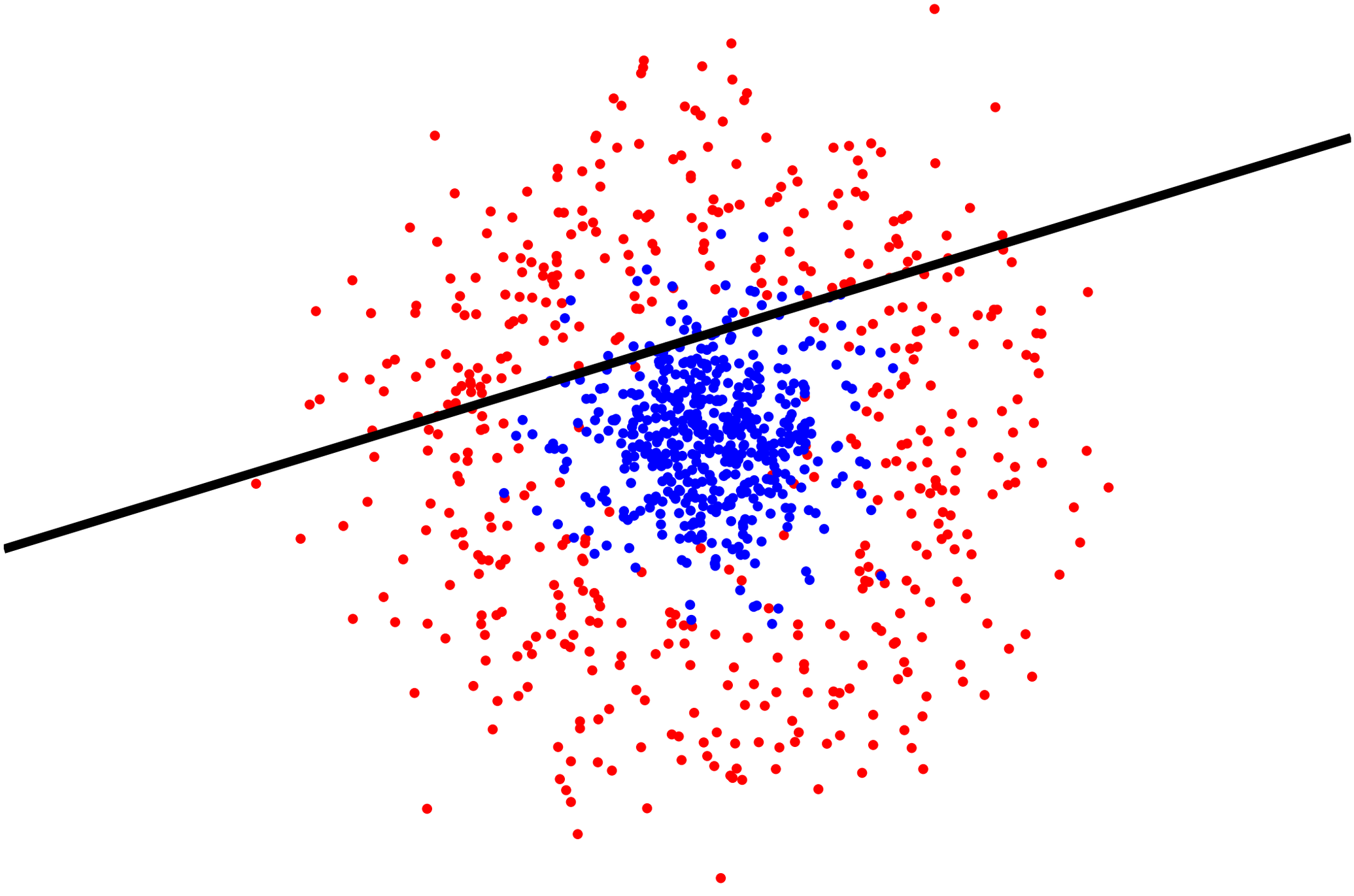


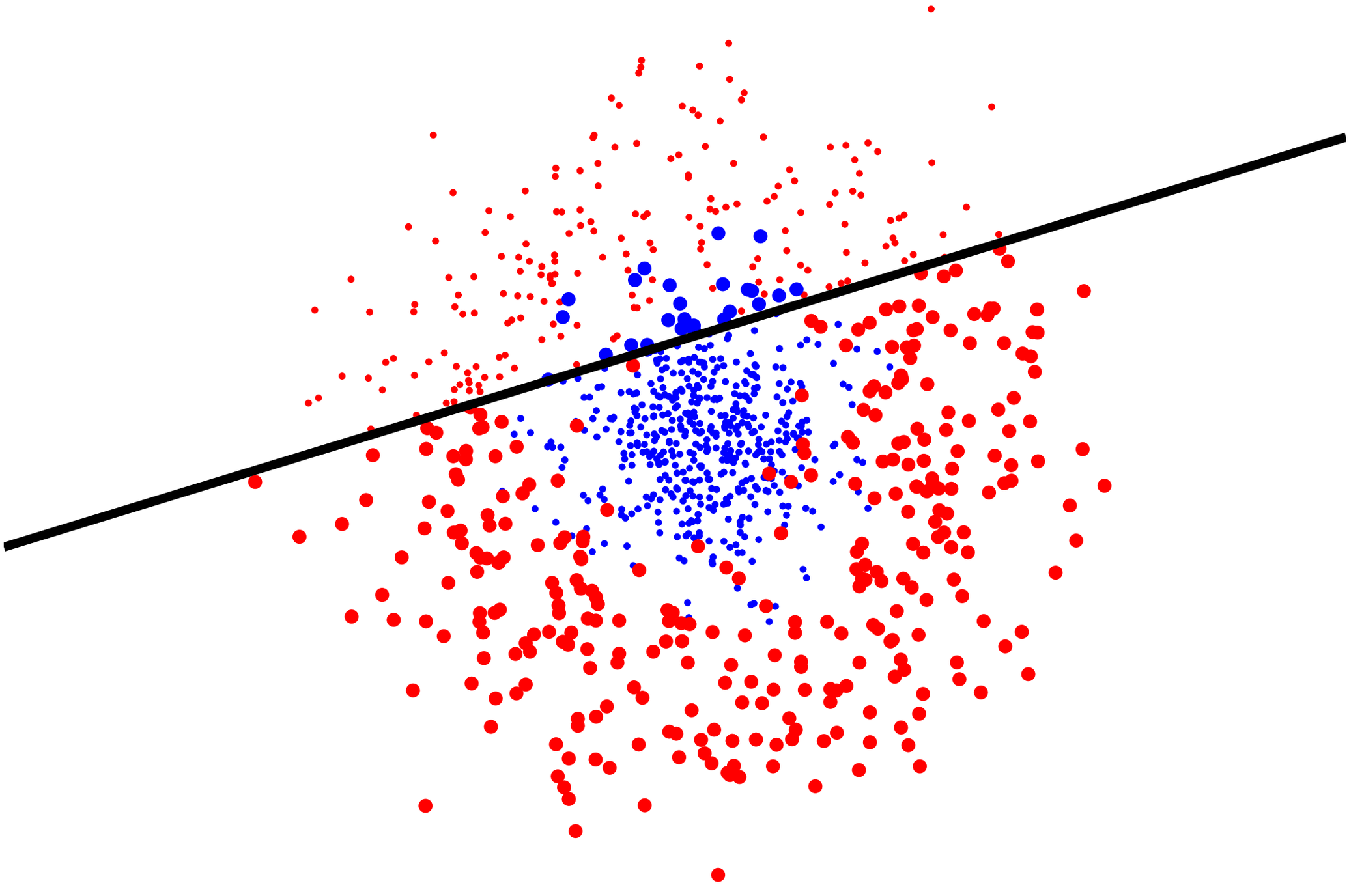


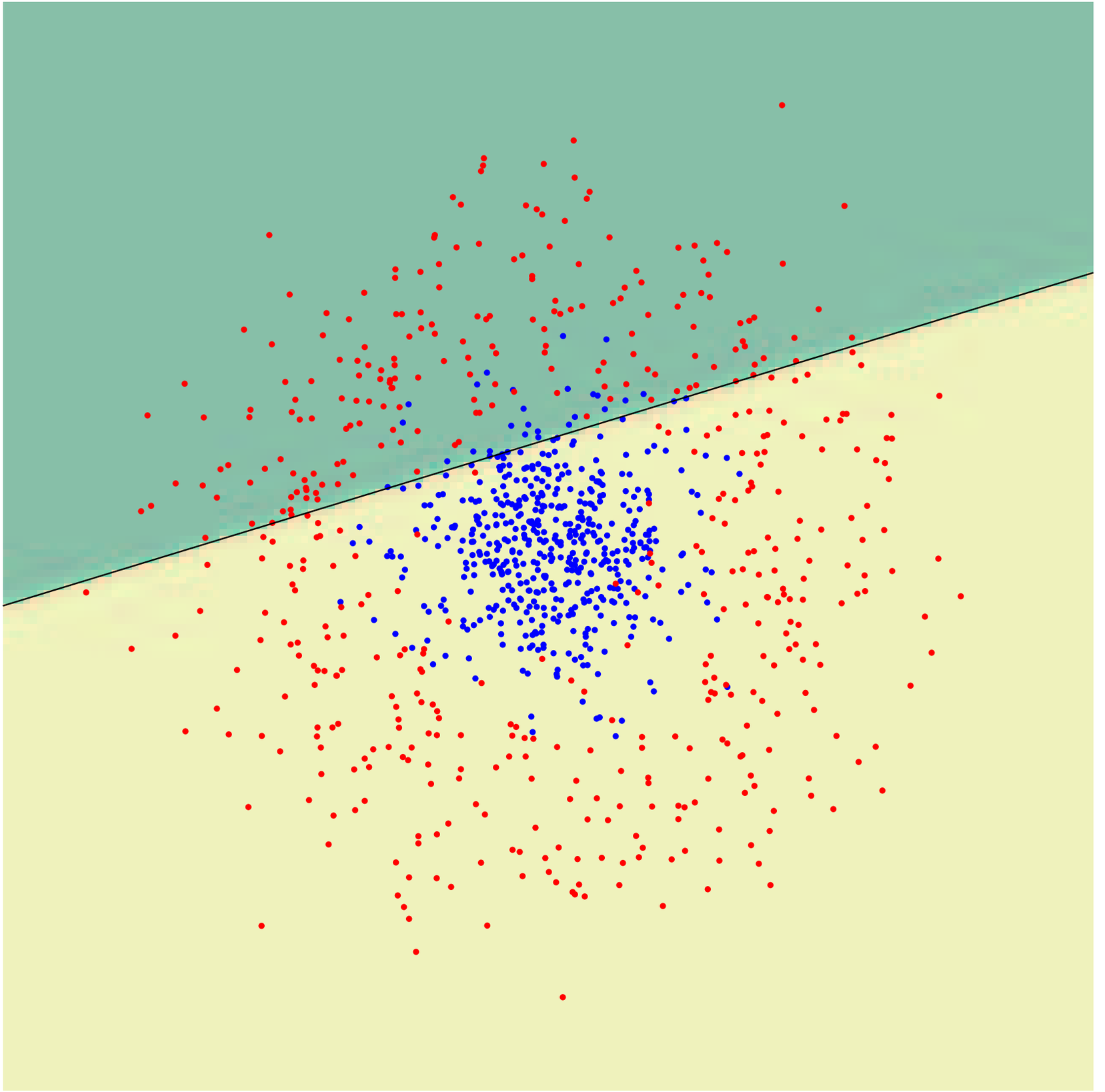


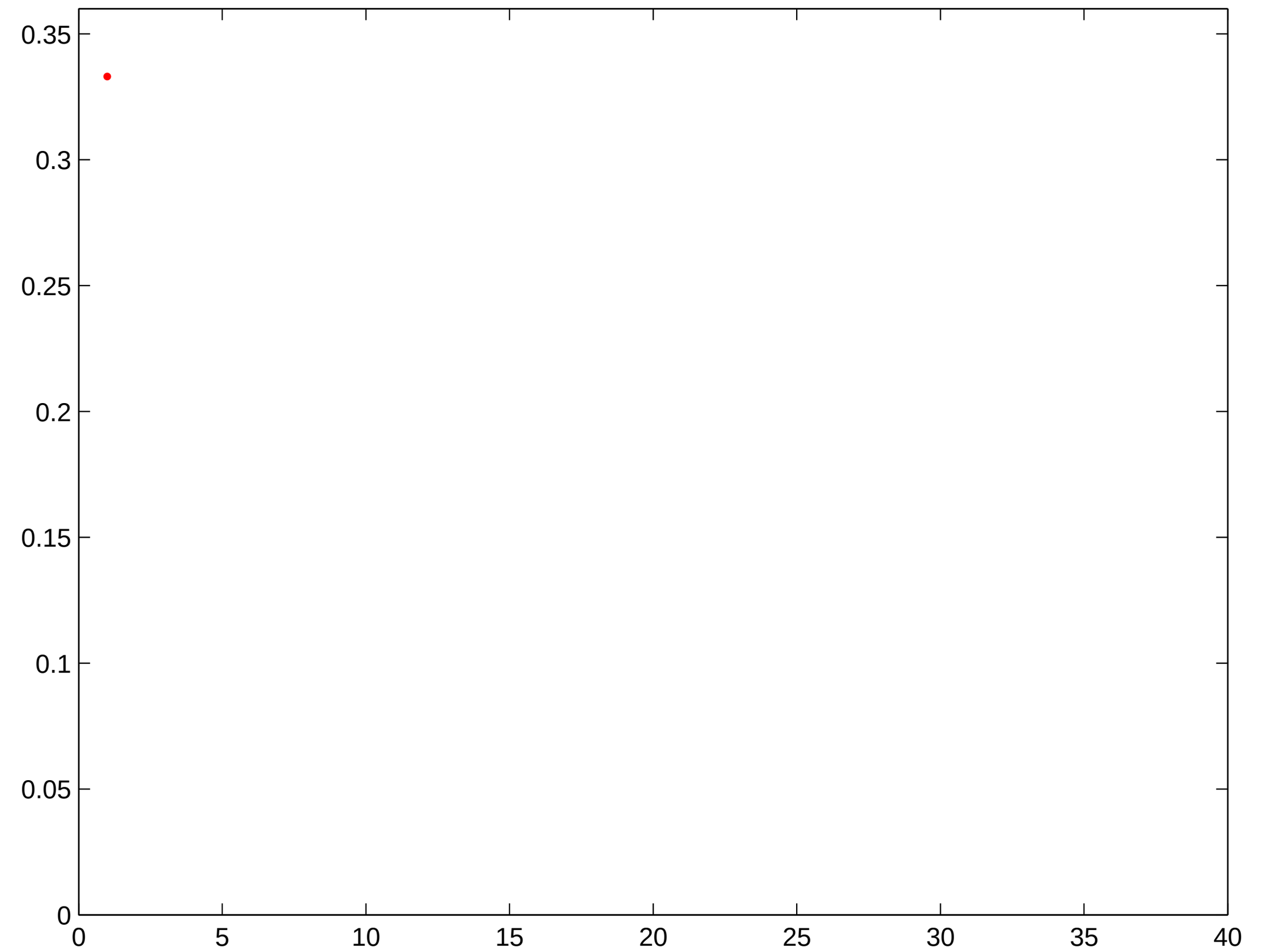


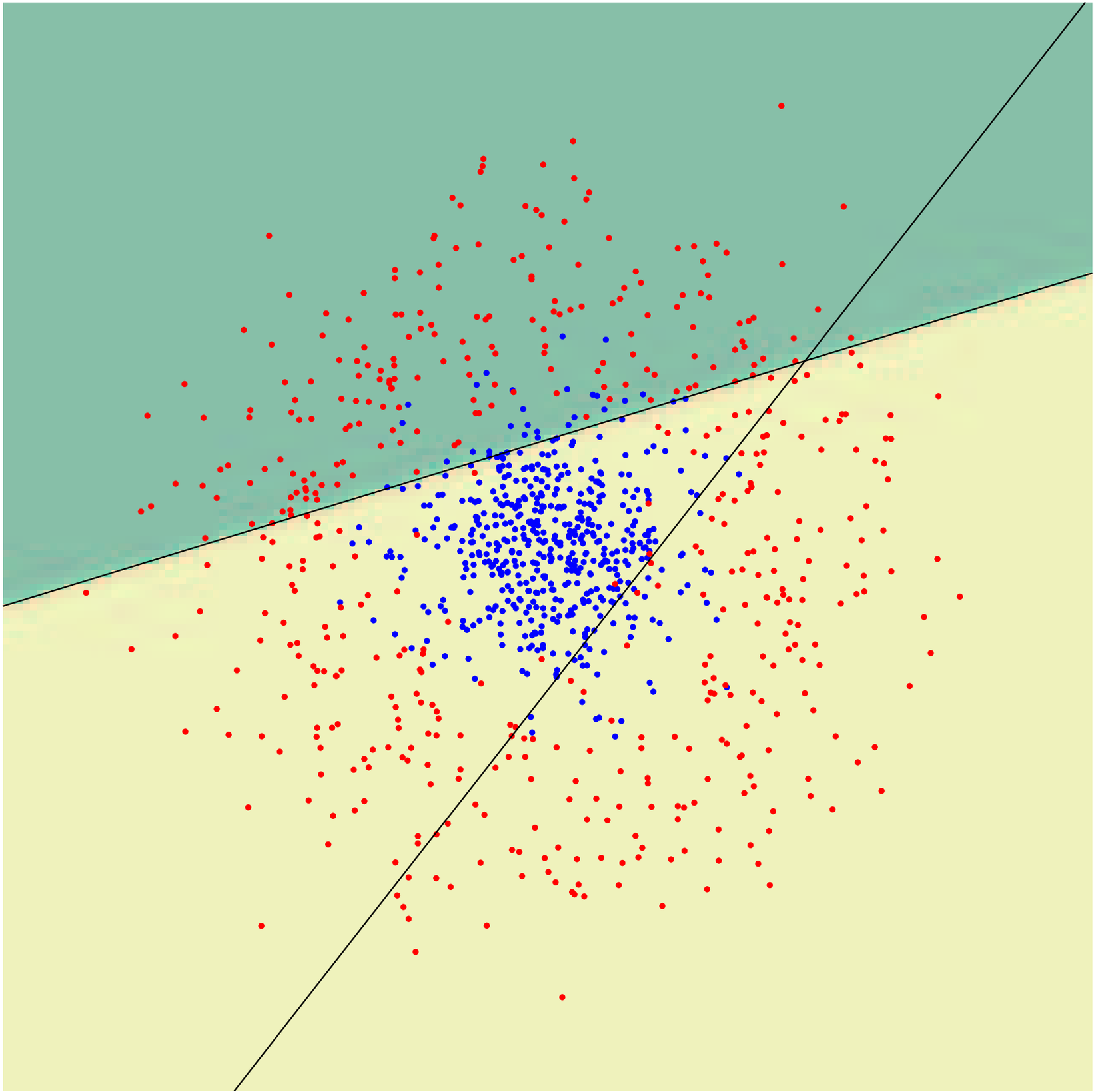


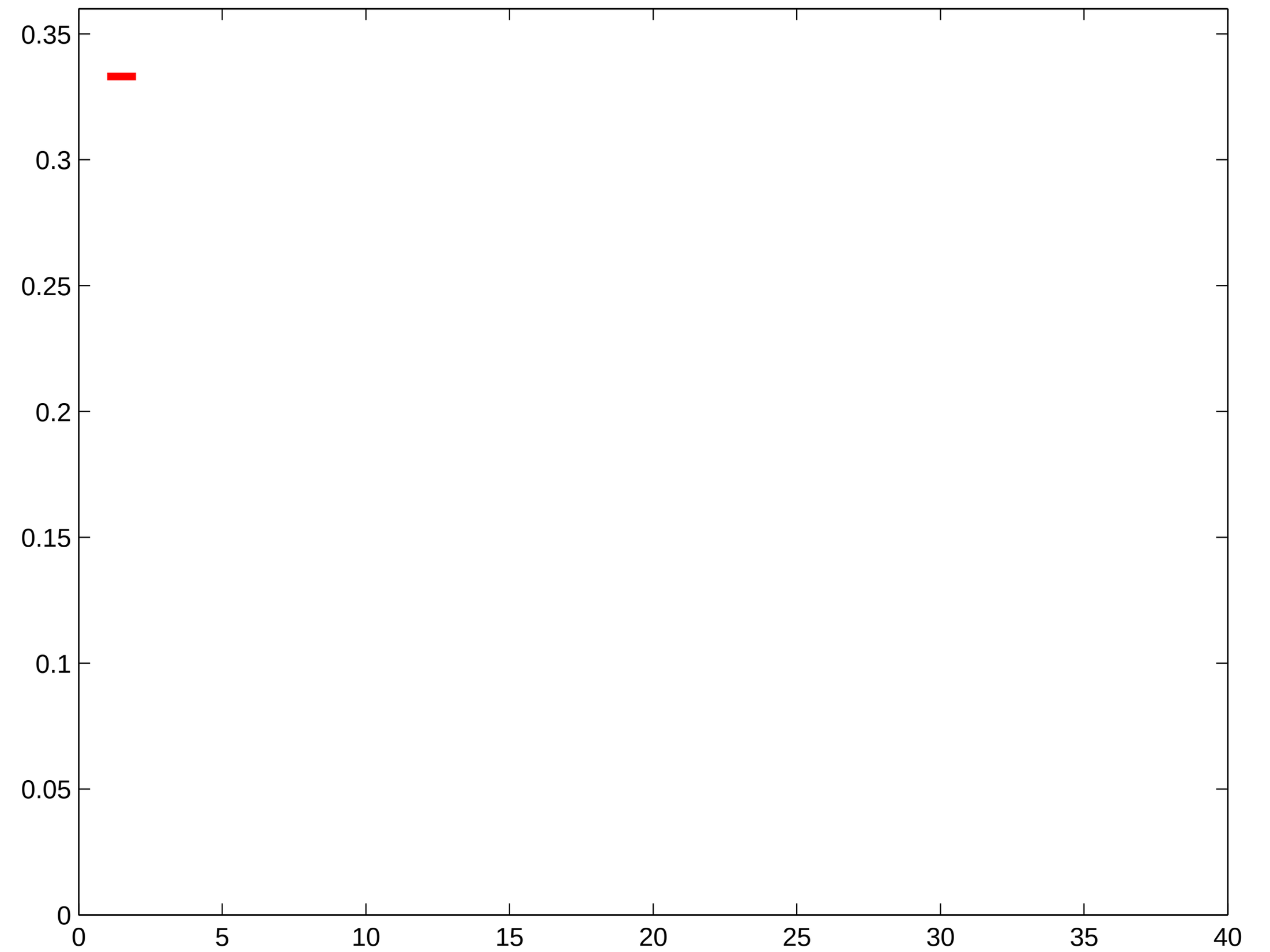


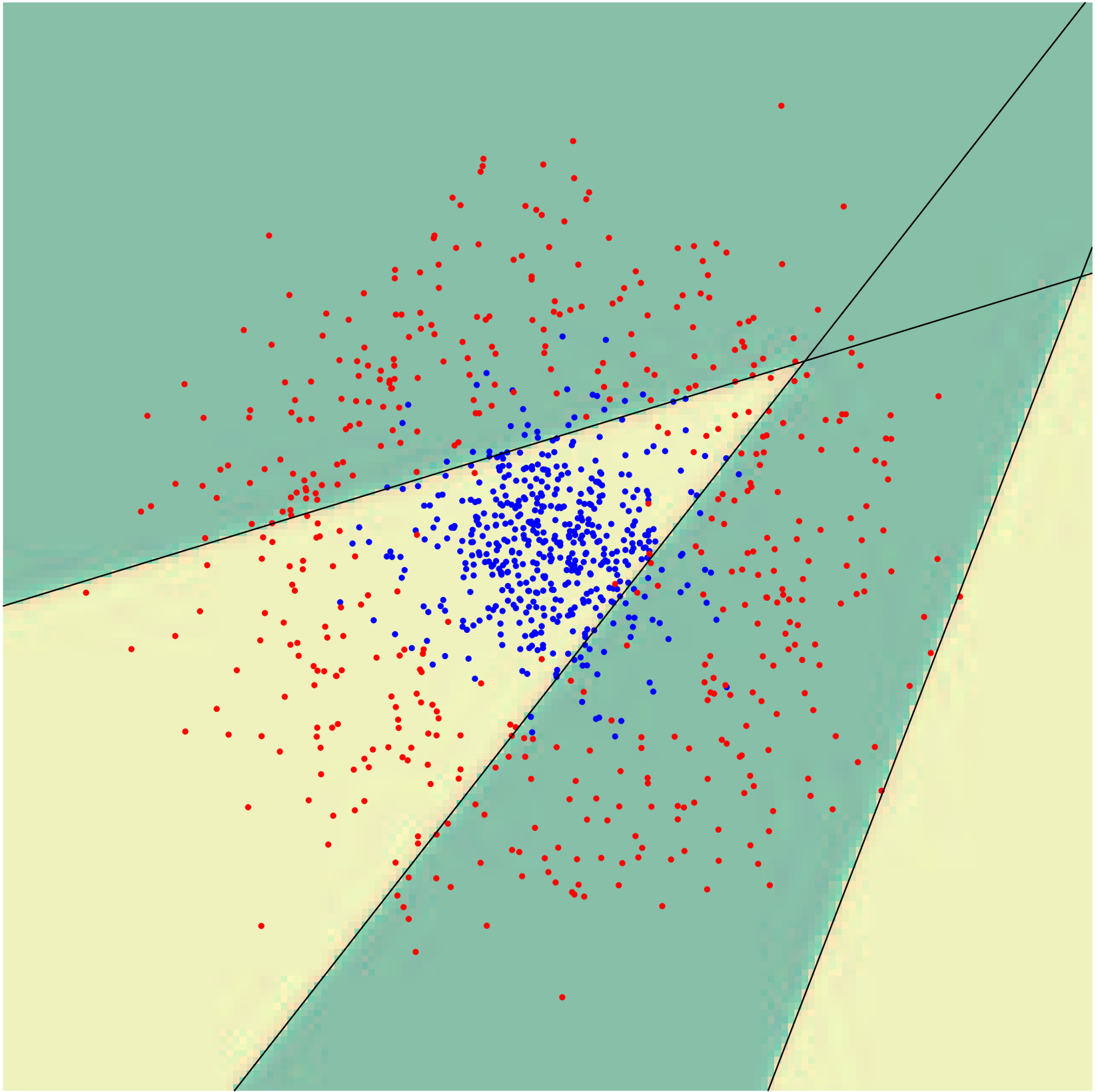


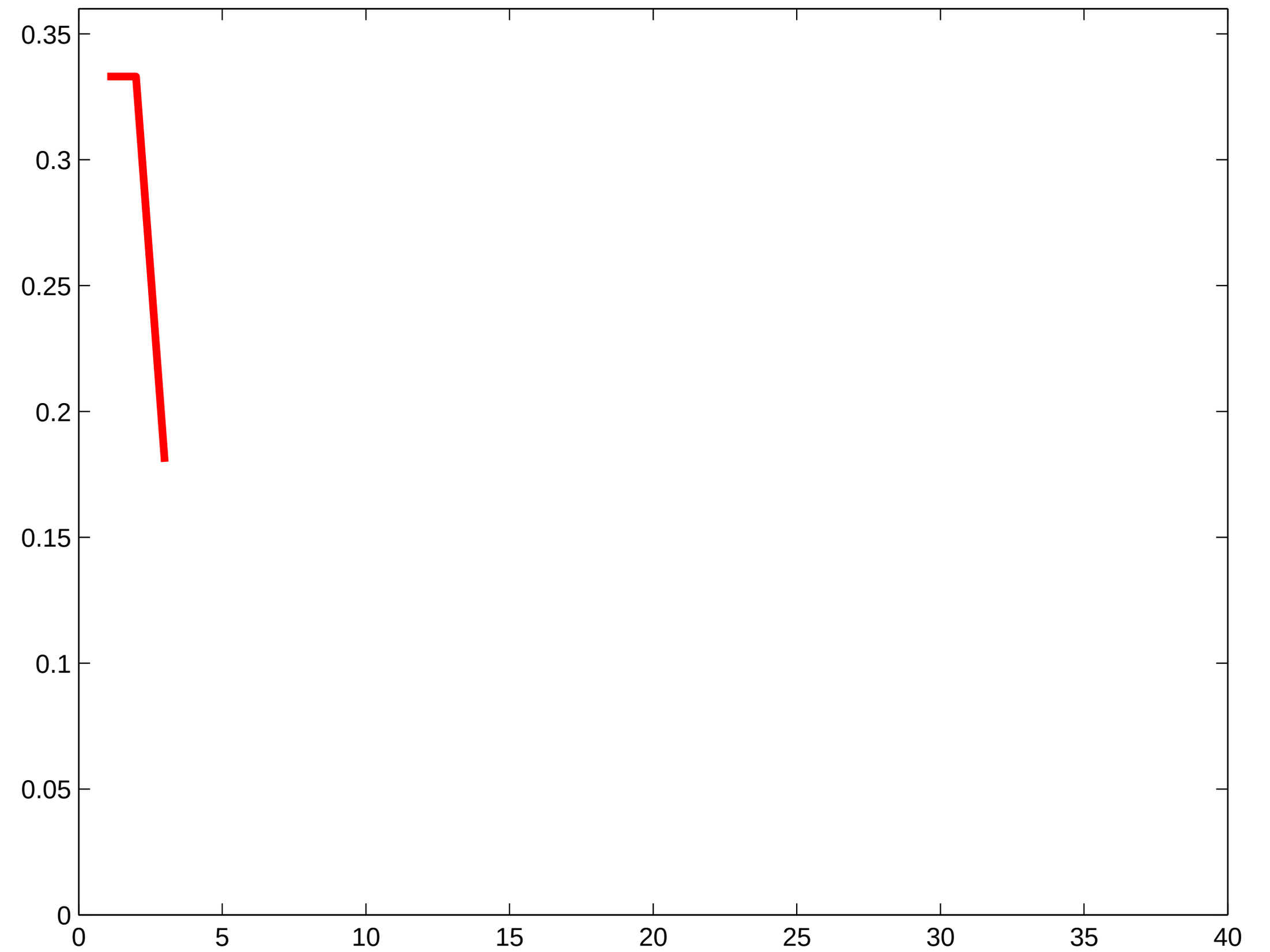


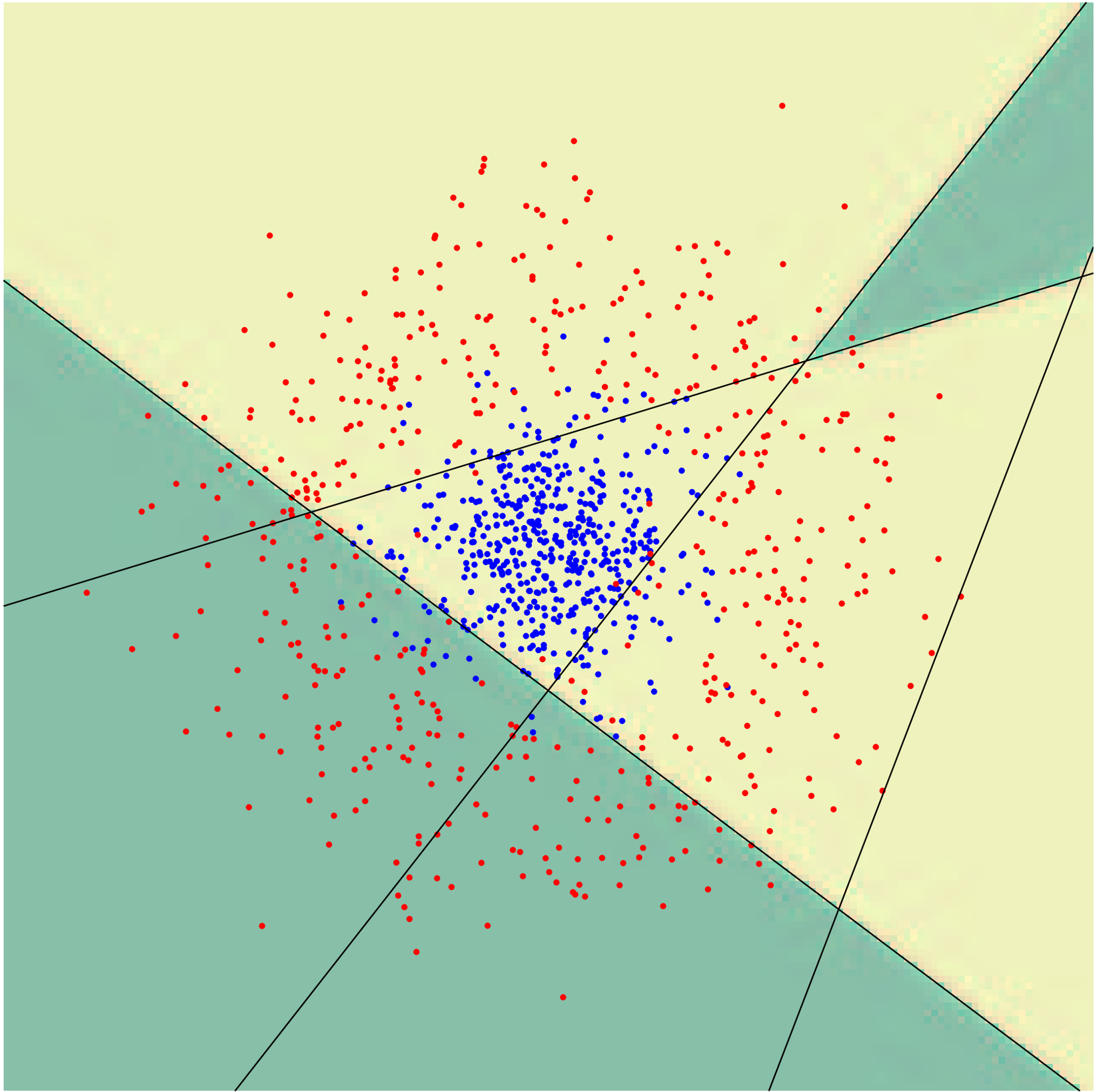

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

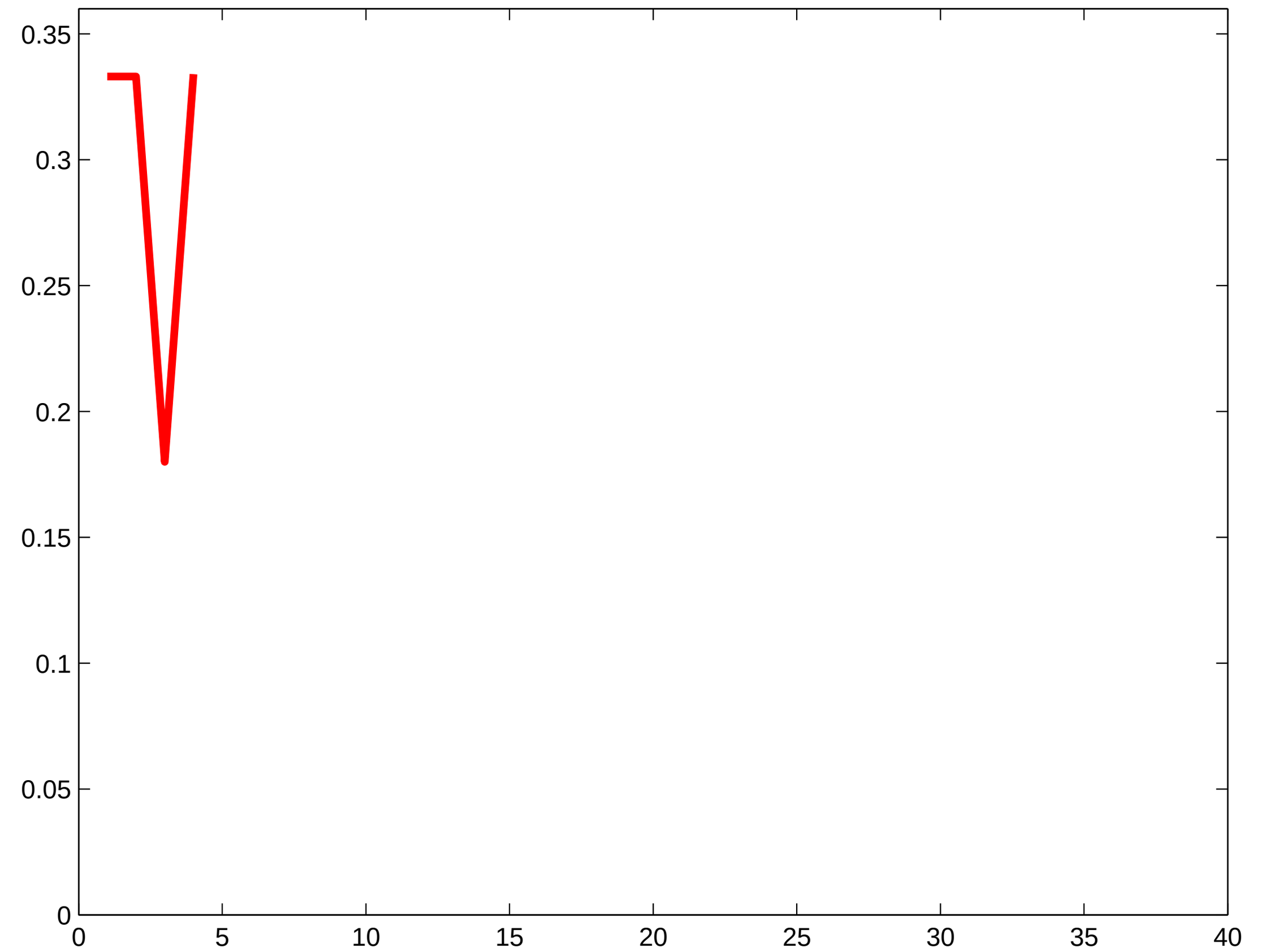


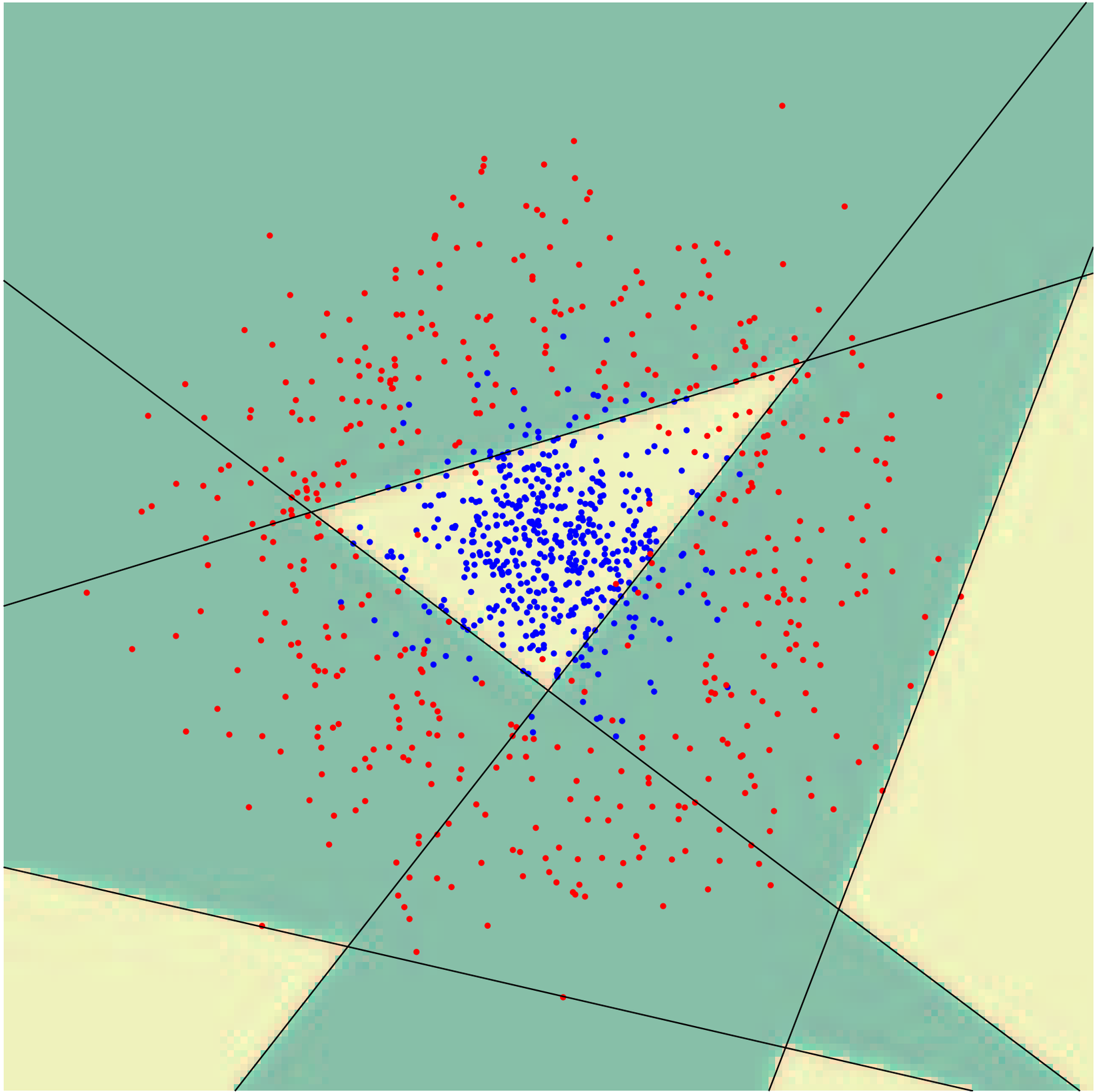

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

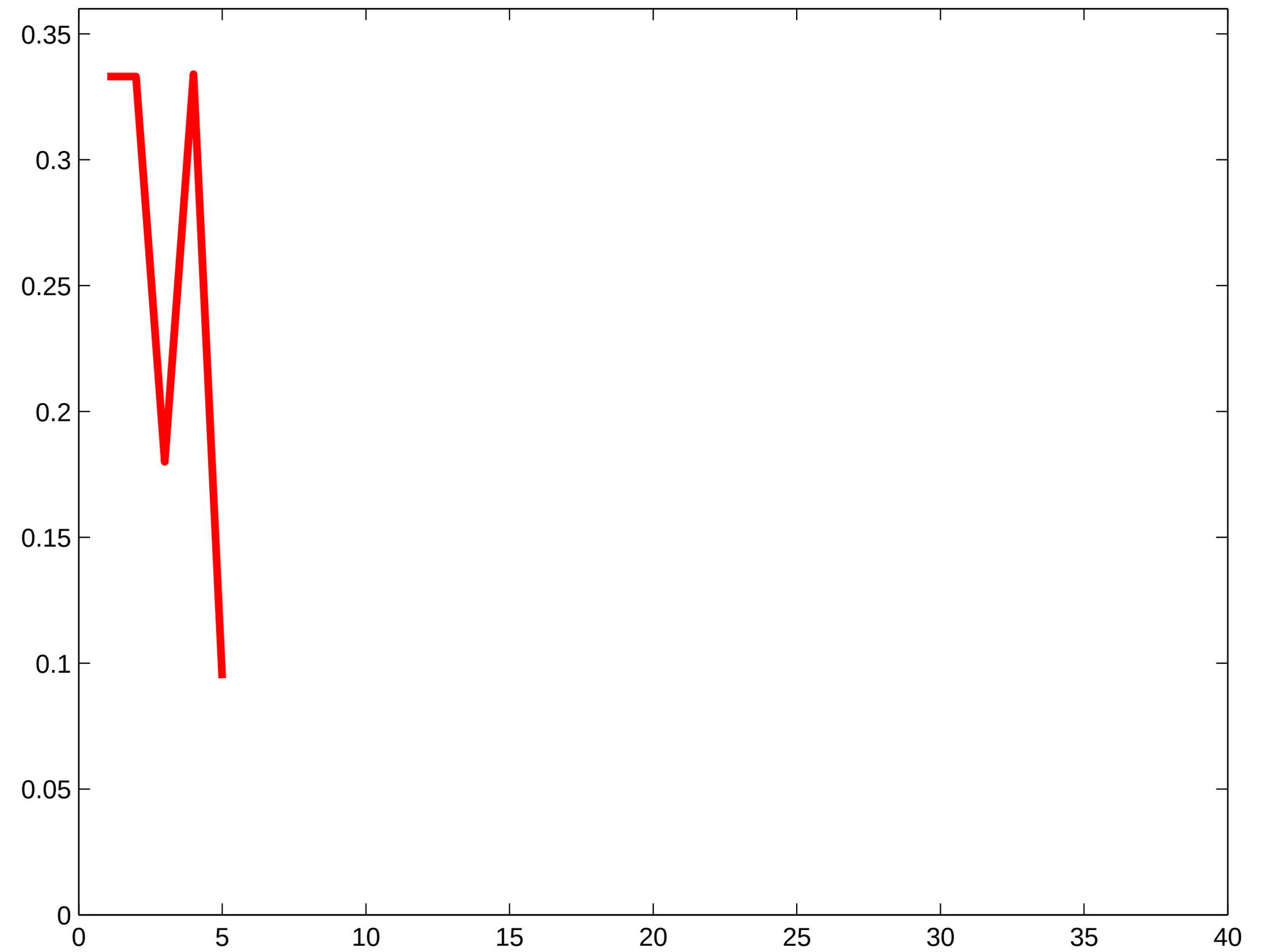


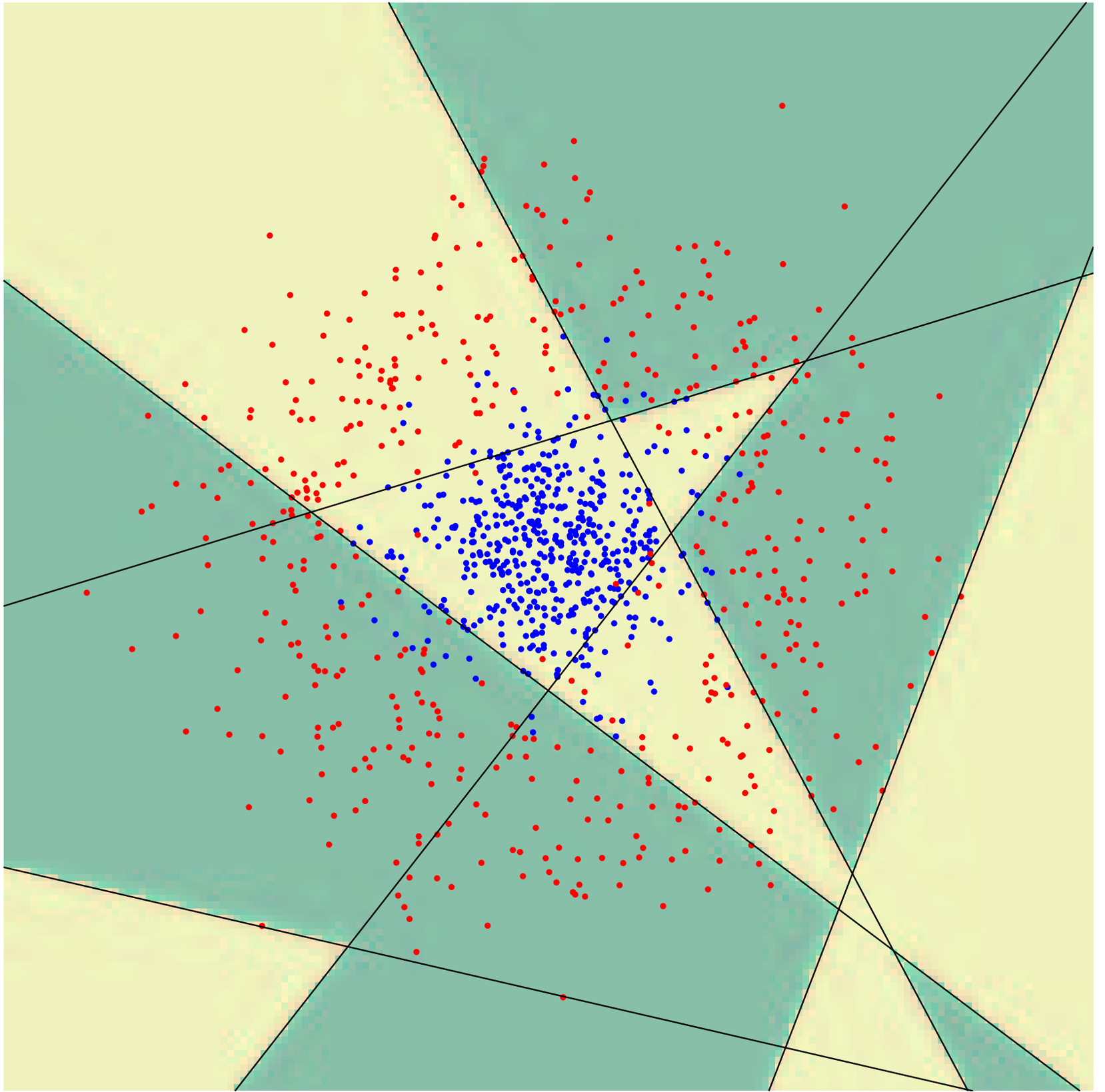

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

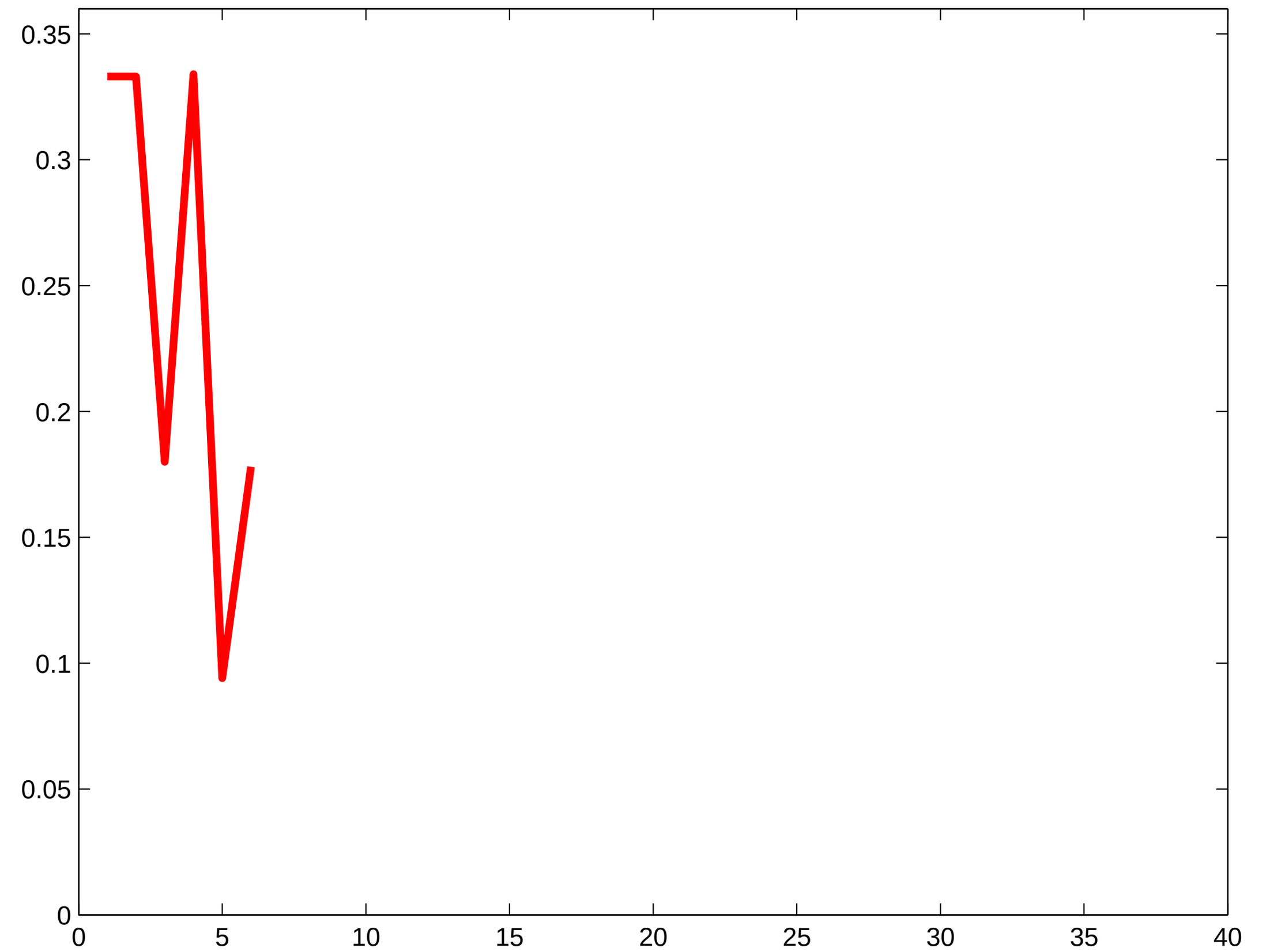


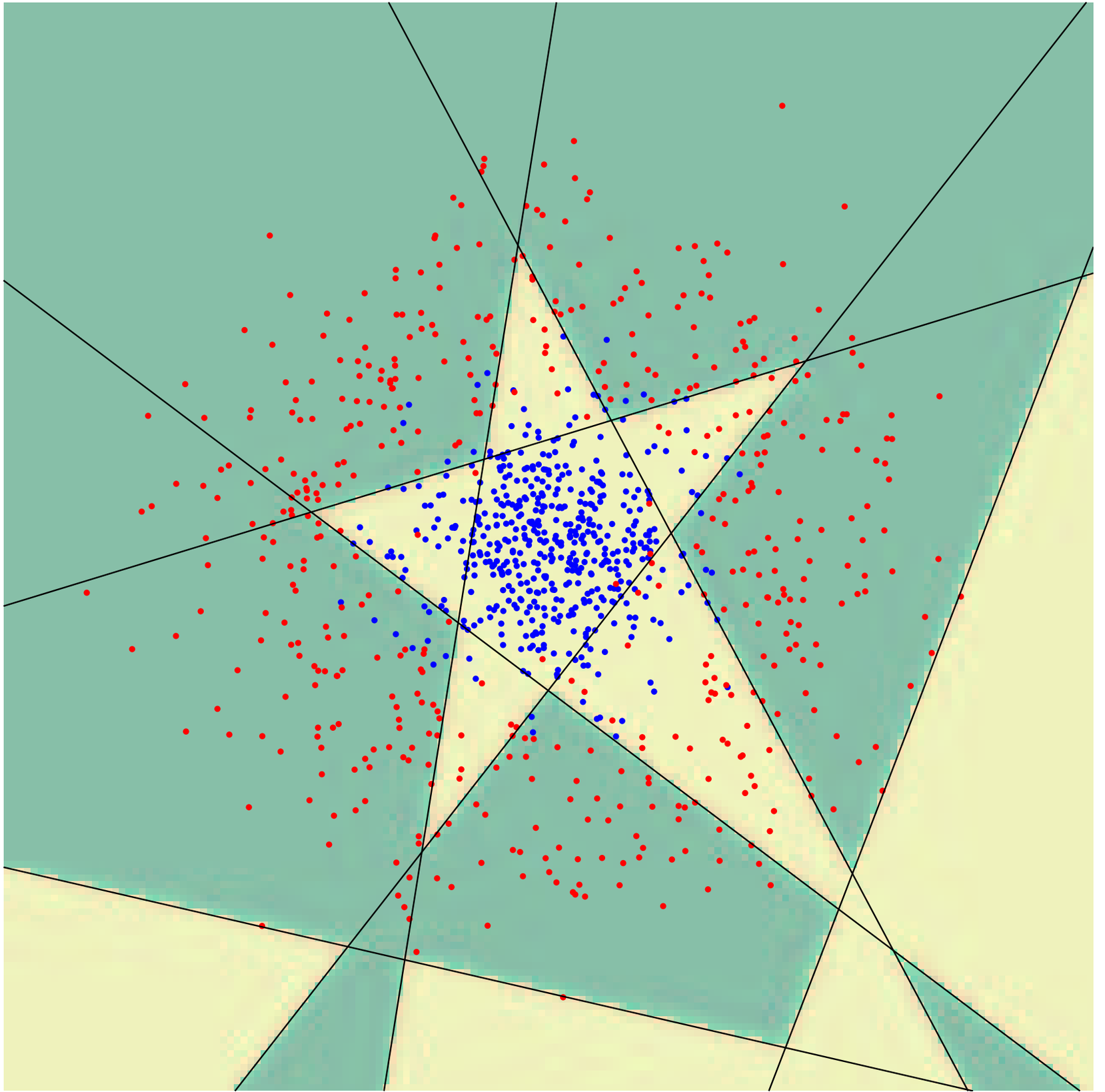

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

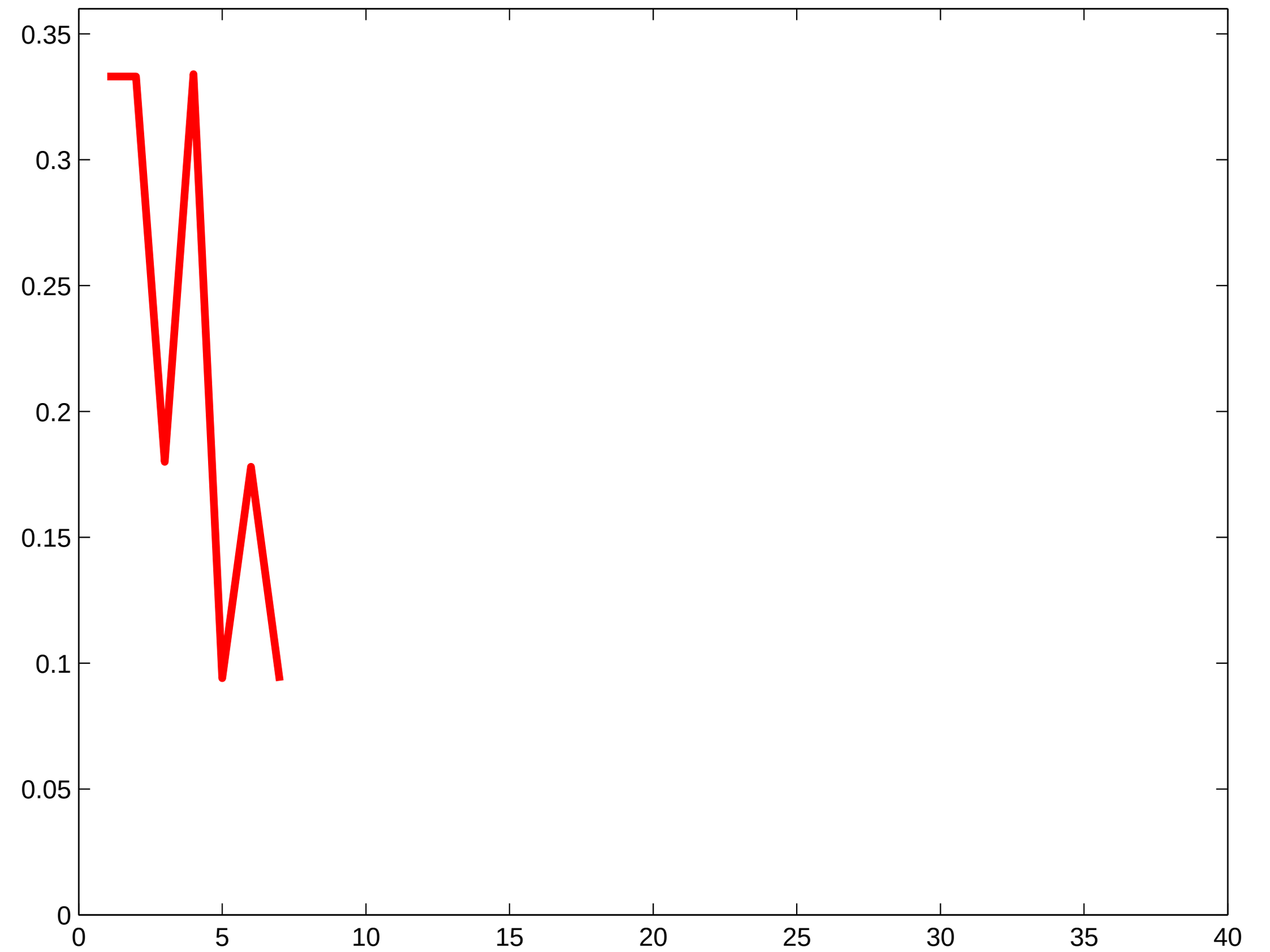


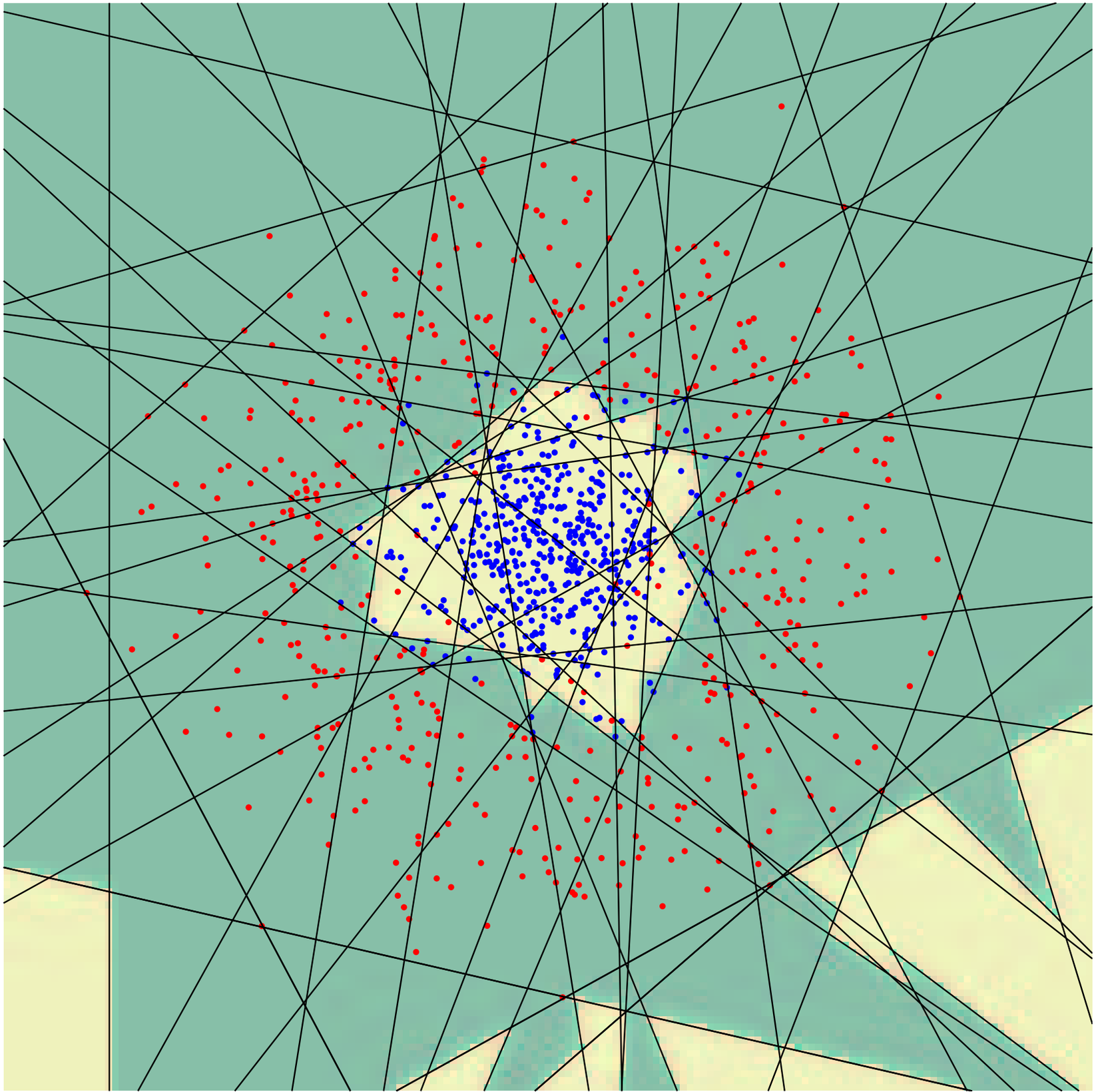

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

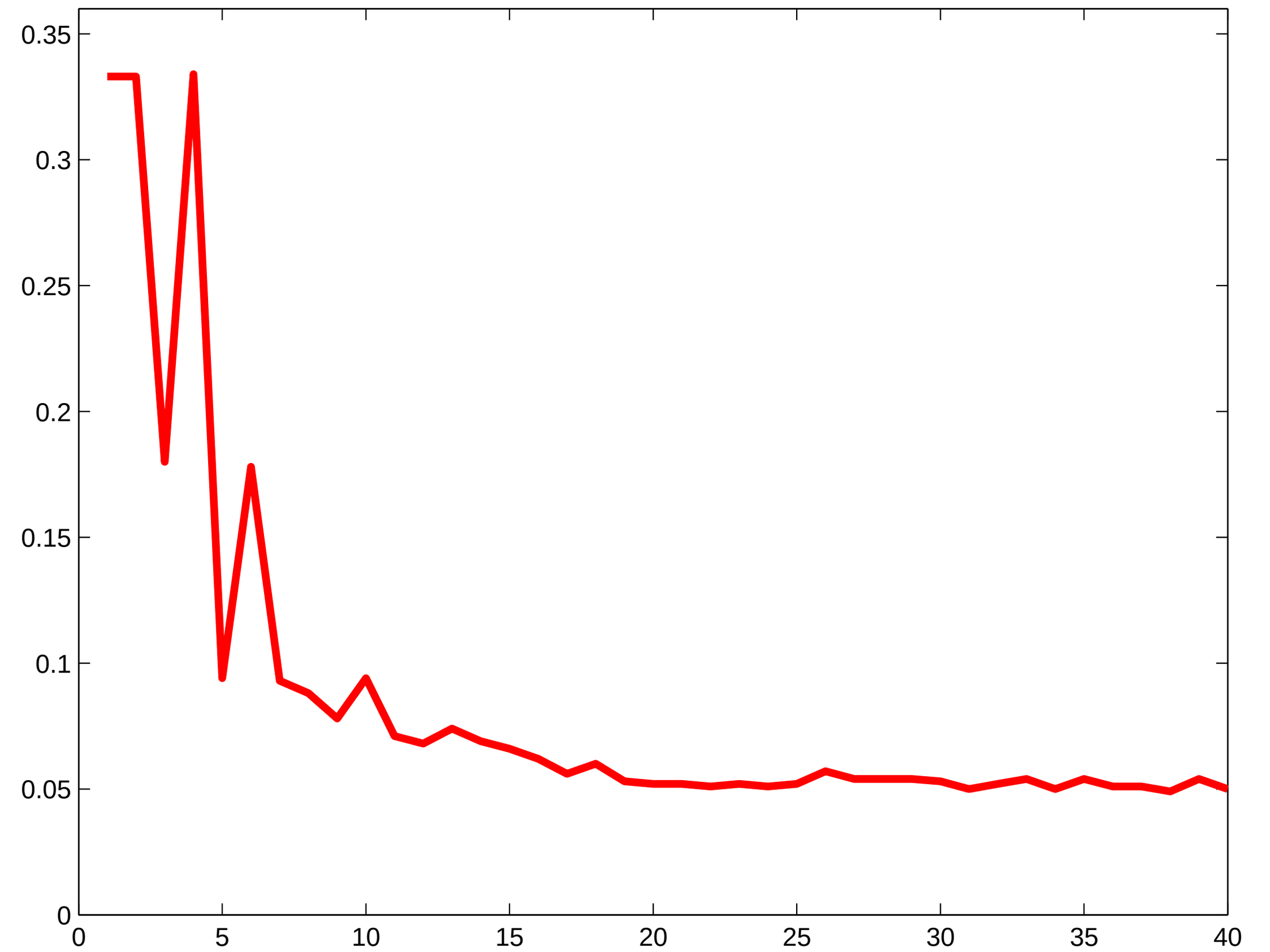


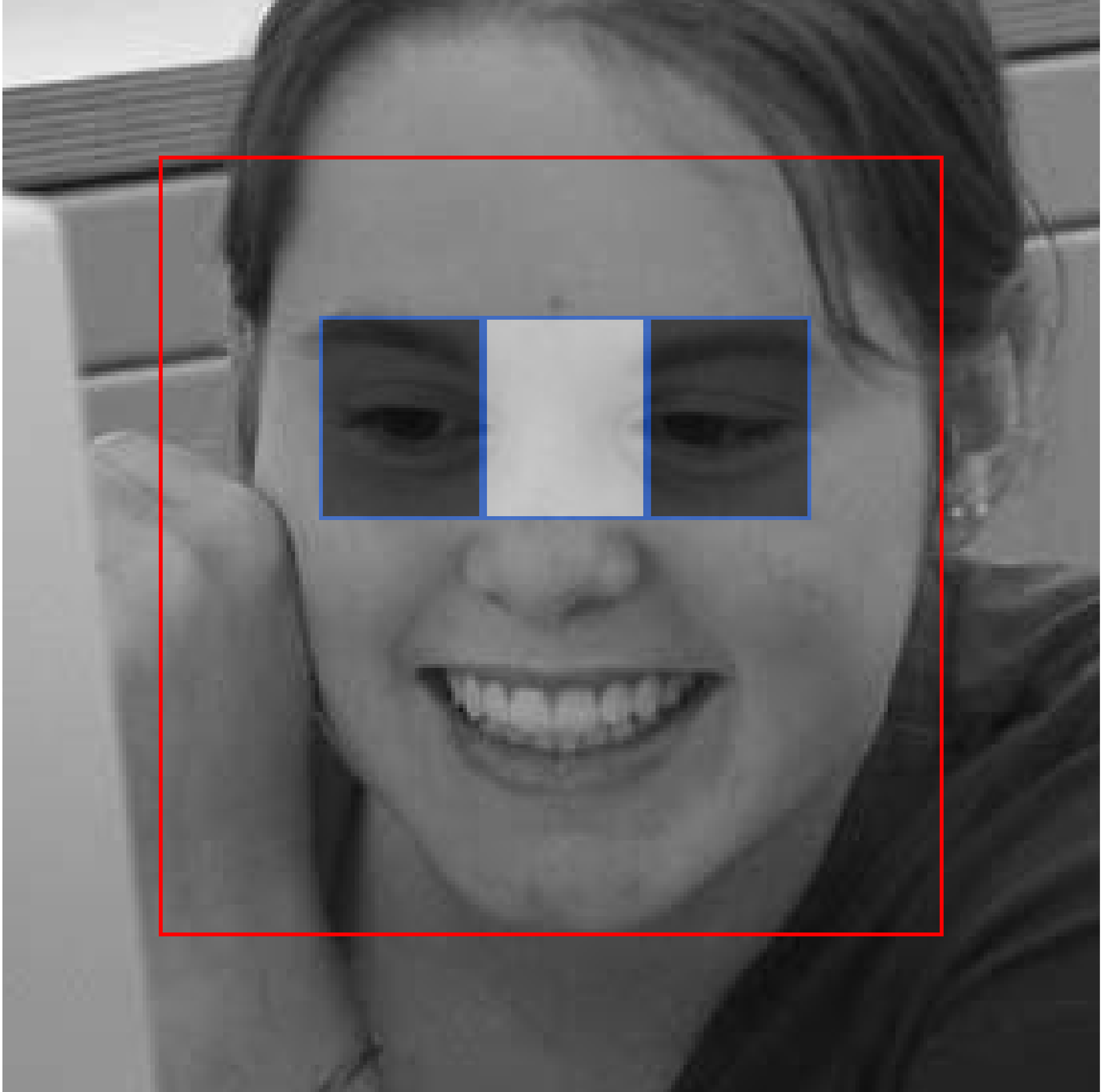

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

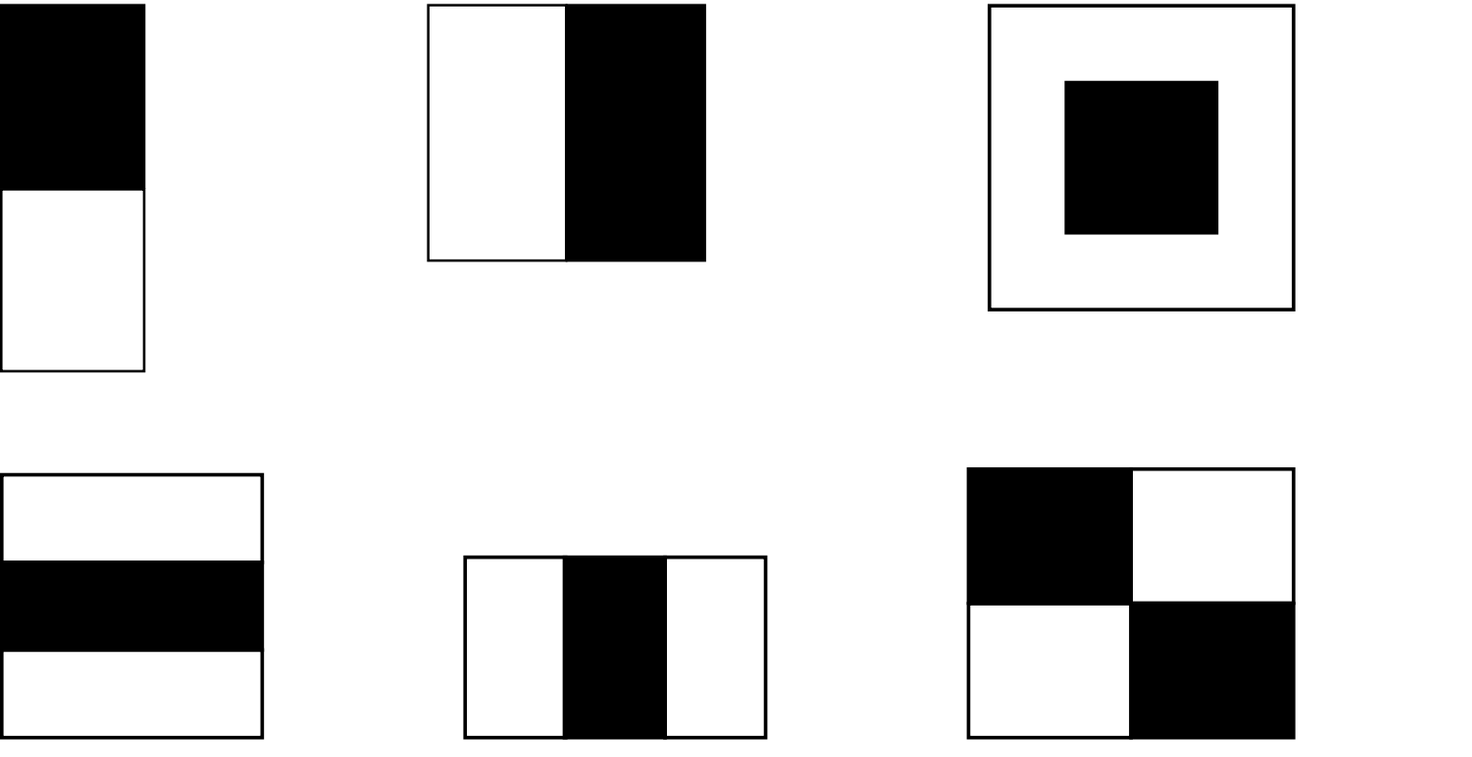


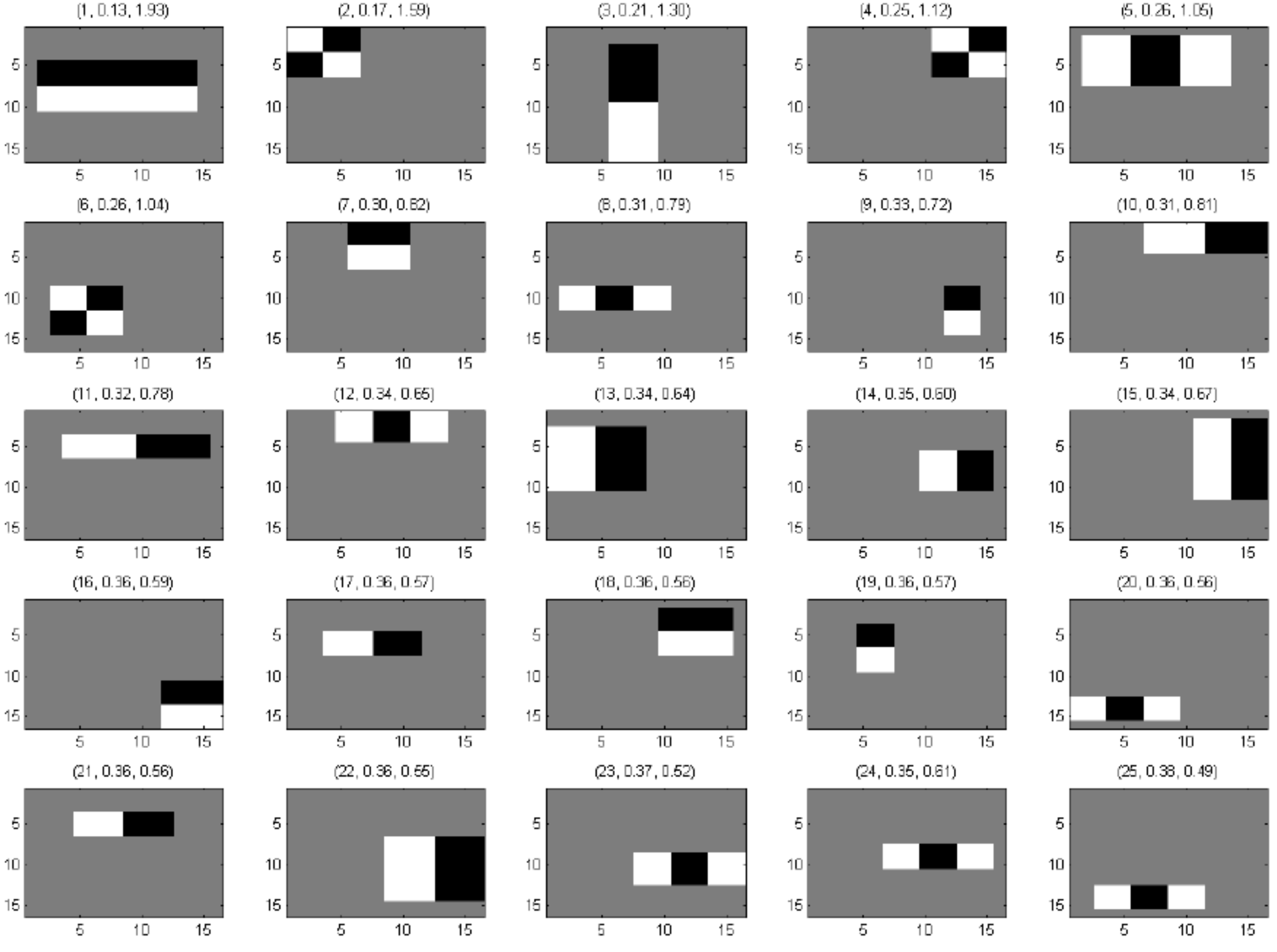

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

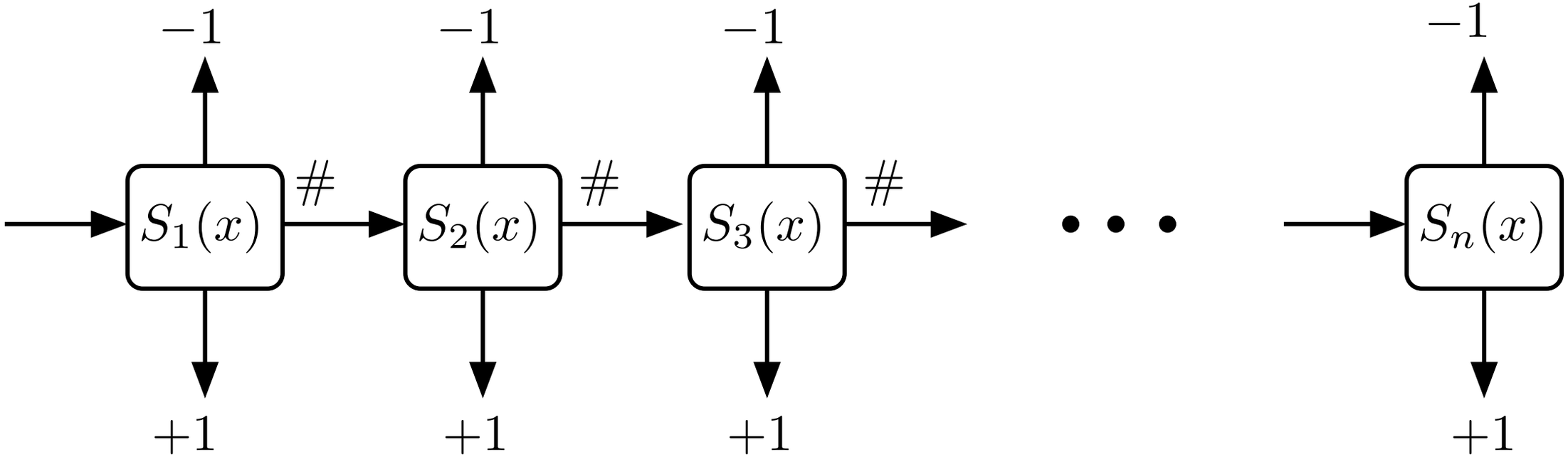



0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40




..... -1
..... +1


(1, 0.13, 1.93)
(2, 0.17, 1.59)
(3, 0.21, 1.30)
(4, 0.25, 1.12)
(5, 0.26, 1.05)
(6, 0.26, 1.04)
(7, 0.30, 0.82)
(8, 0.31, 0.79)
(9, 0.33, 0.72)
(10, 0.31, 0.81)
(11, 0.32, 0.78)
(12, 0.34, 0.65)
(13, 0.34, 0.64)
(14, 0.35, 0.60)
(15, 0.34, 0.67)
(16, 0.36, 0.59)
(17, 0.36, 0.57)
(18, 0.36, 0.56)
(19, 0.36, 0.57)
(20, 0.36, 0.56)
(21, 0.36, 0.56)
(22, 0.36, 0.55)
(23, 0.37, 0.52)
(24, 0.35, 0.61)
(25, 0.38, 0.49)


−1
−1
−1
−1
$S_1(x)$
#
$S_2(x)$
#
$S_3(x)$
#
$S_n(x)$
+1
+1
+1
+1