

Rozpoznávání tváří I

Vojtěch Franc

Centrum strojového vnímání, ČVUT FEL Praha

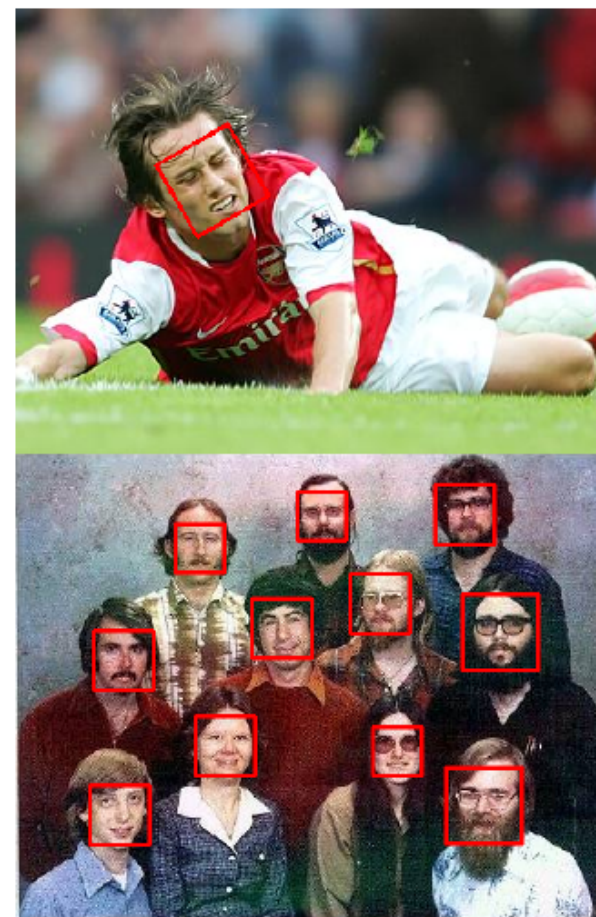
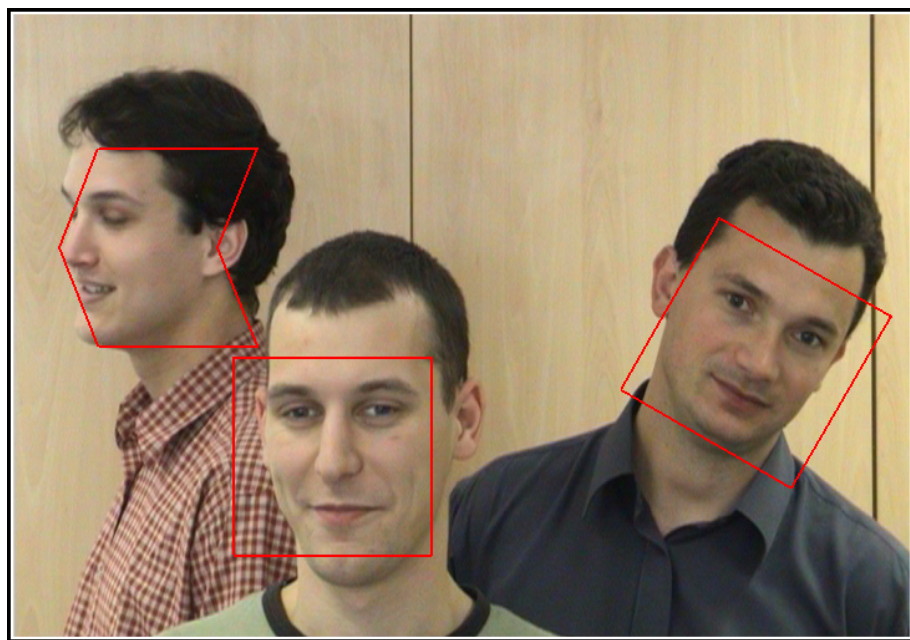


Biometrie ZS 2013

Poděkování Janu Šochmanovi za slajdy vysvětlující AdaBoost

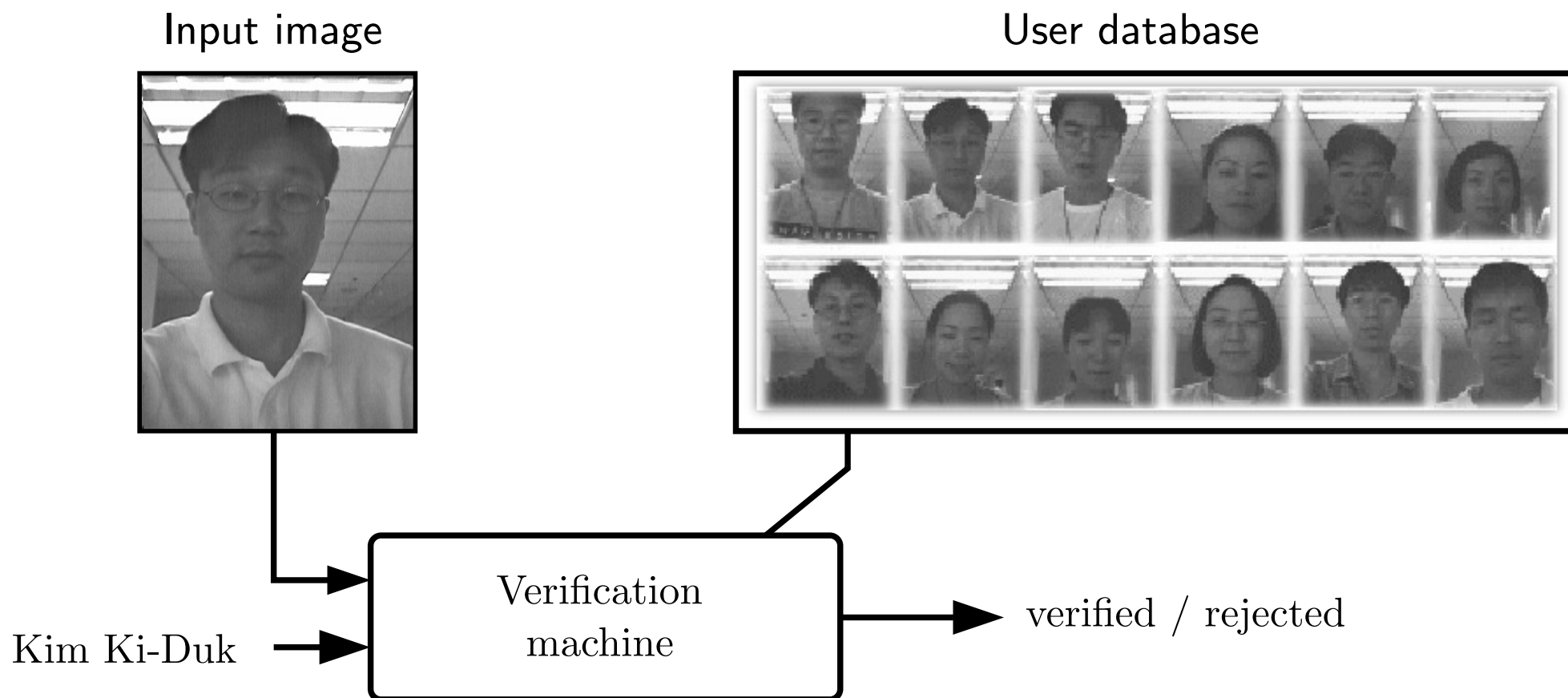
Úlohy rozpoznávání tváří: Detekce

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



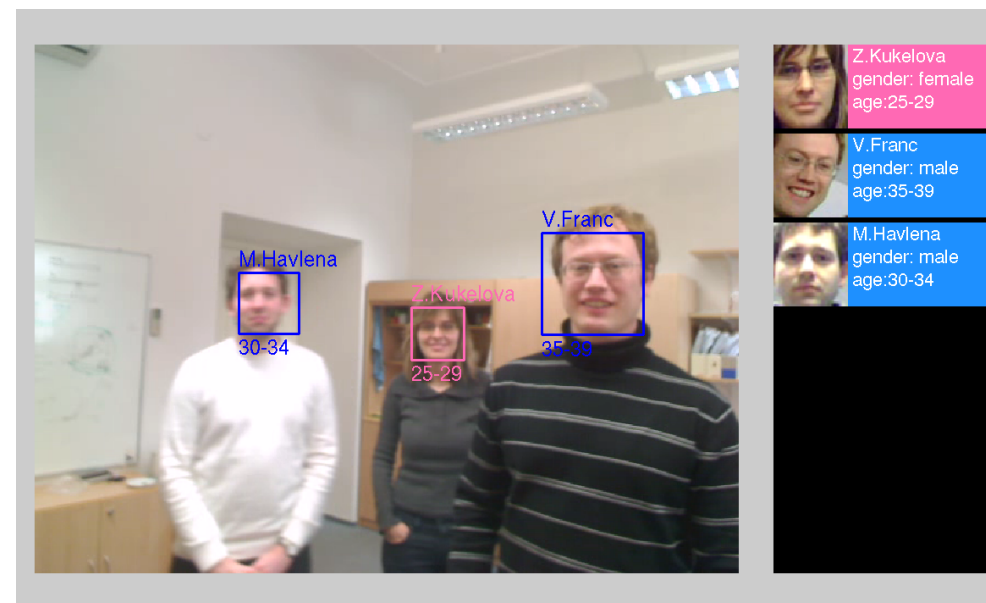
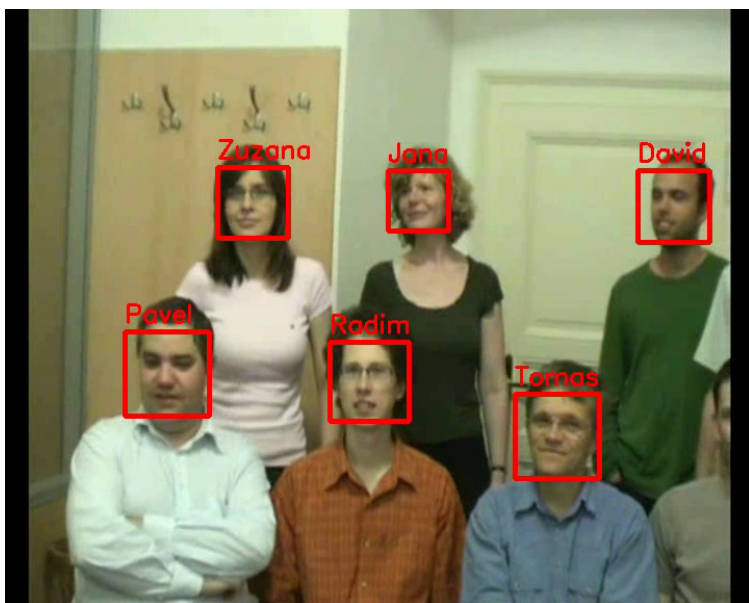
Úlohy rozpoznávání tváří: Verifikace

- ◆ Cíl: ověřit identitu na základě porovnání tváře s databází
- ◆ Vstup: obrázek (nebo video) tváře a jméno
- ◆ Výstup: binární rozhodnutí - identita ověřena nebo zamítnuta



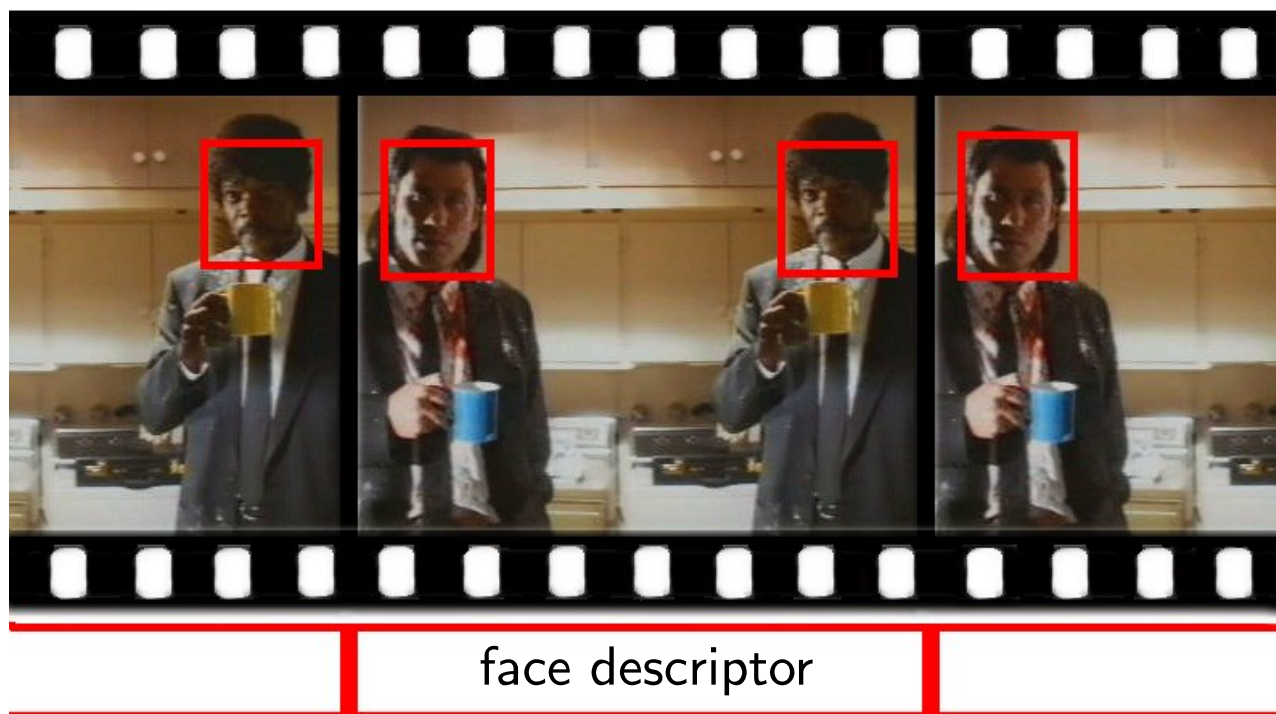
Úlohy rozpoznávání tváří: Identifikace

- ◆ Cíl: přiřadit tváři identitu na základě porovnání s databází
- ◆ Vstup: obrázek (nebo video)
- ◆ Výstup: jedna z N identit popřípadě odpověď není v databázi



Úlohy rozpoznávání tváří: Vyhledávání

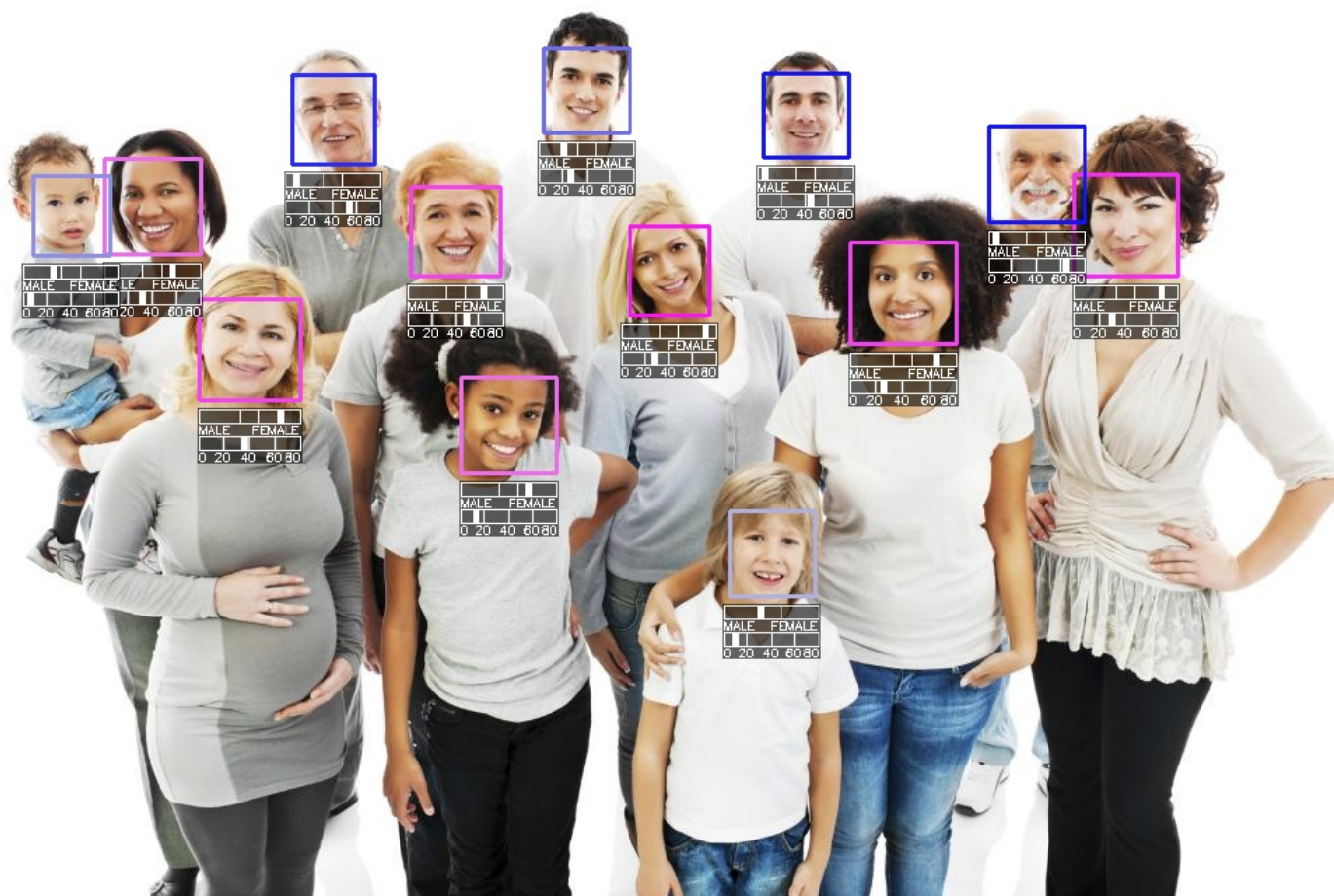
- ◆ Cíl: nalézt v databázi tváře podle zadaného popisu
- ◆ Vstup: obrázek tváře (popřípadě slovní popis)
- ◆ Výstup: N nejbližších obrázků v databázi




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

Úlohy rozpoznávání tváří: Kategorizace

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

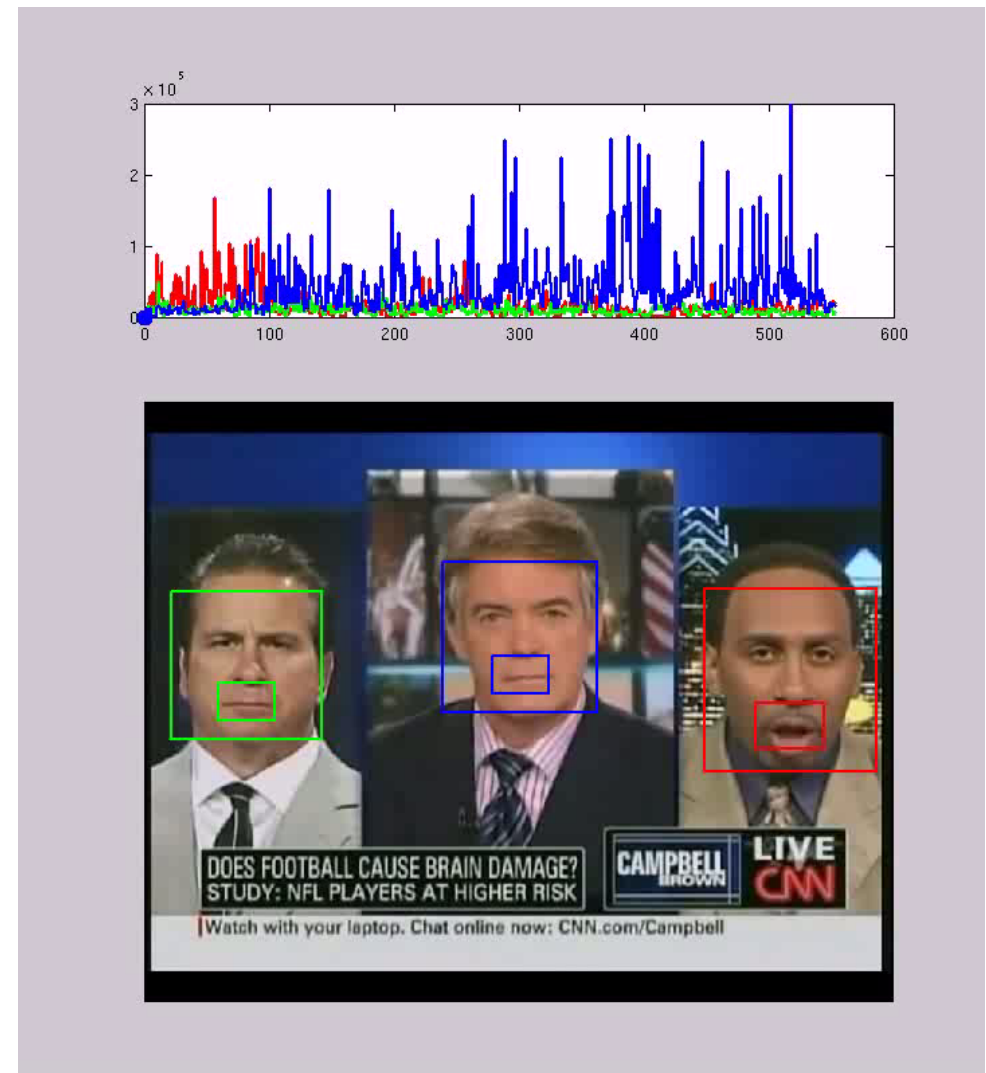


HUMAVIPS
 WP4: Gender Recognition
Proof of concept demo
 Paris, 29 March 2011



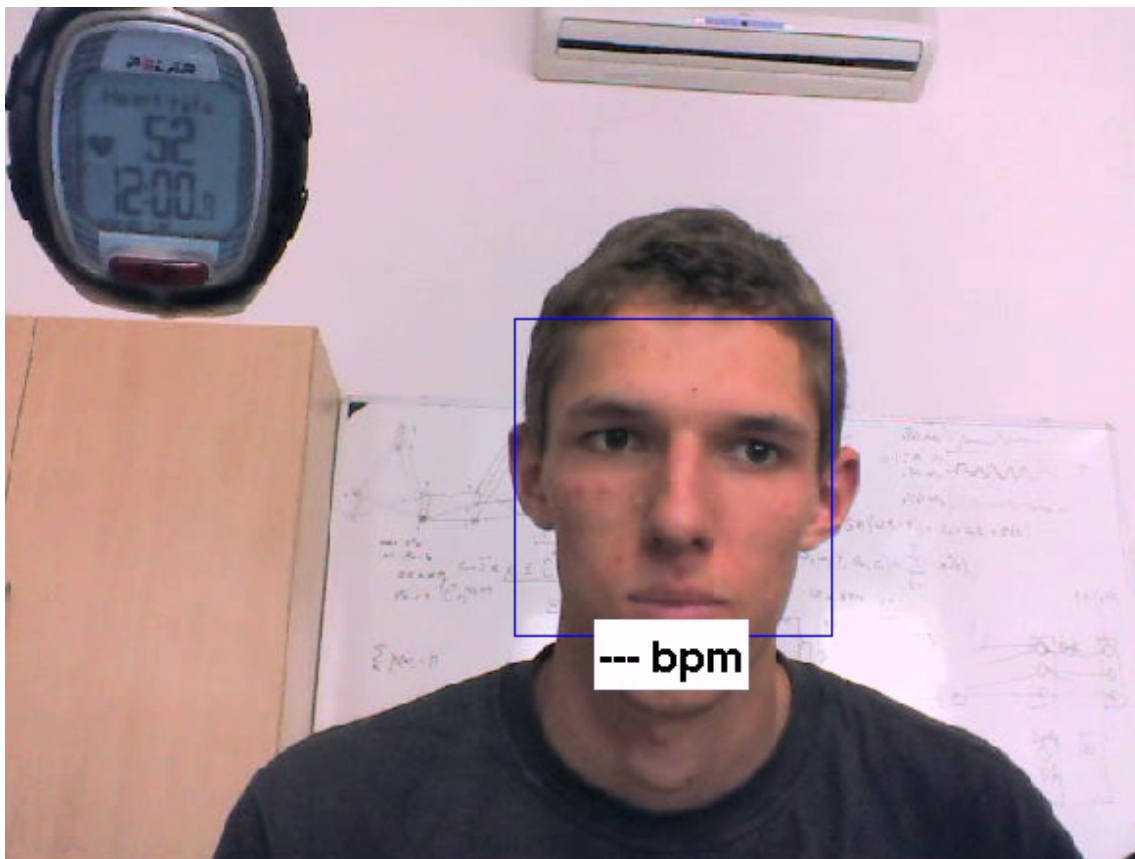
Úlohy rozpoznávání tváří: Detekce mluvčího

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



Úlohy rozpoznávání tváří: Odhad tepové frekvence

- ◆ Cíl: odhadnout tepovou frekvenci sledovaného subjektu
- ◆ Vstup: video tváře
- ◆ Výstup: tepová frekvence



Proč je rozpoznávání tváří v obecném případě těžké?

- ◆ Tváře se typicky reprezentují vektorem příznaků a podobnost tváří se měří podobností těchto vektorů (např. jejich Euklidovskou vzdáleností).
- ◆ Tváře patřící do stejné třídy mají velkou variabilitu:
 - změna pozice, měřítko, rotace (roll, pan, tilt)
 - změna osvětlení
 - změna výrazu tváře (úsměv, smutek, neutrální, ...)
 - zákryty (brýle, pokrývka hlavy), změna účesu, make up, stárnutí ...



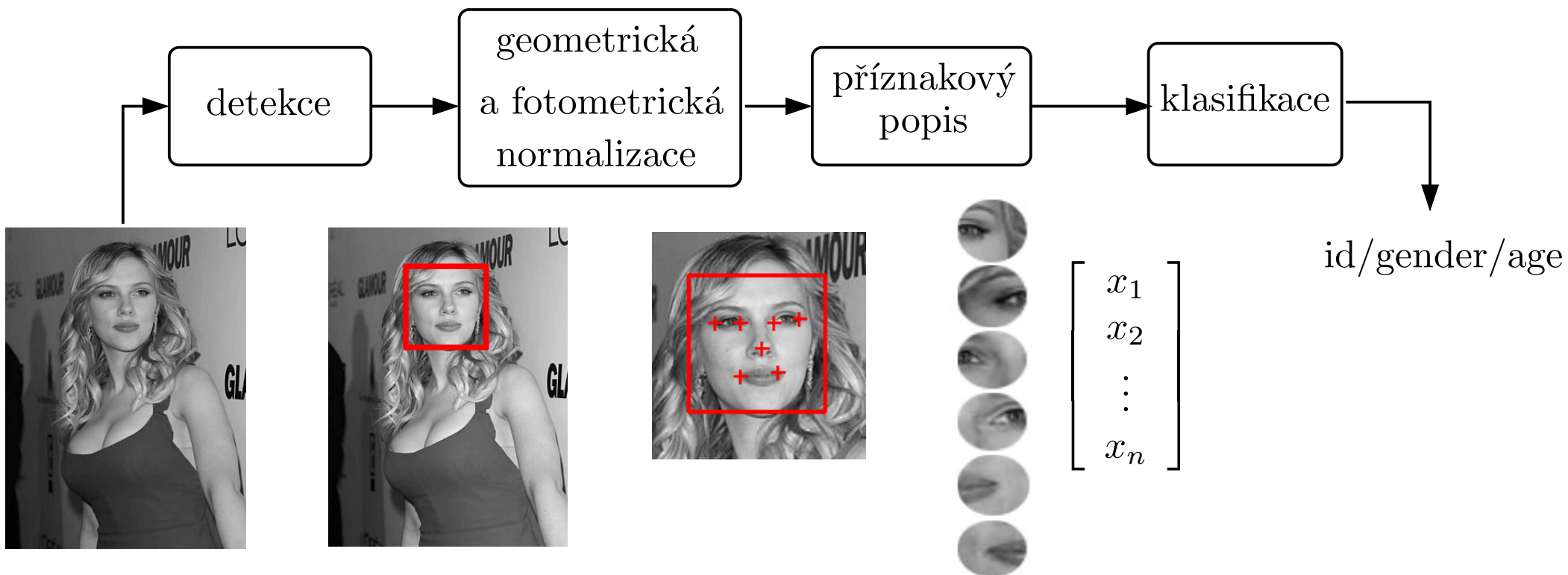
Složitost rozpoznávání tváří je ovlivněna mnoha faktory

- ◆ Světelné podmínky (venkovní/vnitřní prostor, stíny, přesevětlení obrazu ...)
- ◆ Použitý snímací senzor:
 - kamera pracující ve viditelném spektru
 - infračervená kamera (s IR přísvícením)
 - stereo kamera
 - 3D skener
- ◆ Statický obrázek / video sekvence
- ◆ Spolupracující / nespolupracující subjekt
- ◆ . . .



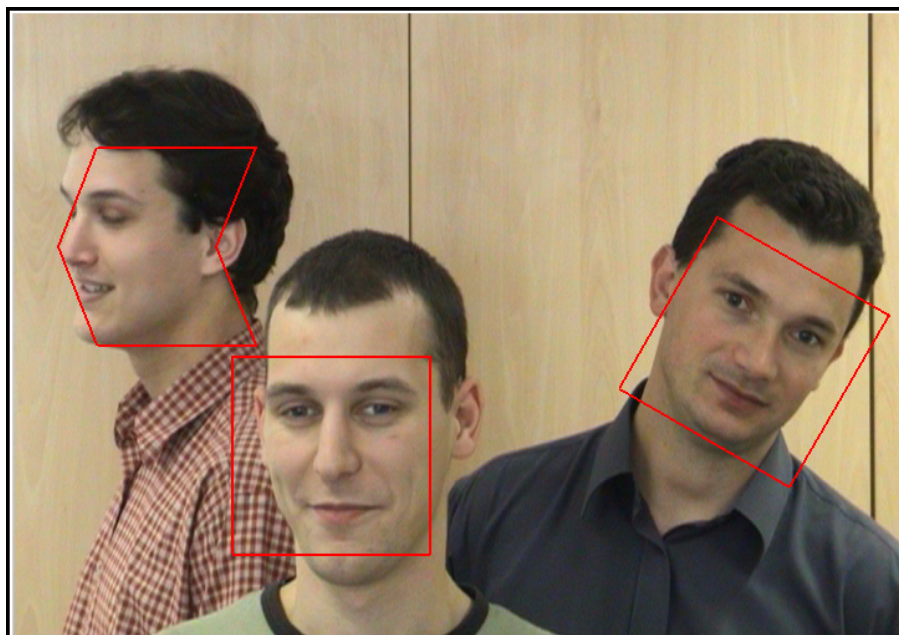
Příklad rozpoznávacího systému, stavební bloky

- ◆ Geometrická a fotometrická normalizace:
 - cílem je získat reprezentaci tváře invariantní vůči geometrické transformaci či změně osvětlení
 - geometrická normalizace se typicky řeší nalezením významných bodů na tváři
- ◆ Příznakový popis: tvář se popíše vektorem čísel (jasové hodnoty, popis textury, ...)
- ◆ Klasifikace: zařazení vektoru příznaků do třídy



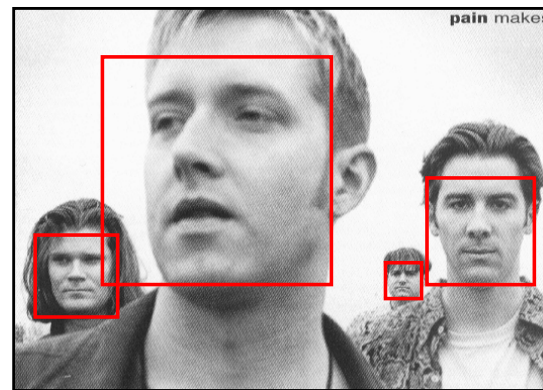
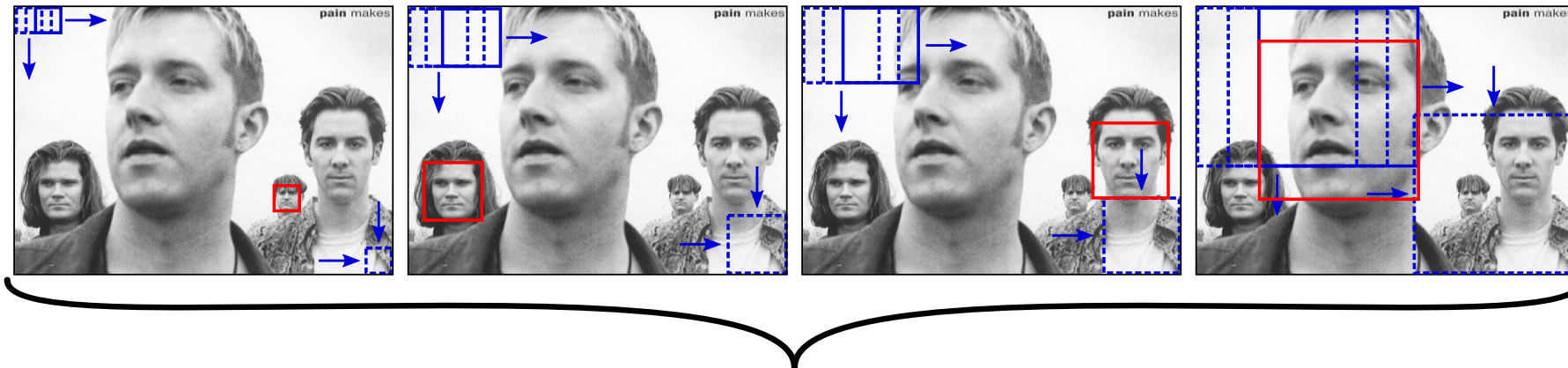
Detektor tváří

- ◆ Vstup detektoru: obrázek $x \in X$
- ◆ Výstup detektoru: množina $d \in D$ oblastí v obrázku x , které obsahují tváře; oblast je popsána polohou a velikostí (popřípadě orientací) tváře
- ◆ Kritéria pro hodnocení detektoru $Q: X \rightarrow D$: **přesnost** a **rychlost**
- ◆ Přesnost detektoru tváří se charakterizuje dvěma čísly:
 1. **True Positive Rate (TPR)**: pravděpodobnost, že tvář je v obraze nalezena
 2. **False Positive Rate (FPR)**: pravděpodobnost, že nalezená oblast neobsahuje tvář



Scanning window approach

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



Těžký rozhodovací problém $Q: X \rightarrow D$ převedeme na mnoho jednodušších $H: X \rightarrow \{-1, +1\}$

Binární klasifikátor tváří

- ◆ Klasifikátor $h: X \rightarrow \{+1, -1\}$ (tvář/netvář) rozhoduje podle znaménka skórovací funkce $f: X \rightarrow \mathcal{R}$,

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

kteřá je lineární kombinací n slabých klasifikátorů

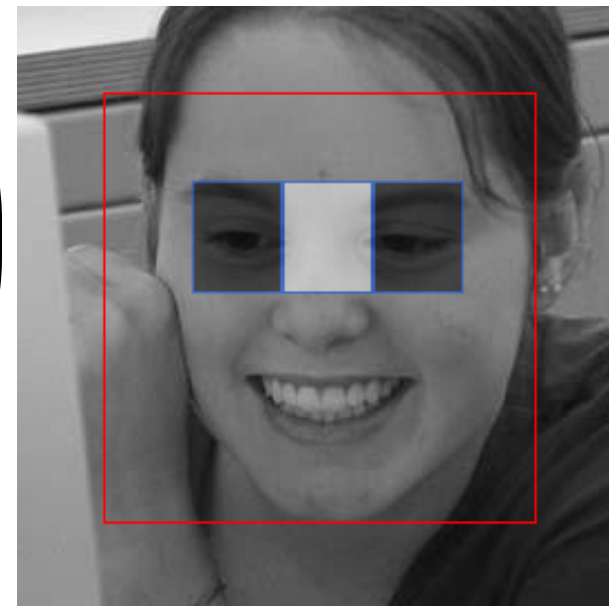
$$f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

- ◆ $h_i: X \rightarrow \{+1, -1\}$... i -tý slabý klasifikátor
- ◆ $\alpha_i \in \mathcal{R}$... váha i -tého slabého klasifikátoru

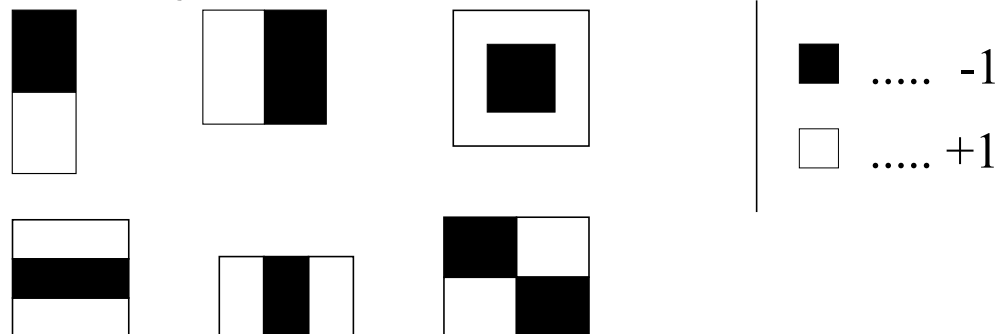
Slabý klasifikátor pro detektor tváří

- ◆ 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

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

Učení klasifikátorů

- ◆ Jak vybrat takovou podmnožinu slabých klasifikátorů a jejich váhy, abychom výsledný silný klasifikátor byl přesný?

$$H(x) = \begin{cases} +1 & \text{pro } f(x) \geq 0 \\ -1 & \text{pro } f(x) < 0 \end{cases} \quad \text{kde } f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

Učení klasifikátorů

- ◆ Jak vybrat takovou podmnožinu slabých klasifikátorů a jejich váhy, abychom výsledný silný klasifikátor byl přesný?

$$H(x) = \begin{cases} +1 & \text{pro } f(x) \geq 0 \\ -1 & \text{pro } f(x) < 0 \end{cases} \quad \text{kde } f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

- ◆ Klasifikátor se učí sám z množiny trénovacích příkladů $\{(x_1, y_1), \dots, (x_m, y_m)\}$
 $(x_i, y_i = +1)$ $(x_i, y_i = -1)$



Učení klasifikátorů

- ◆ Jak vybrat takovou podmnožinu slabých klasifikátorů a jejich váhy, abychom výsledný silný klasifikátor byl přesný?

$$H(x) = \begin{cases} +1 & \text{pro } f(x) \geq 0 \\ -1 & \text{pro } f(x) < 0 \end{cases} \quad \text{kde } f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

- ◆ Klasifikátor se učí sám z množiny trénovacích příkladů $\{(x_1, y_1), \dots, (x_m, y_m)\}$
 $(x_i, y_i = +1)$ $(x_i, y_i = -1)$



- ◆ Učení: optimalizační úloha kdy hledáme vektor vah $\alpha \in \mathcal{R}^L$, tak aby trénovací chyba

$$P(\alpha) = \sum_{i=1}^m [H(x_i, \alpha) \neq y_i]$$

byla minimální za podmínky, že počet nenulových komponent vektoru α je roven n .

Učení klasifikátorů

- ◆ Jak vybrat takovou podmnožinu slabých klasifikátorů a jejich váhy, abychom výsledný silný klasifikátor byl přesný?

$$H(x) = \begin{cases} +1 & \text{pro } f(x) \geq 0 \\ -1 & \text{pro } f(x) < 0 \end{cases} \quad \text{kde } f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

- ◆ Klasifikátor se učí sám z množiny trénovacích příkladů $\{(x_1, y_1), \dots, (x_m, y_m)\}$
 $(x_i, y_i = +1)$ $(x_i, y_i = -1)$



- ◆ Učení: optimalizační úloha kdy hledáme vektor vah $\alpha \in \mathcal{R}^L$, tak aby trénovací chyba

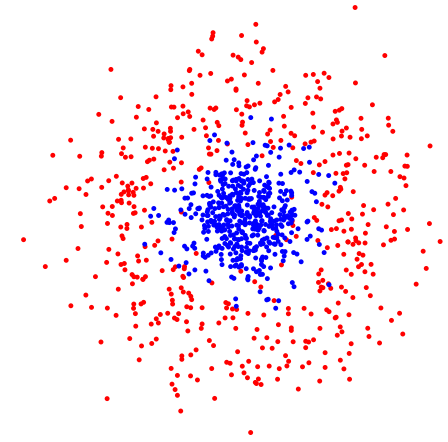
$$P(\alpha) = \sum_{i=1}^m [H(x_i, \alpha) \neq y_i]$$

byla minimální za podmínky, že počet nenulových komponent vektoru α je roven n .

- ◆ AdaBoost: hladová minimalizace horní meze $P(\alpha) \leq U(\alpha) = \sum_{i=1}^m \exp(-y_i H(x_i, \alpha))$.

AdaBoost algoritmus

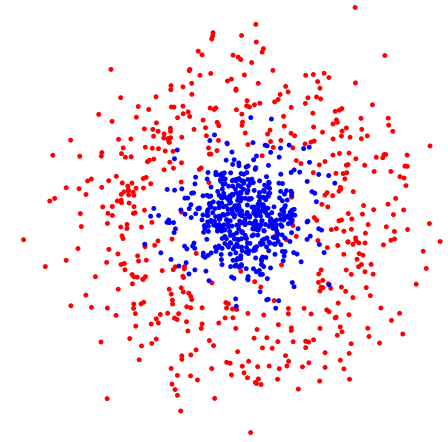
Given: $(x_1, y_1), \dots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$



AdaBoost algoritmus

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

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

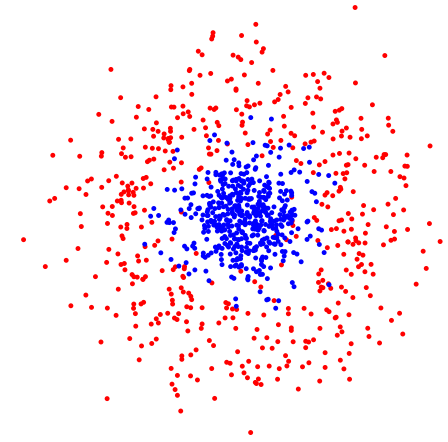


AdaBoost algoritmus

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

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

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



AdaBoost algorithmus

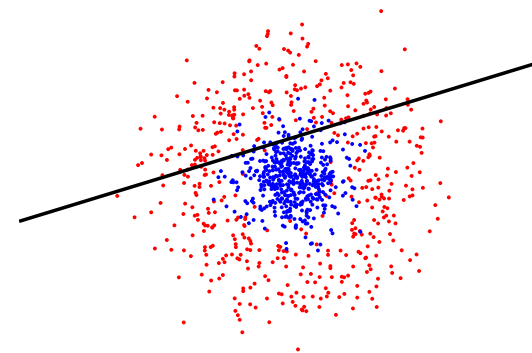
Given: $(x_1, y_1), \dots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

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

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

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

$t = 1$



AdaBoost algorithmus

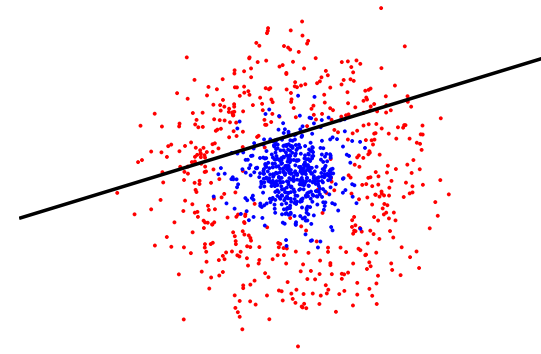
Given: $(x_1, y_1), \dots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

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

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

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

$t = 1$



AdaBoost algorithmus

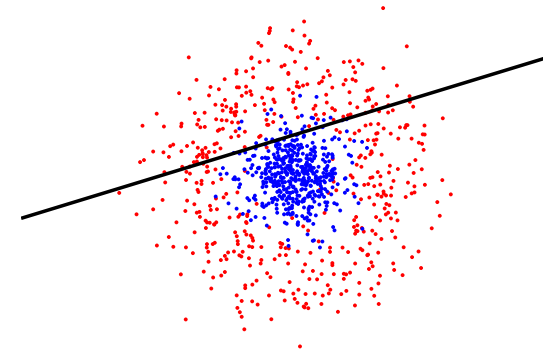
Given: $(x_1, y_1), \dots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

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

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

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

$t = 1$



AdaBoost algoritmus

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

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

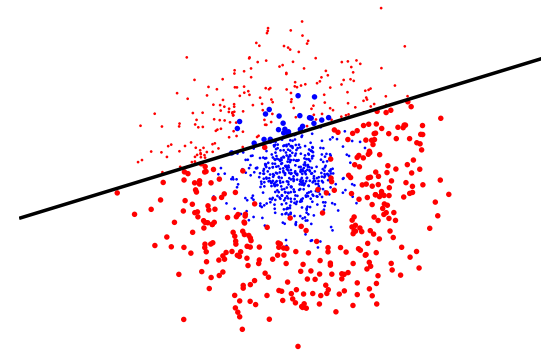
For $t = 1, \dots, T$:

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

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

where Z_t is normalisation factor

$t = 1$



AdaBoost algorithmus

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

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

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

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

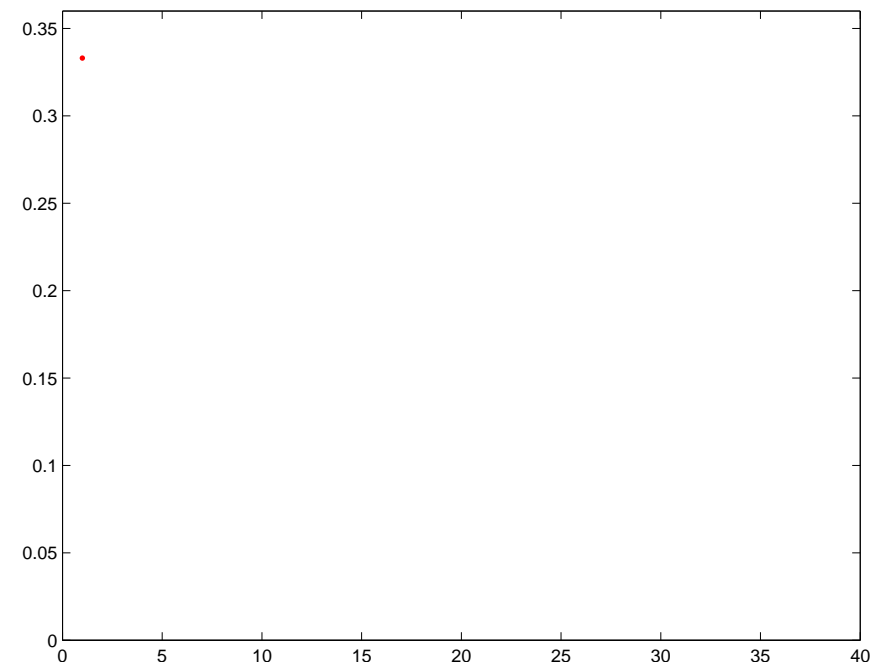
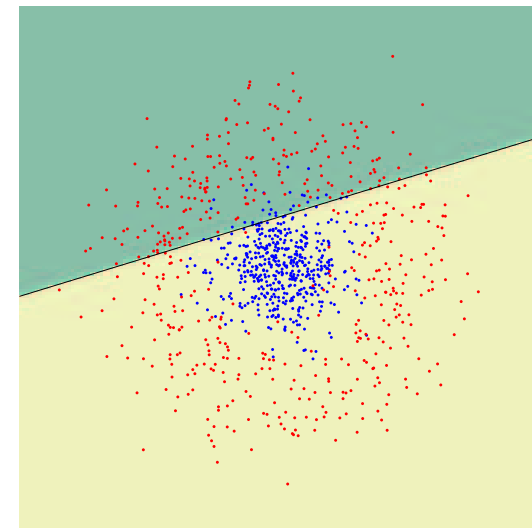
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) \right)$$

$t = 1$



AdaBoost algorithmus

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

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

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

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

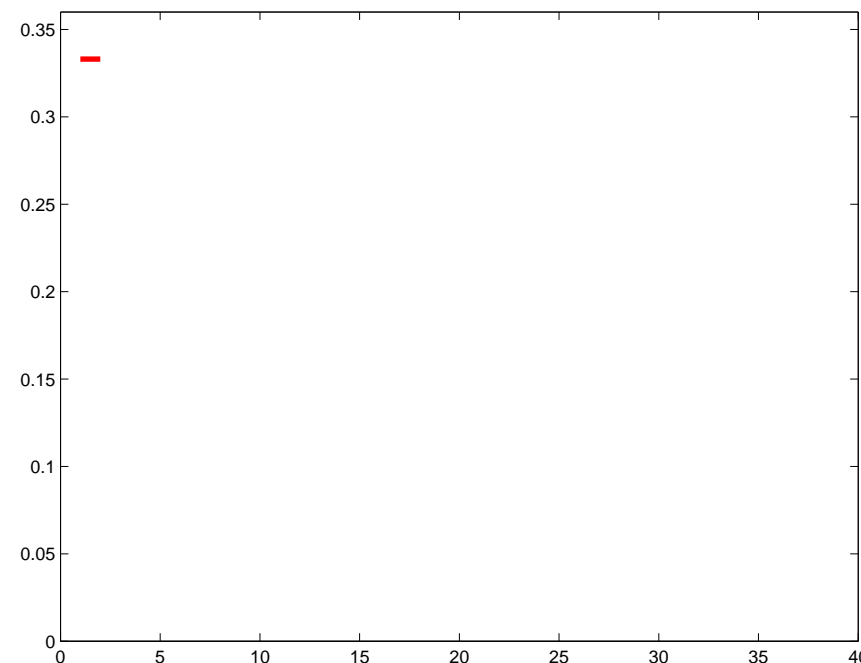
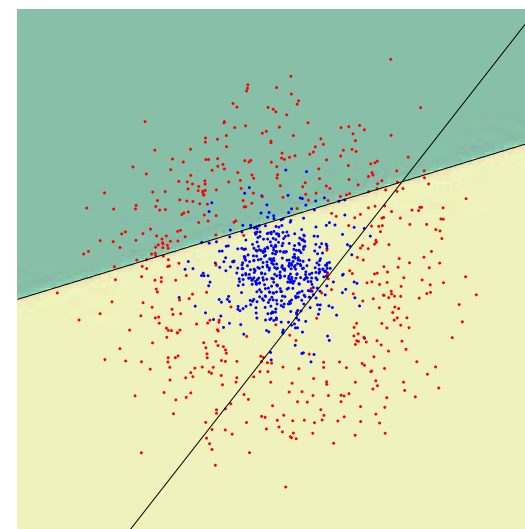
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) \right)$$

$t = 2$



AdaBoost algorithmus

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

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

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

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

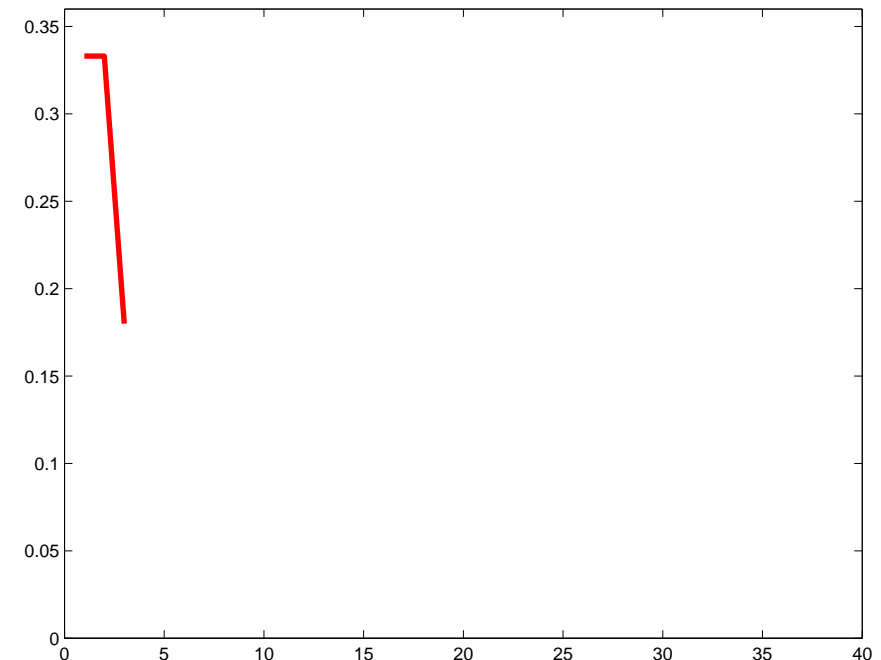
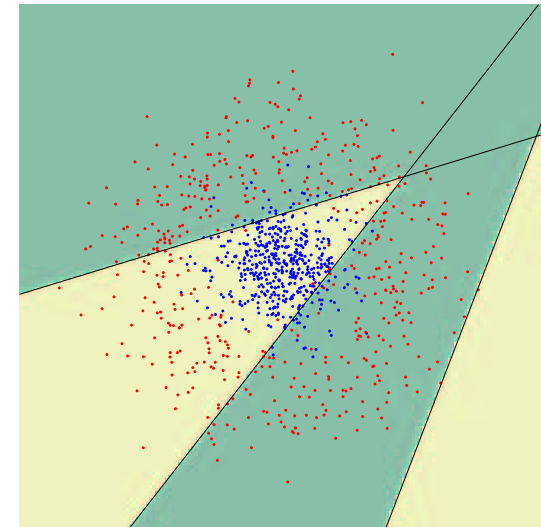
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) \right)$$

$t = 3$



AdaBoost algorithmus

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

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

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

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

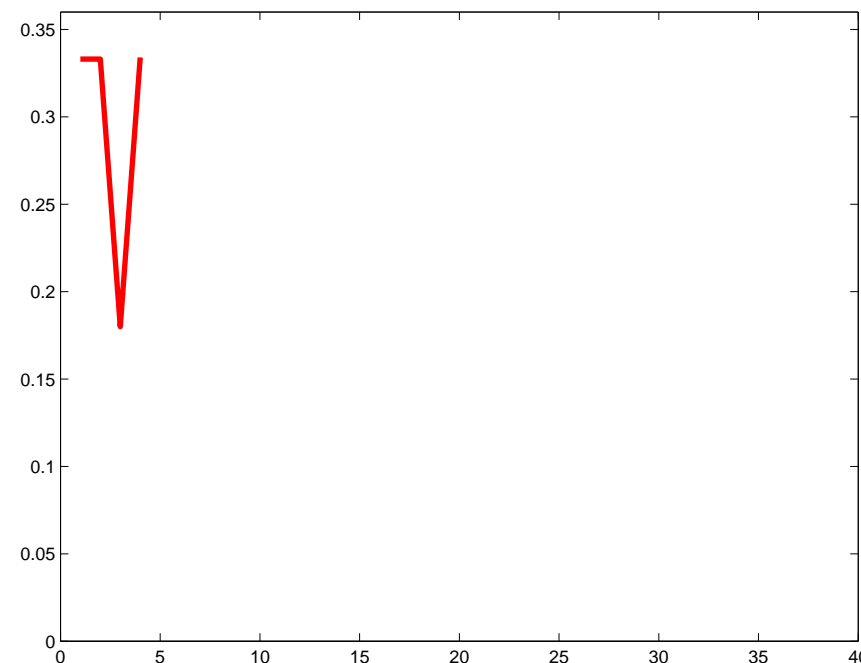
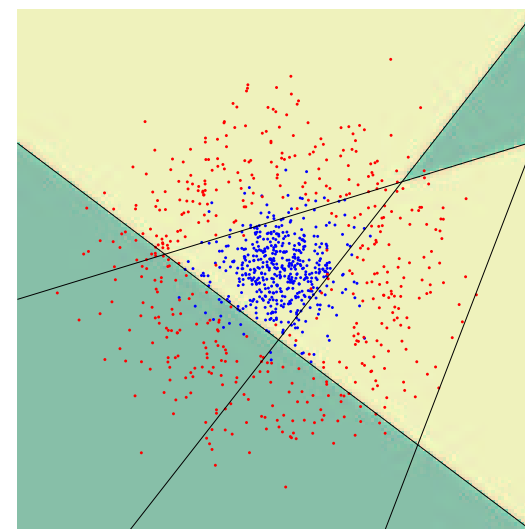
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_4 h_4(x) \right)$$

$t = 4$



AdaBoost algorithmus

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

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

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

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

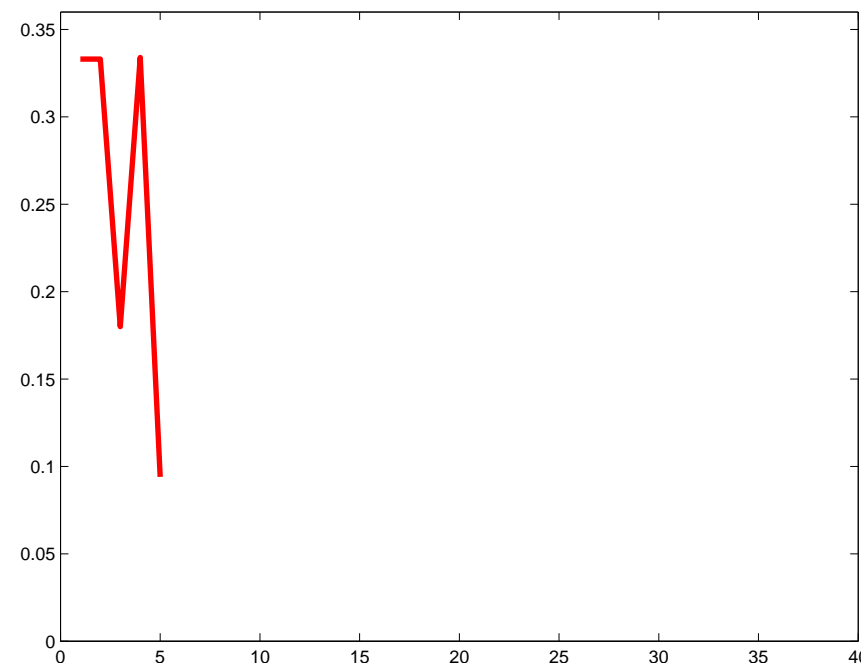
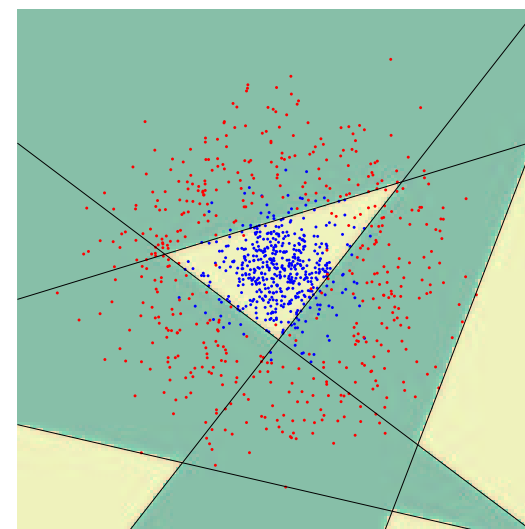
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_5 h_5(x) \right)$$

$t = 5$



AdaBoost algorithmus

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

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

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

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

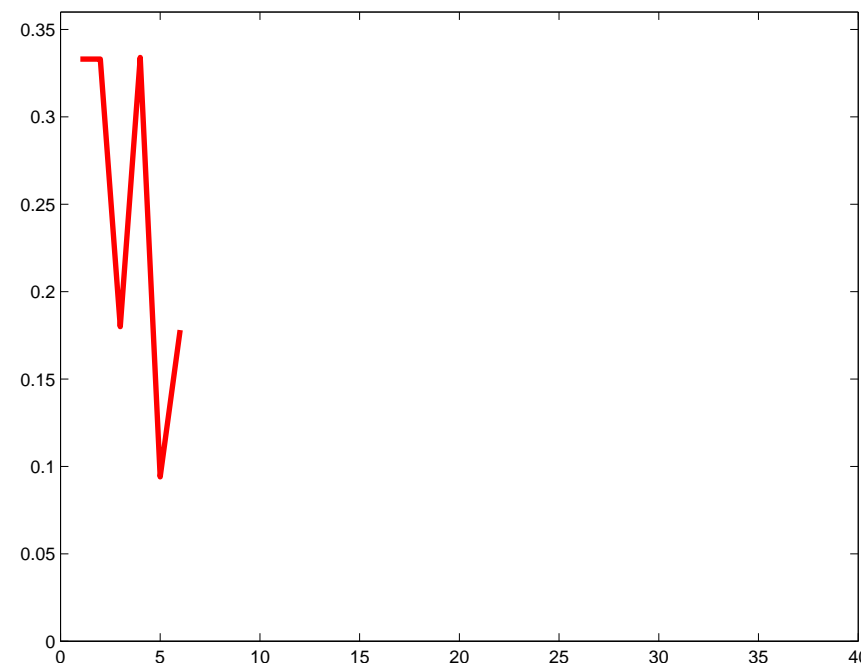
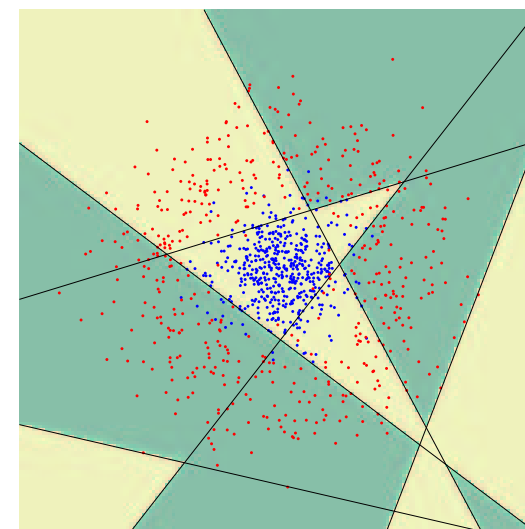
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_6 h_6(x) \right)$$

$t = 6$



AdaBoost algoritmus

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

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

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

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

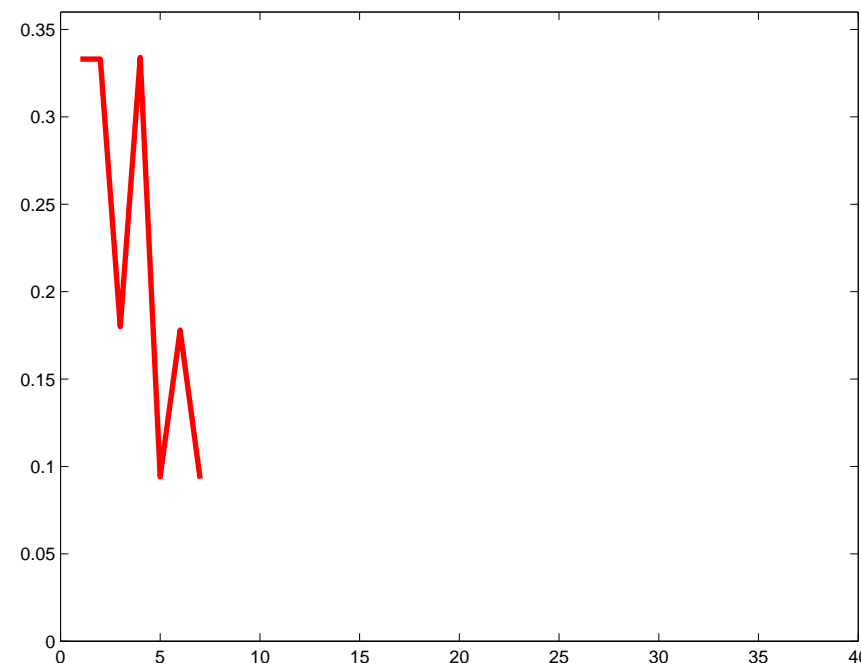
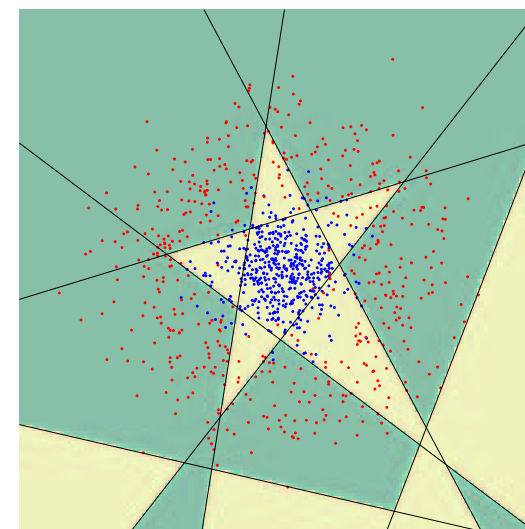
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_7 h_7(x) \right)$$

$t = 7$



AdaBoost algorithmus

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

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

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

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

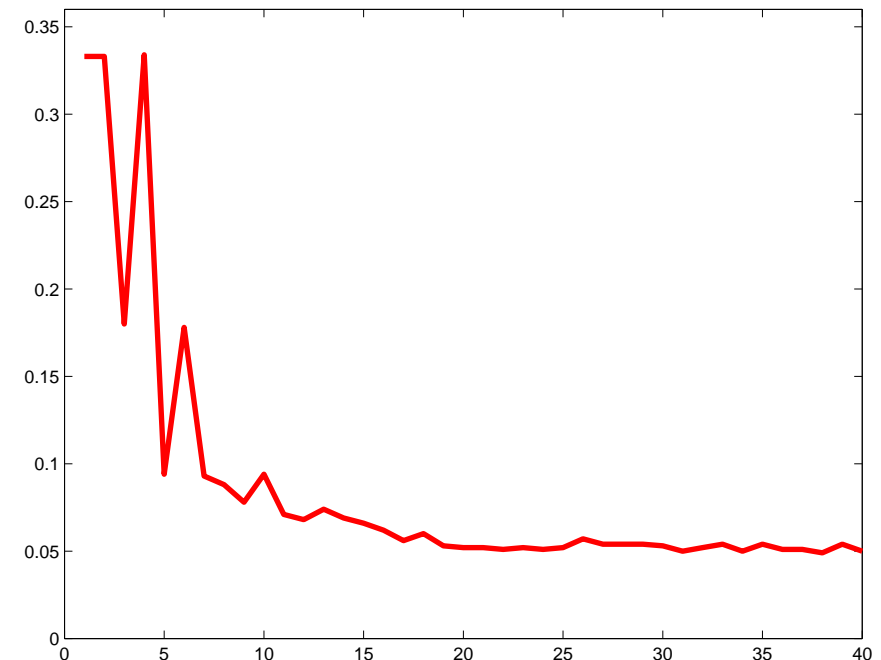
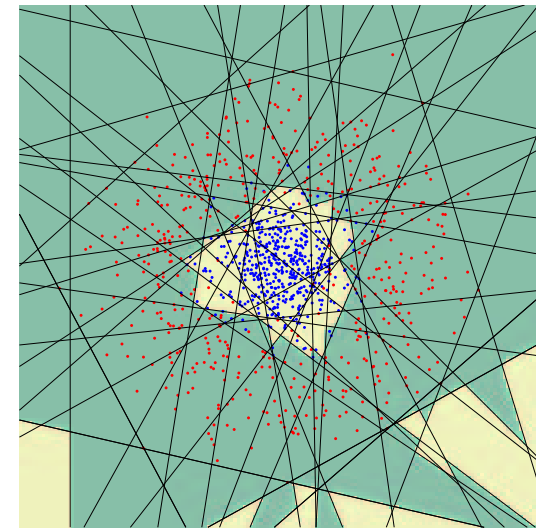
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is normalisation factor

Output the final classifier:

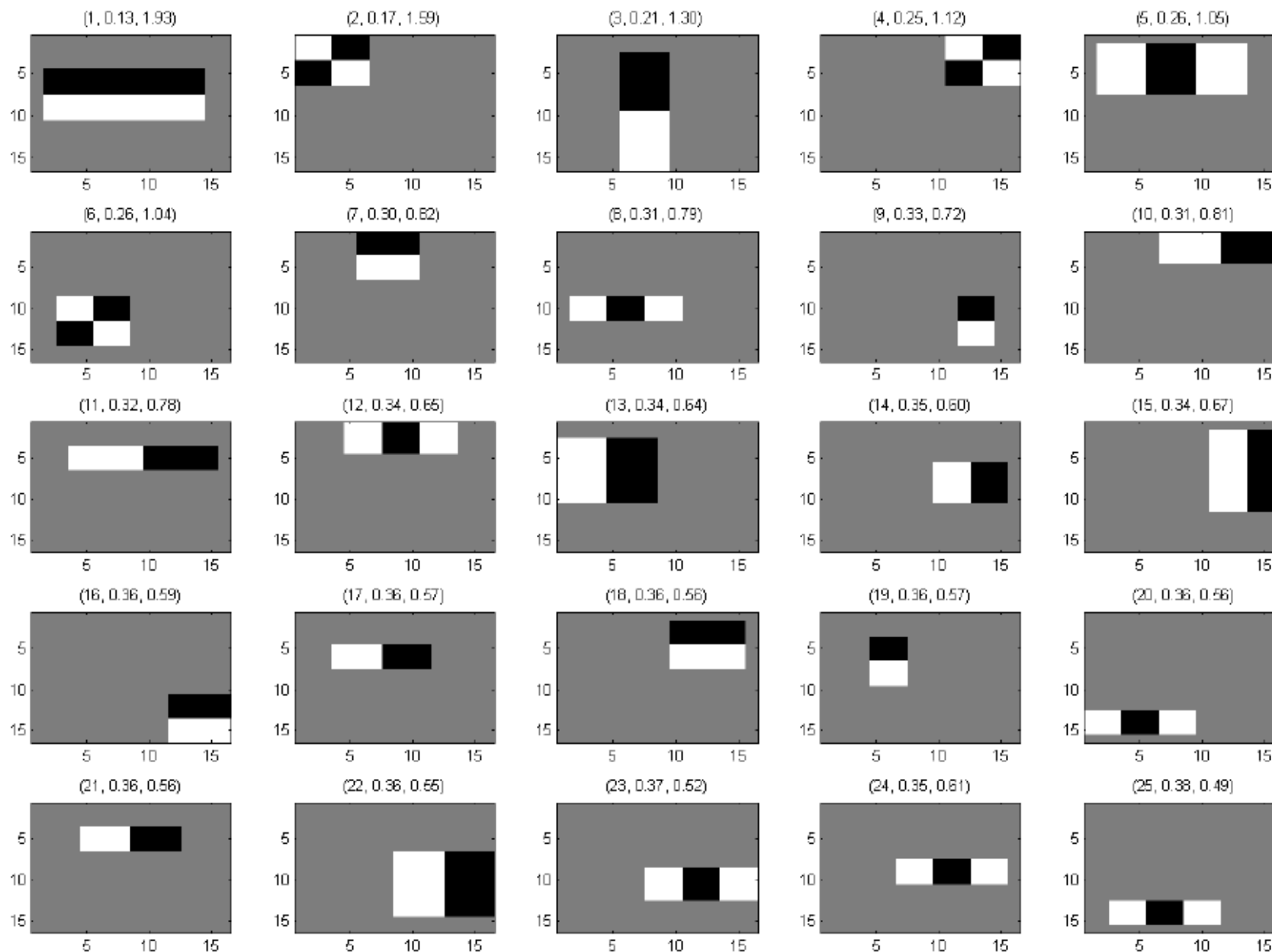
$$H(x) = \text{sign} \left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_{40} h_{40}(x) \right)$$

$t = 40$



Příklady nalezených slabých klasifikátorů

Po prvních 25 iteracích.



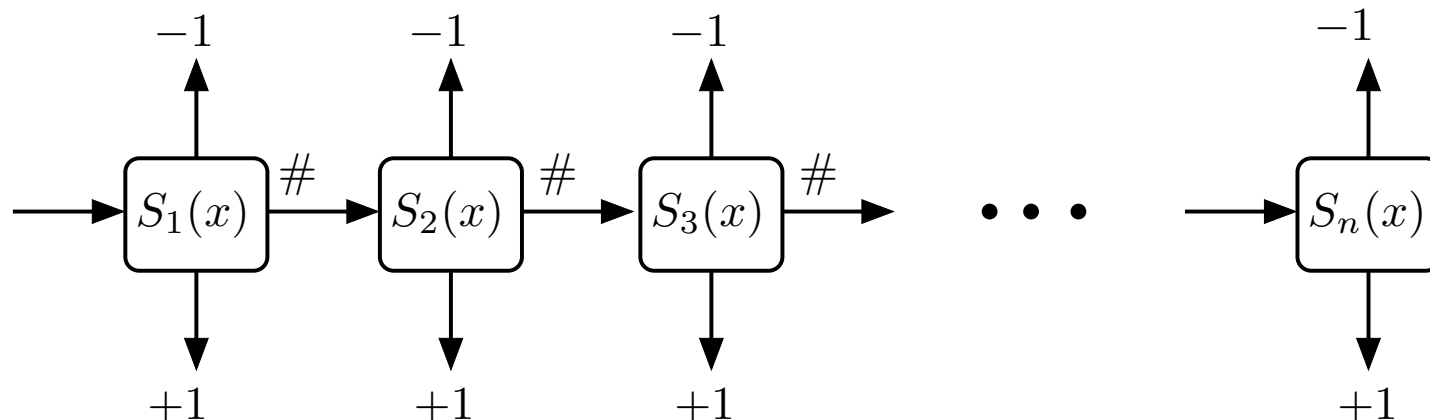
Urychlení detekce pomocí sekvenčního rozhodování

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

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

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

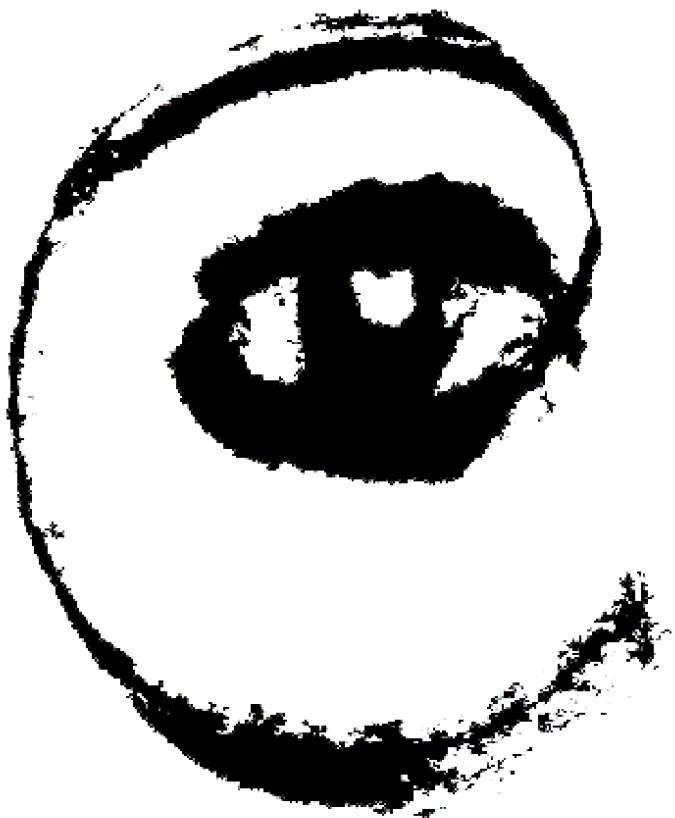
$$S_t(x) = \begin{cases} +1 & f_t(x) \geq \theta_t^A \\ -1 & f_t(x) \leq \theta_t^B \\ \# & \theta_t^B < f_t(x) < \theta_t^A \end{cases} \quad f_t(x) = \sum_{i=1}^t \alpha_i h_i(x)$$



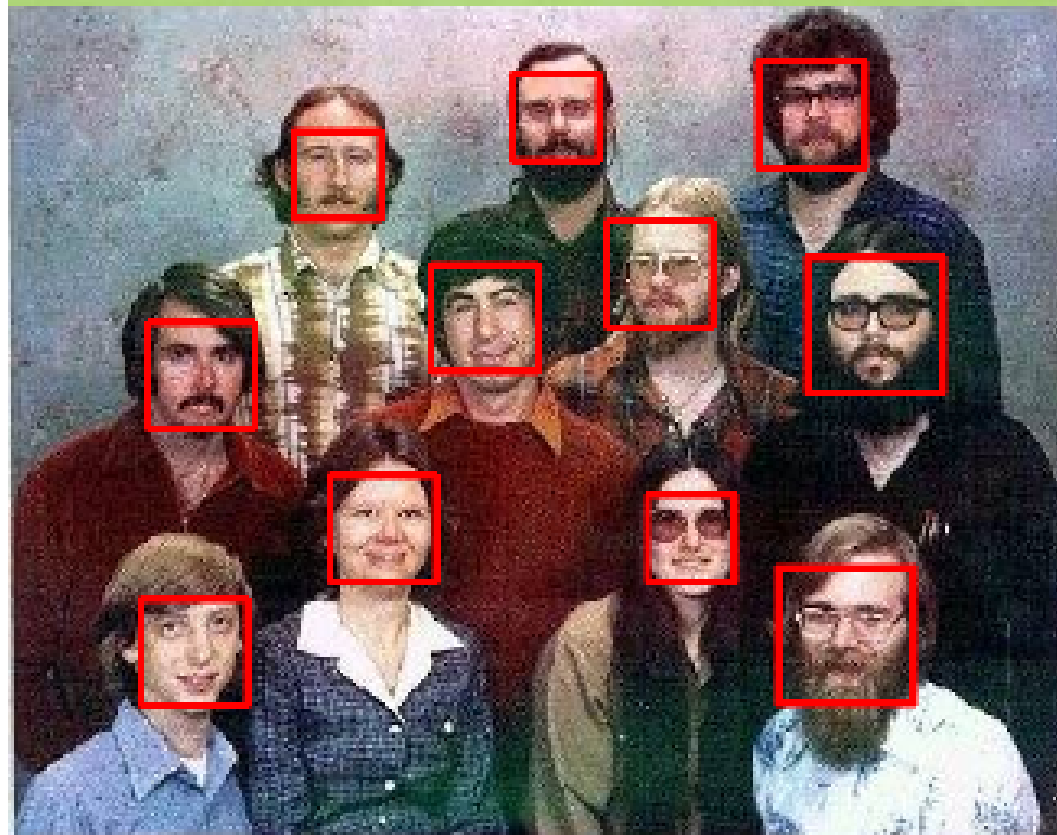
Příklad komerčního detektoru tváří (www.eyedea.cz)

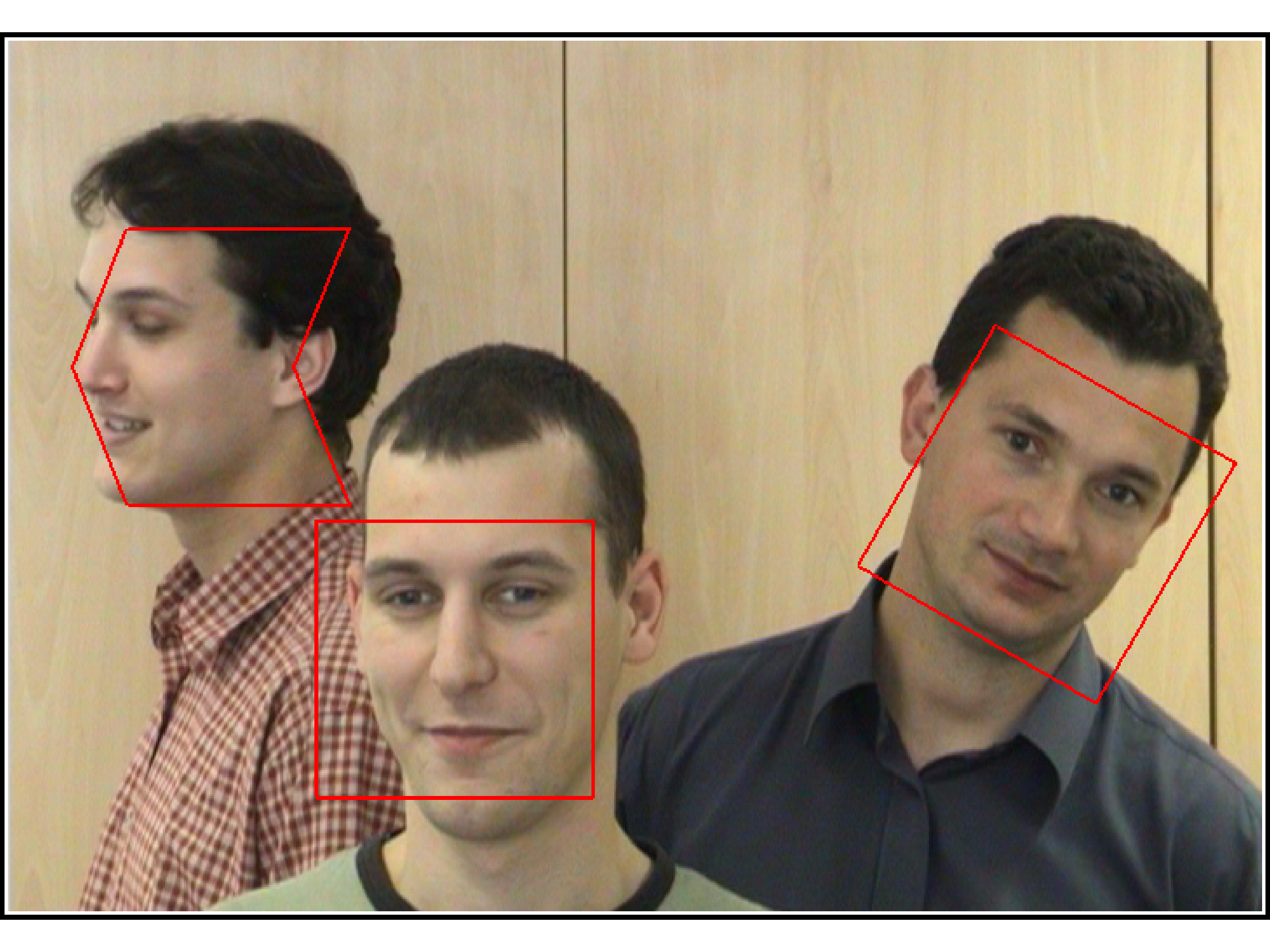
- ◆ Trénovací množina vytvořena tak, aby pokryla co nejvíce variance (rasa, osvětlení, výraz tváře, pozadí obrázku...)
 - **Pozitivní příklady (tváří)**: více než 500,000
 - * Synteticky generované z cca 60,000 tváří aplikováním transformací, které nemění třídu obrázku - malá změna rotace, změna měřítka, 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 640×480 px a minimální velikosti tváře 24×24 px zpracuje detektor cca 10 – 15 snímků za vteřinu.

Konec



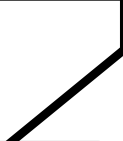
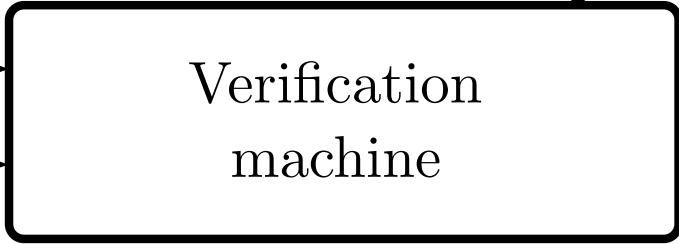
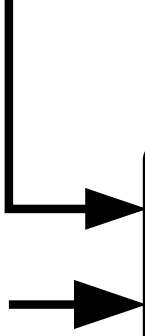
m p







Kim Ki-Duk



verified / rejected















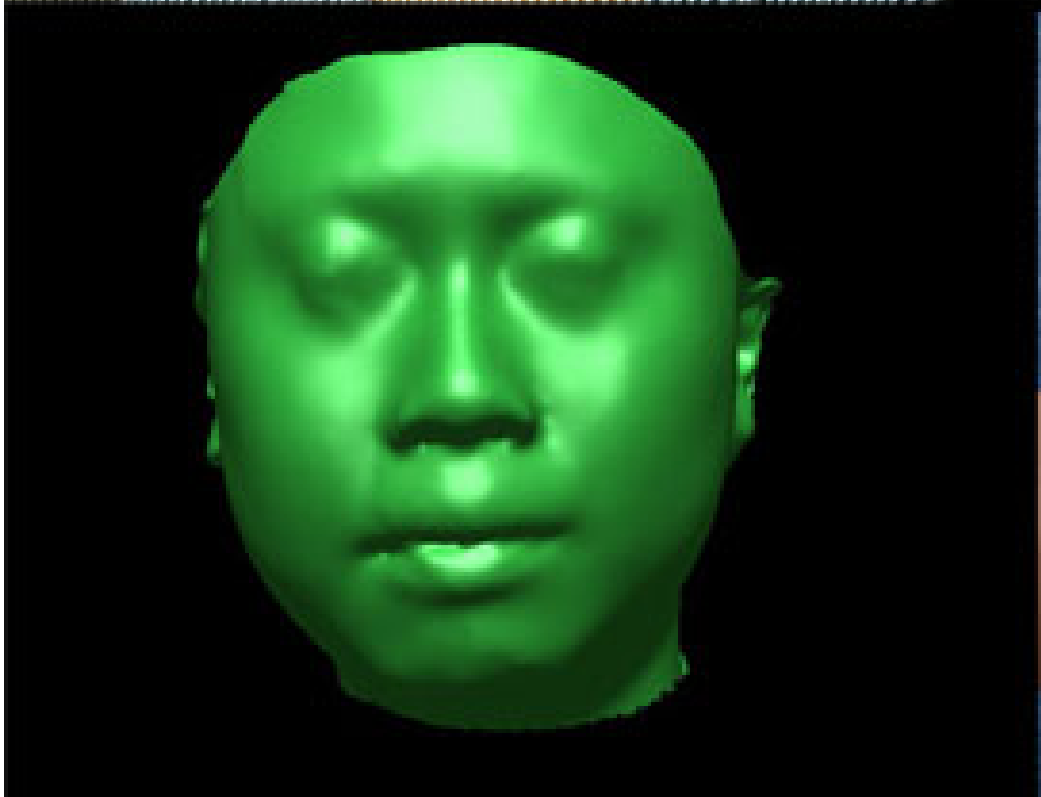


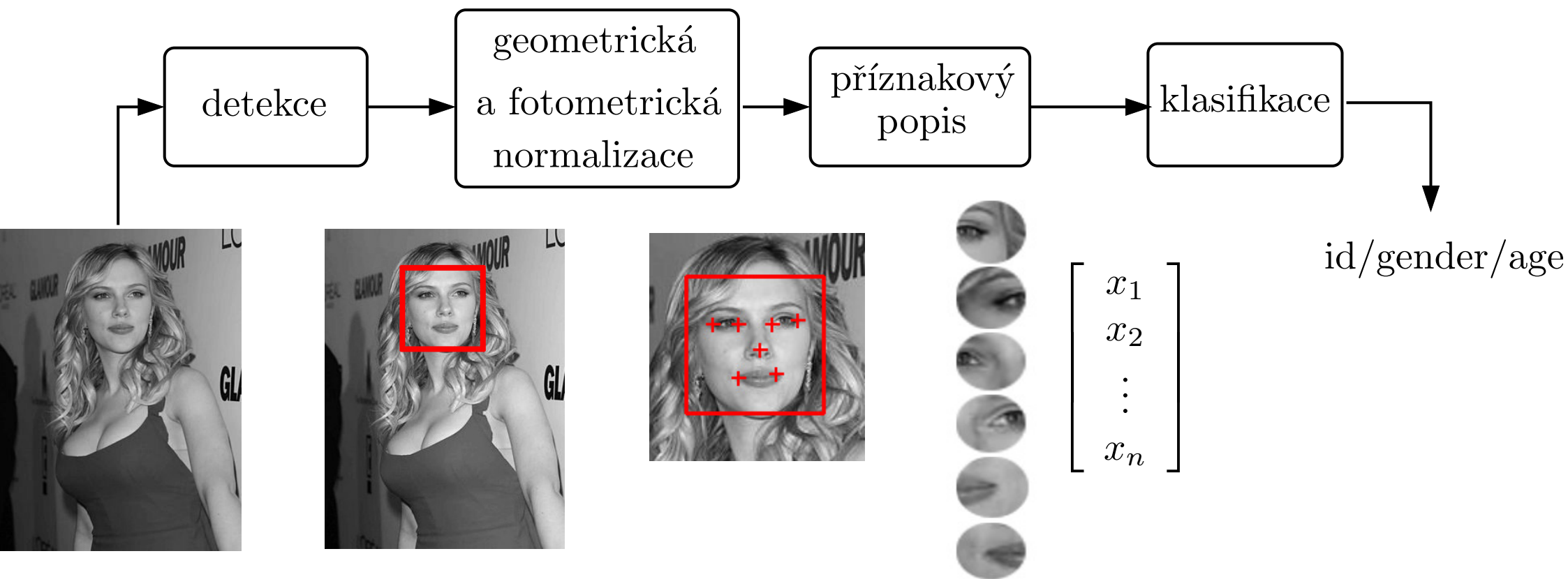


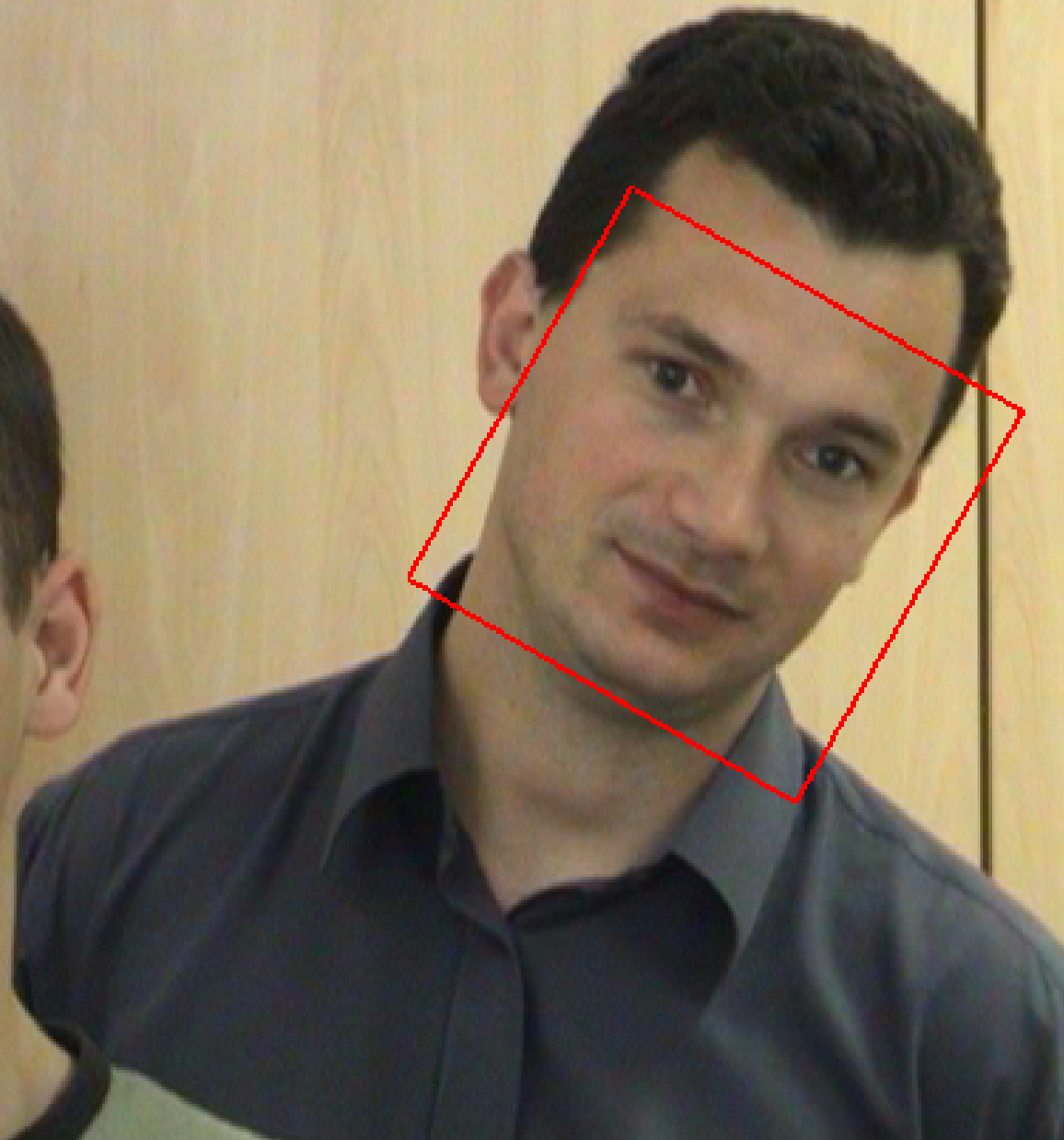
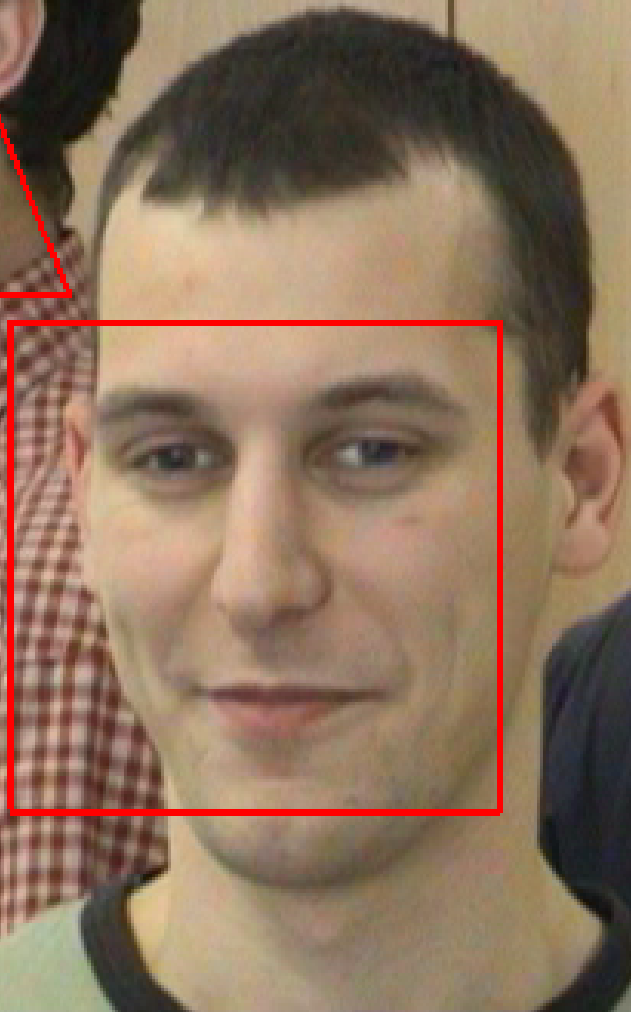
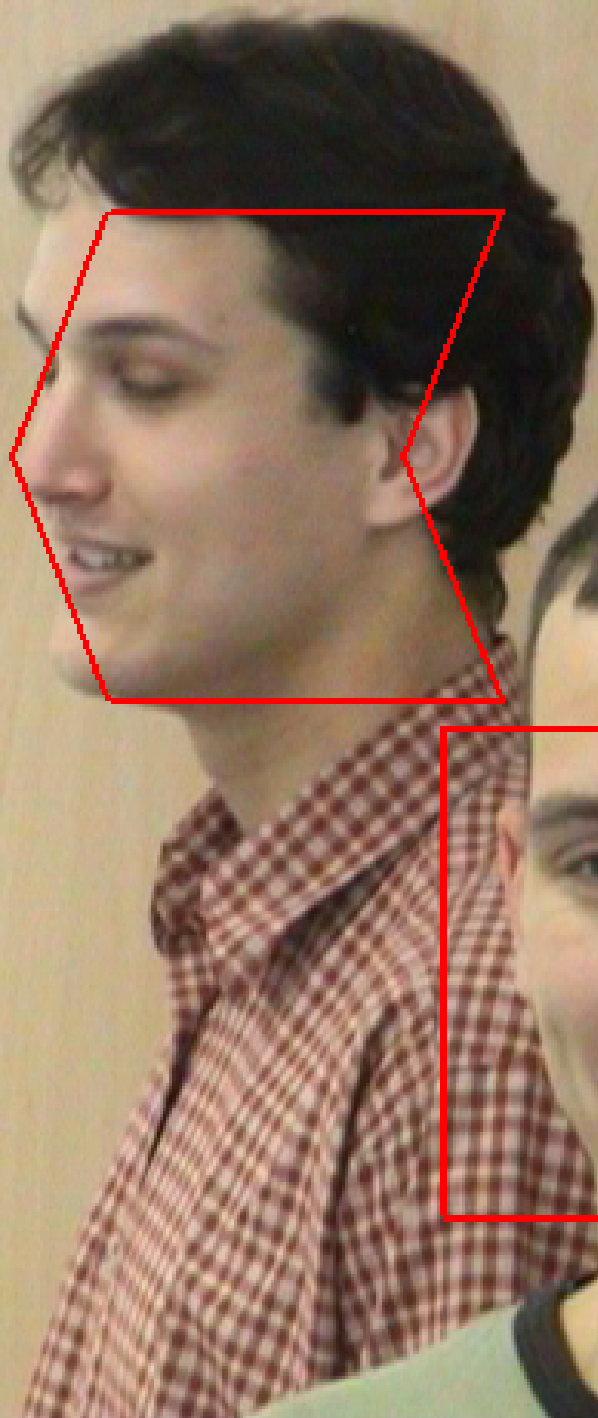
id: 164

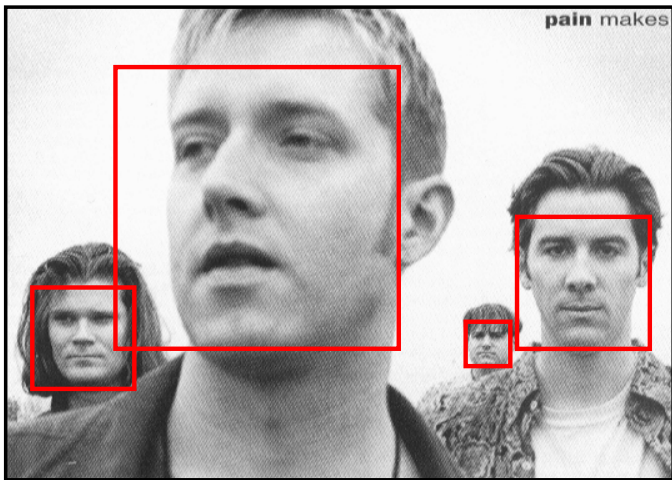
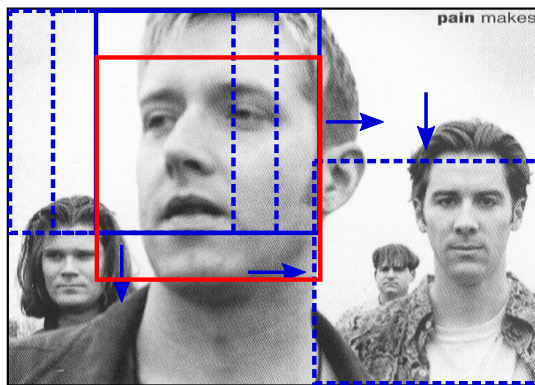
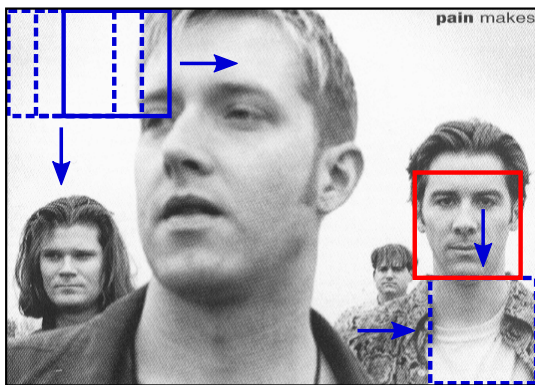
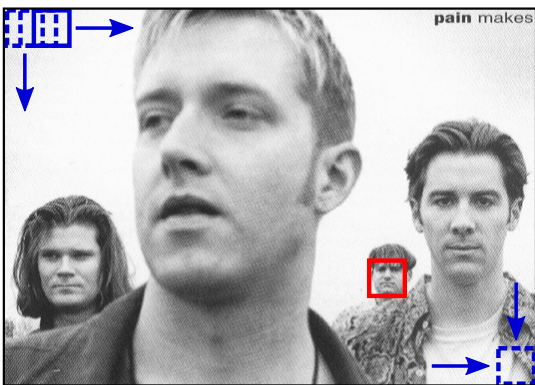


id: 165

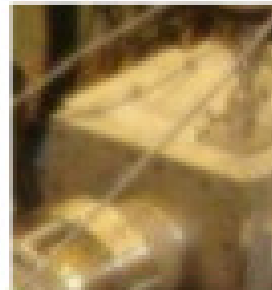
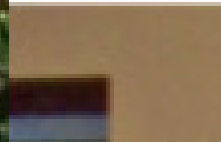
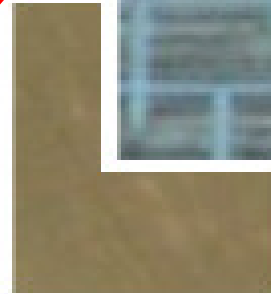
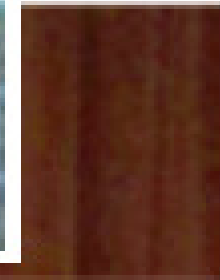
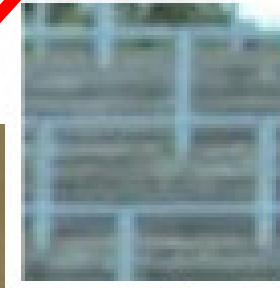
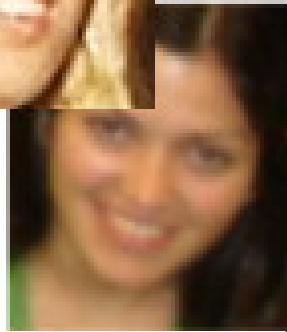
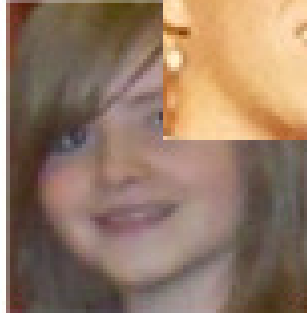




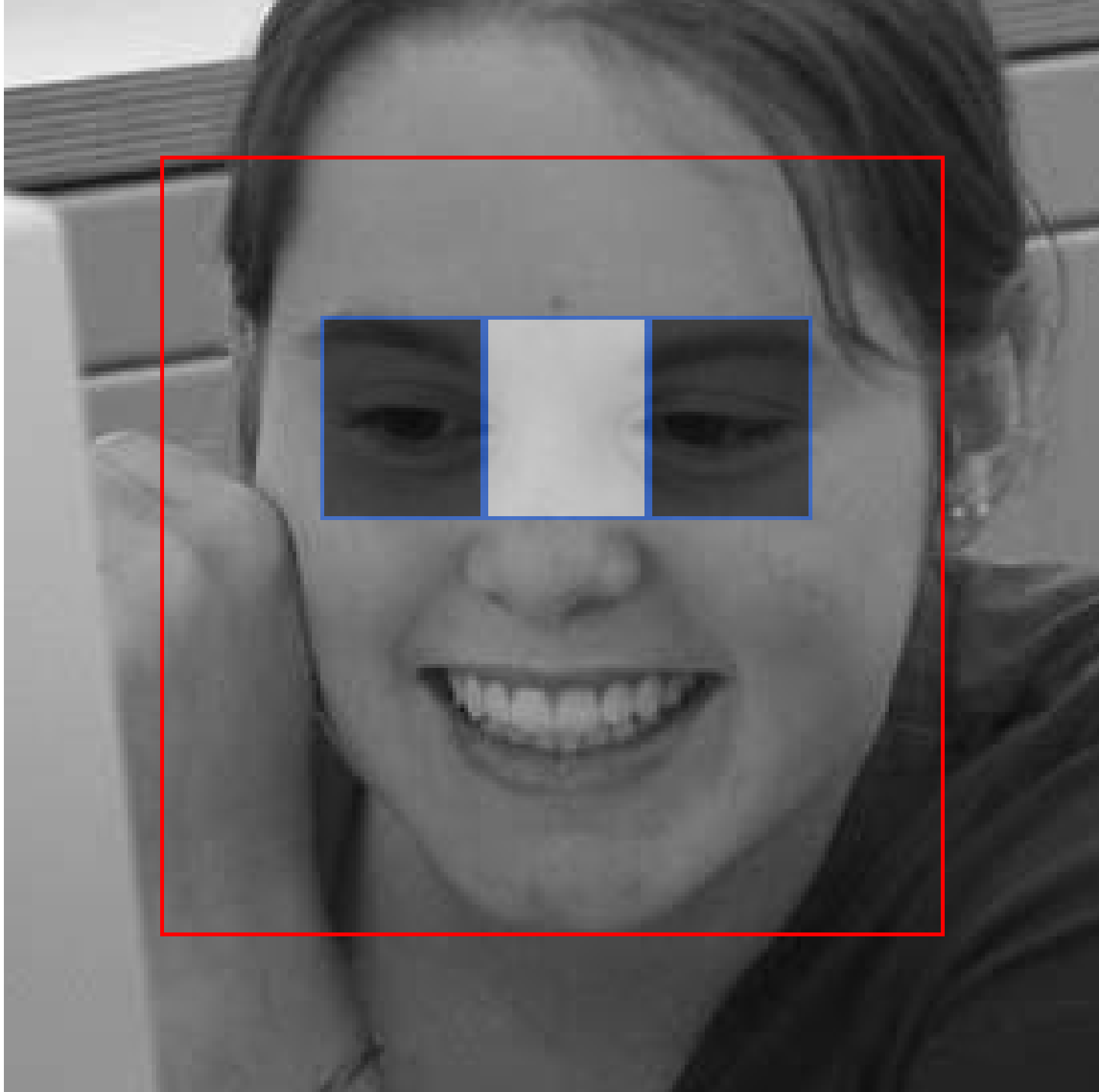


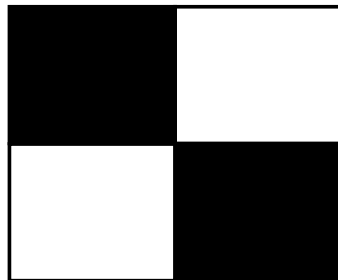
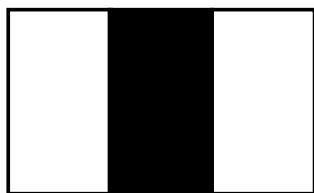
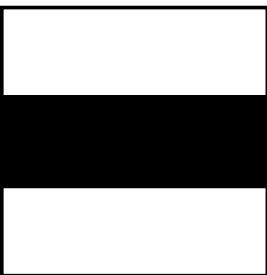
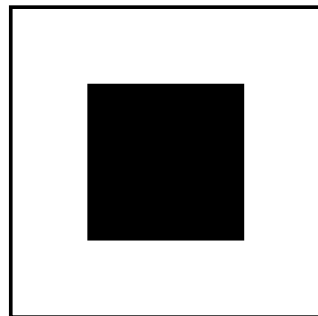


FACE



NONFACE





..... -1

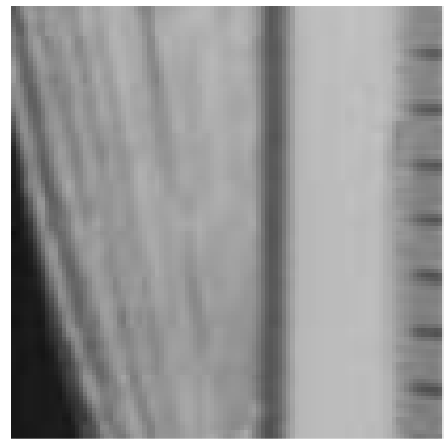
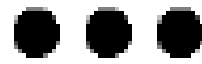
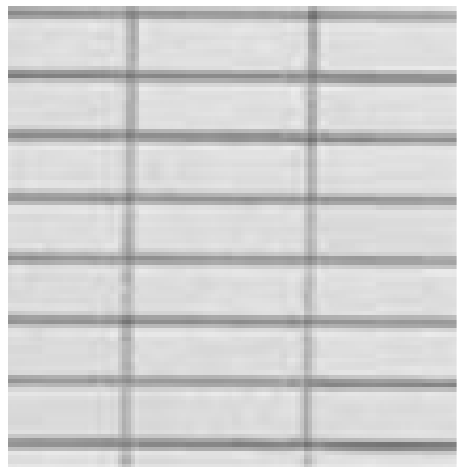
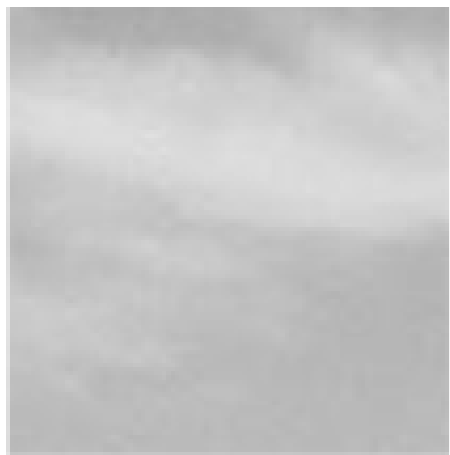
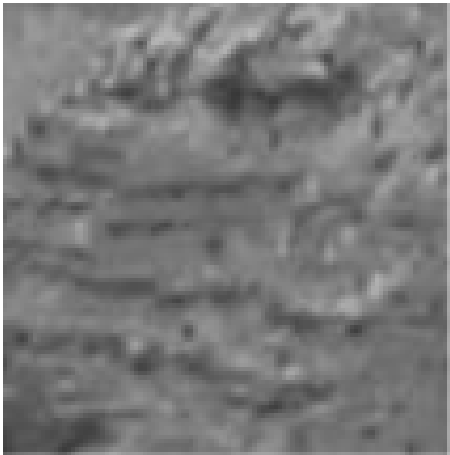


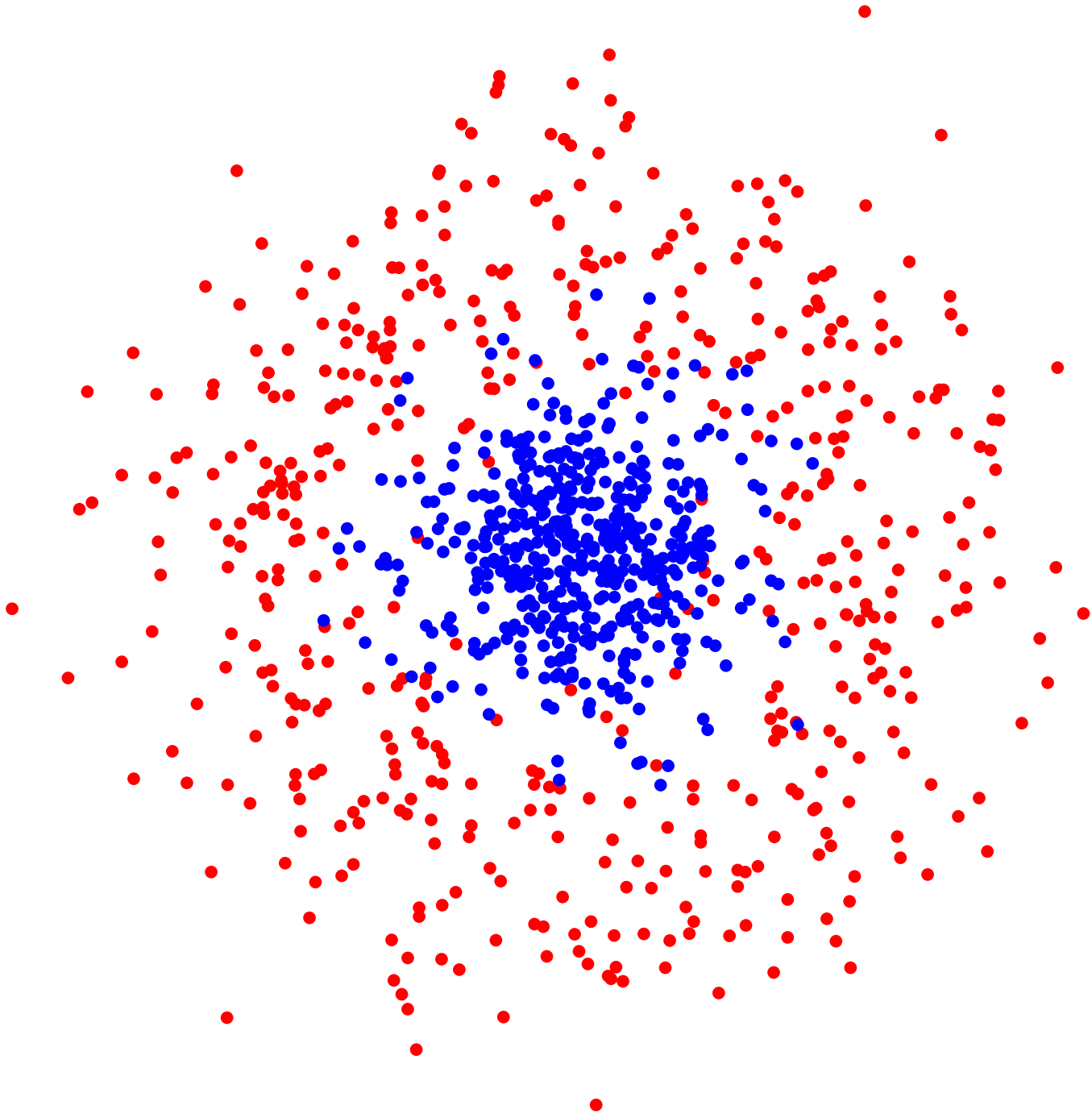
..... +1

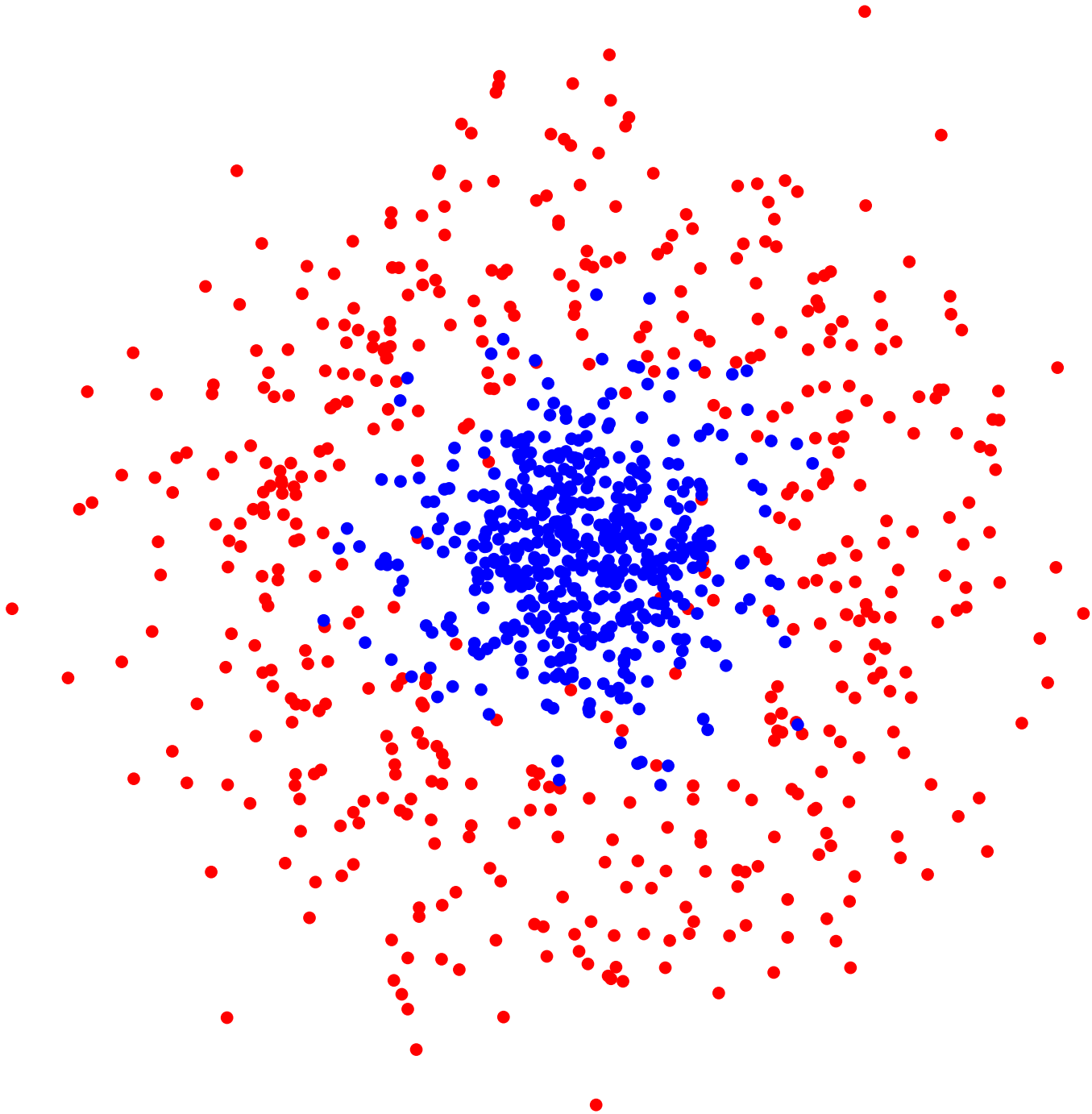


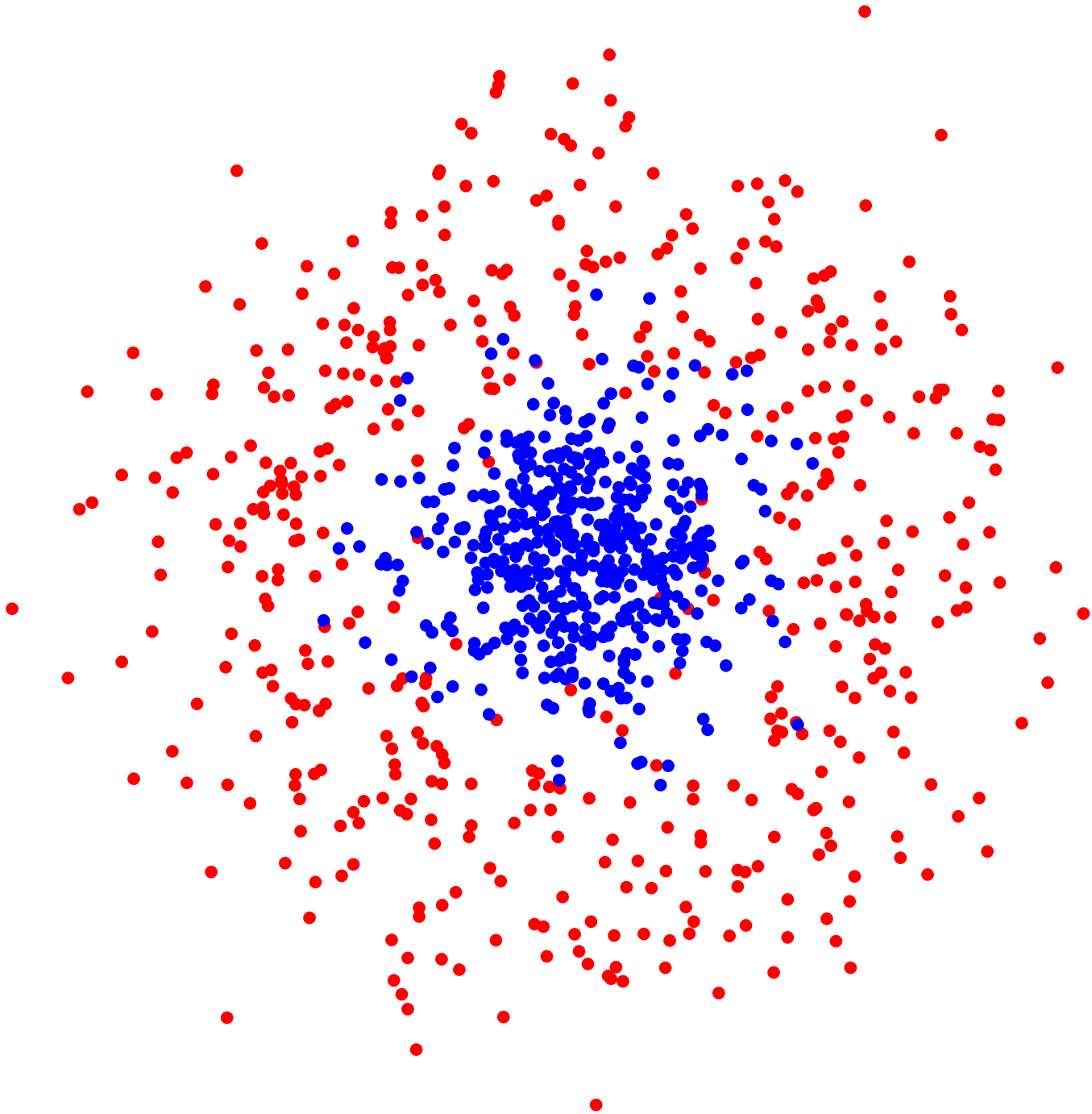
...

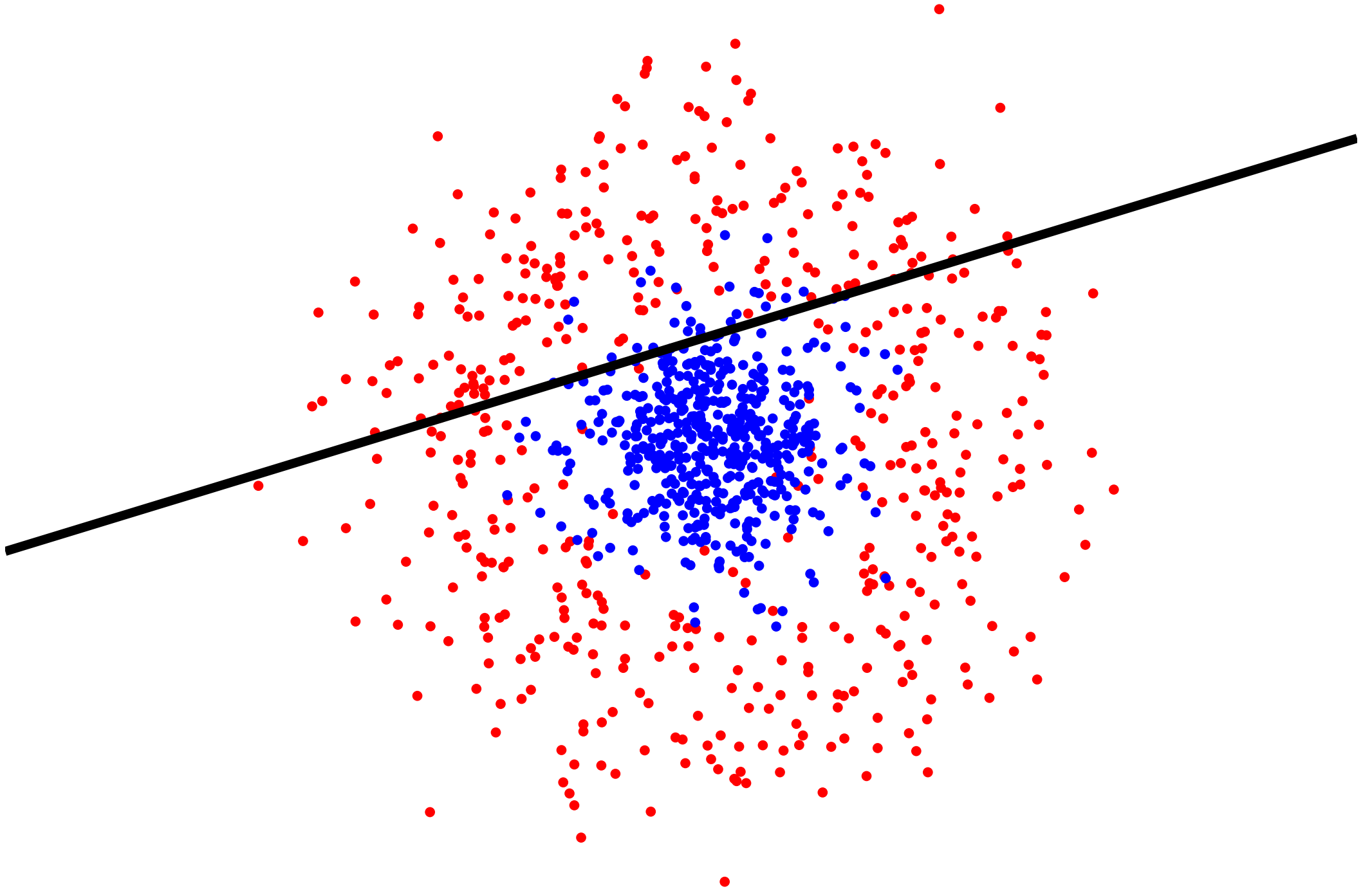


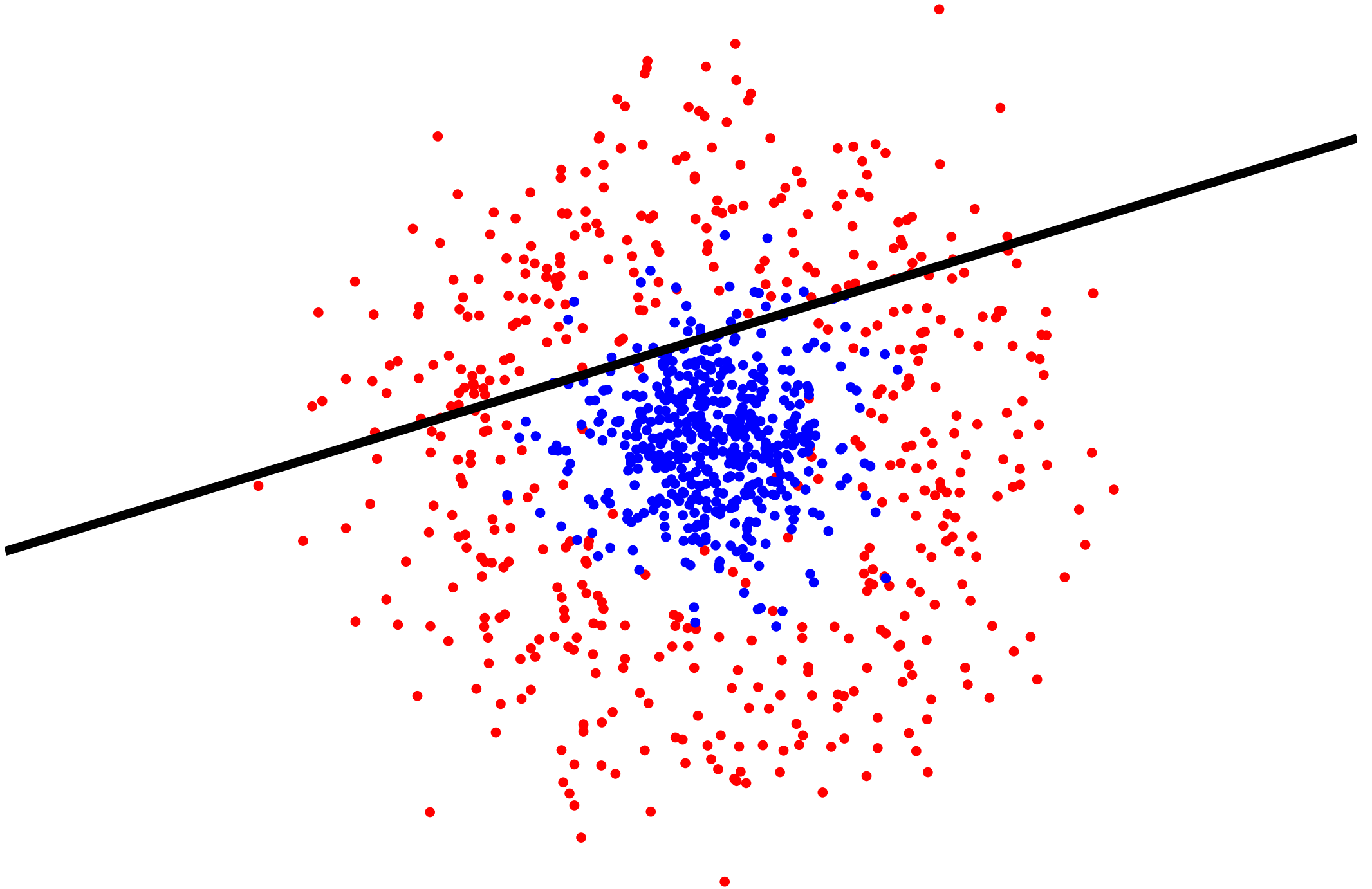


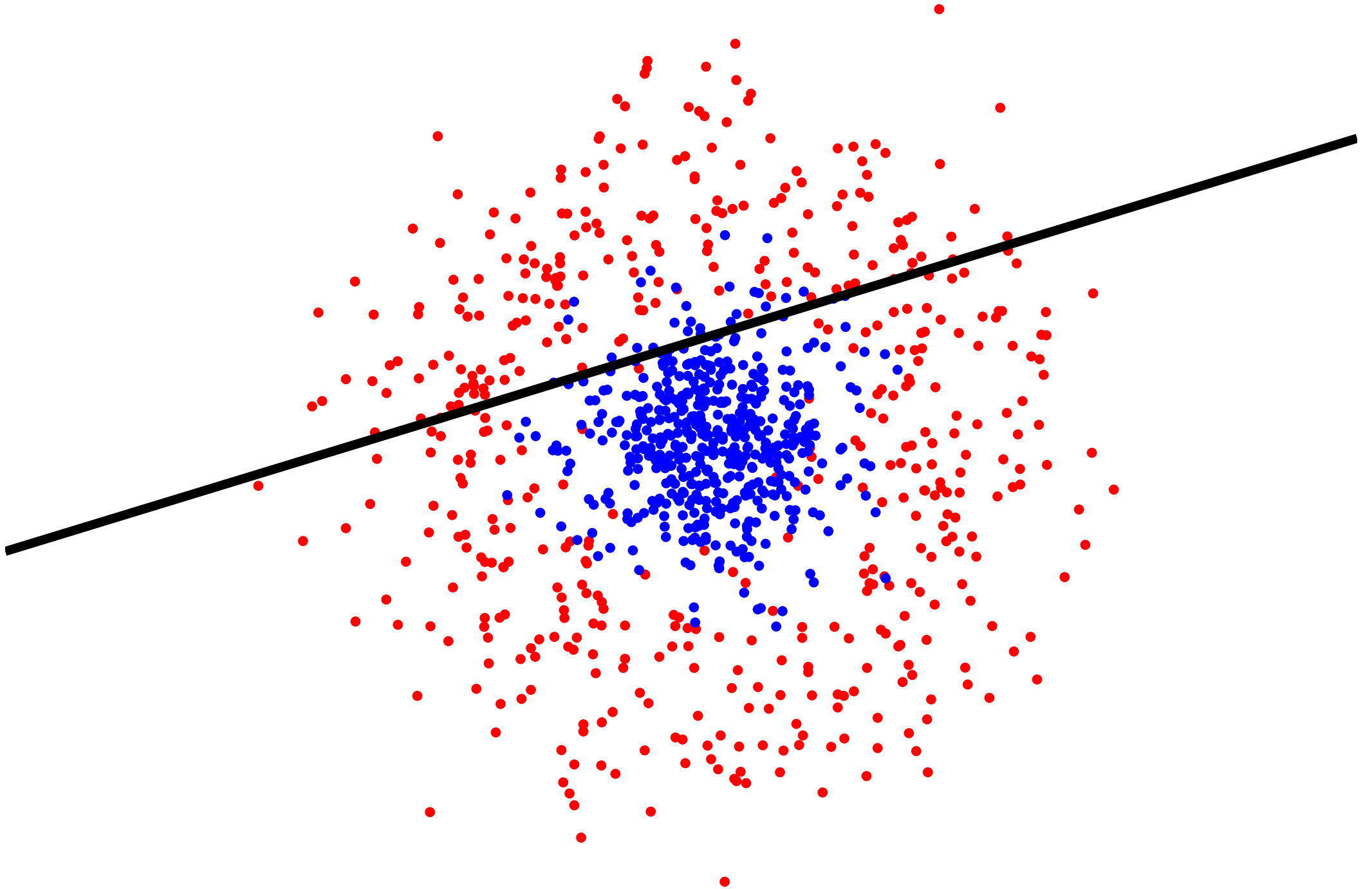


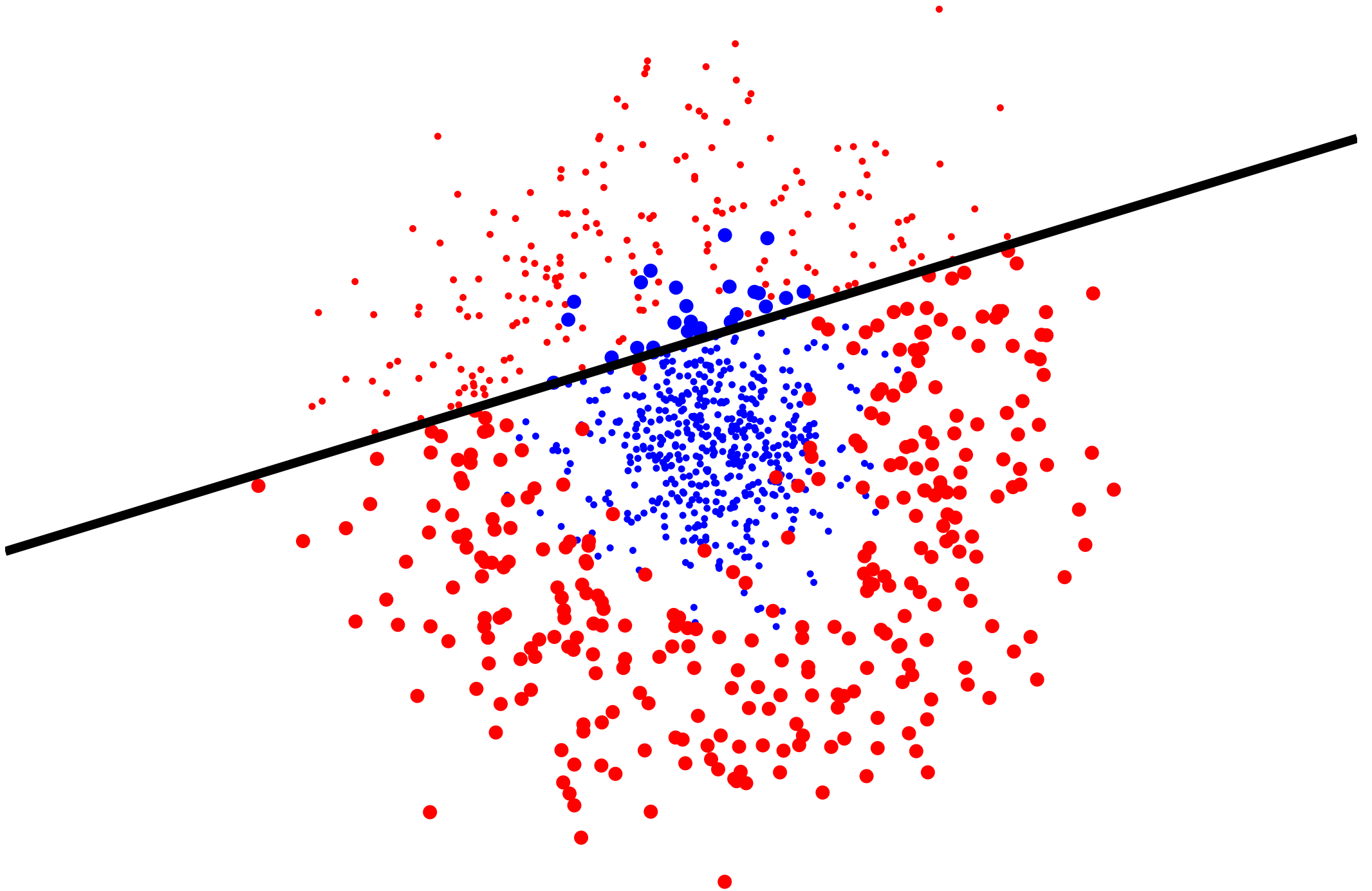


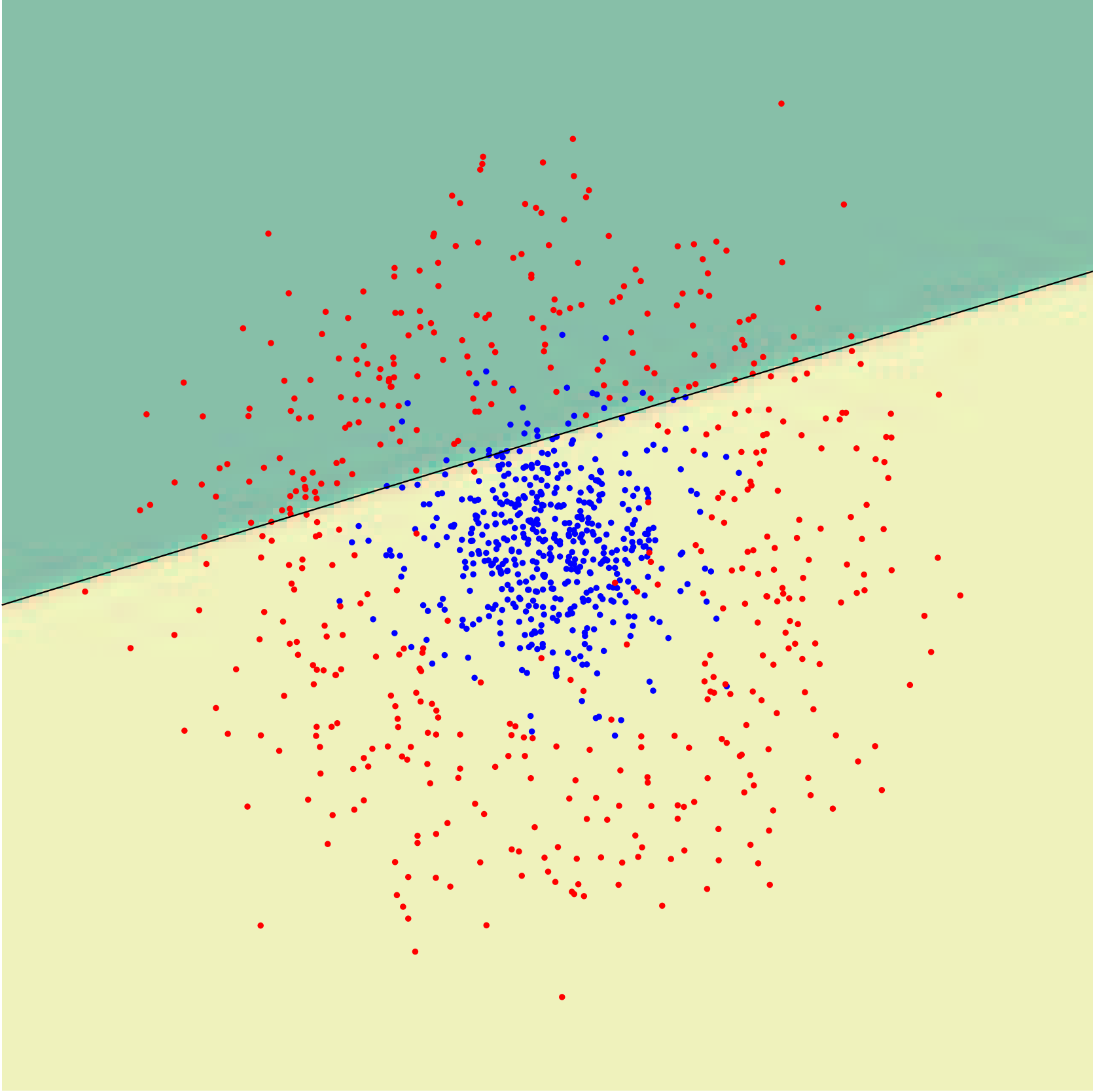


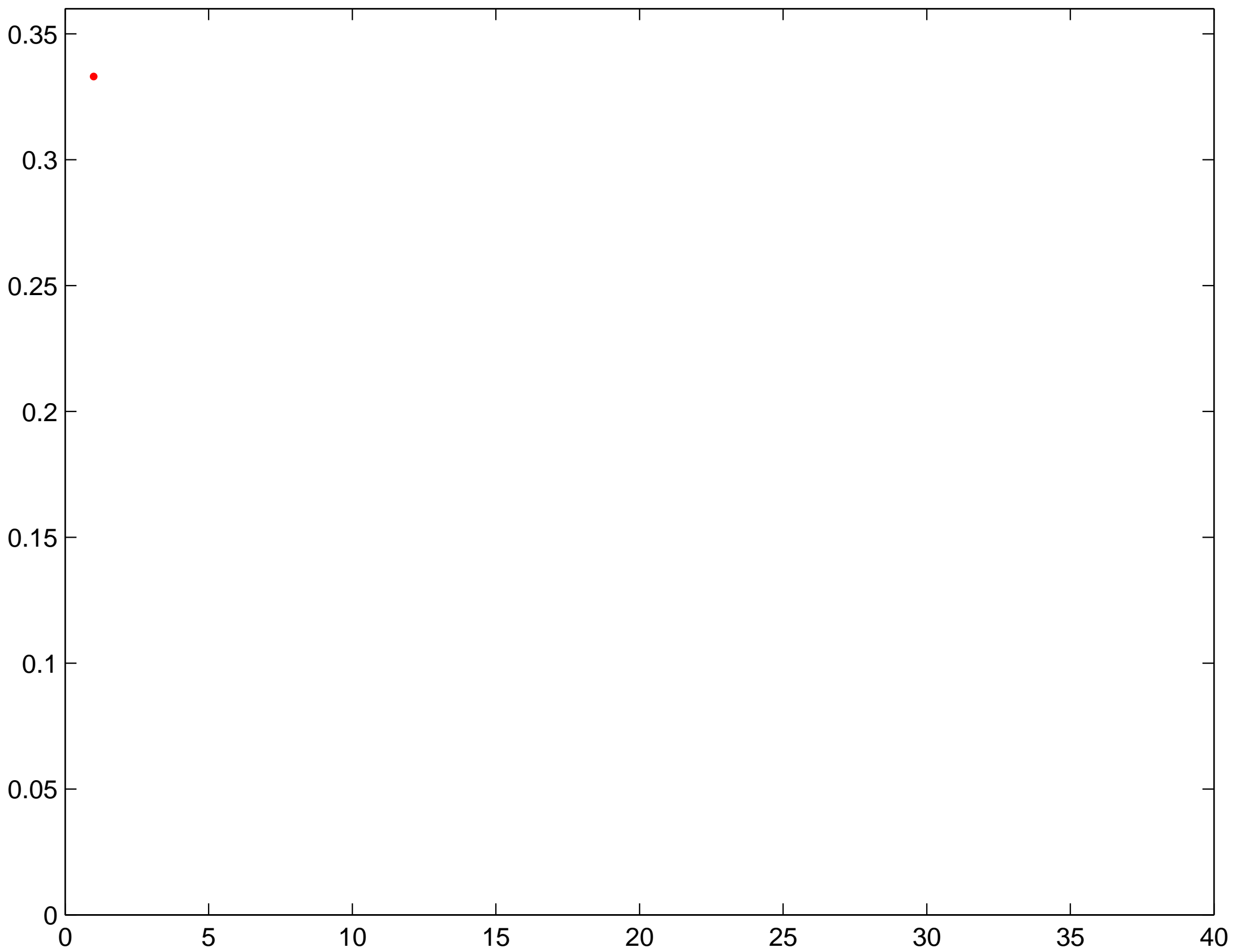


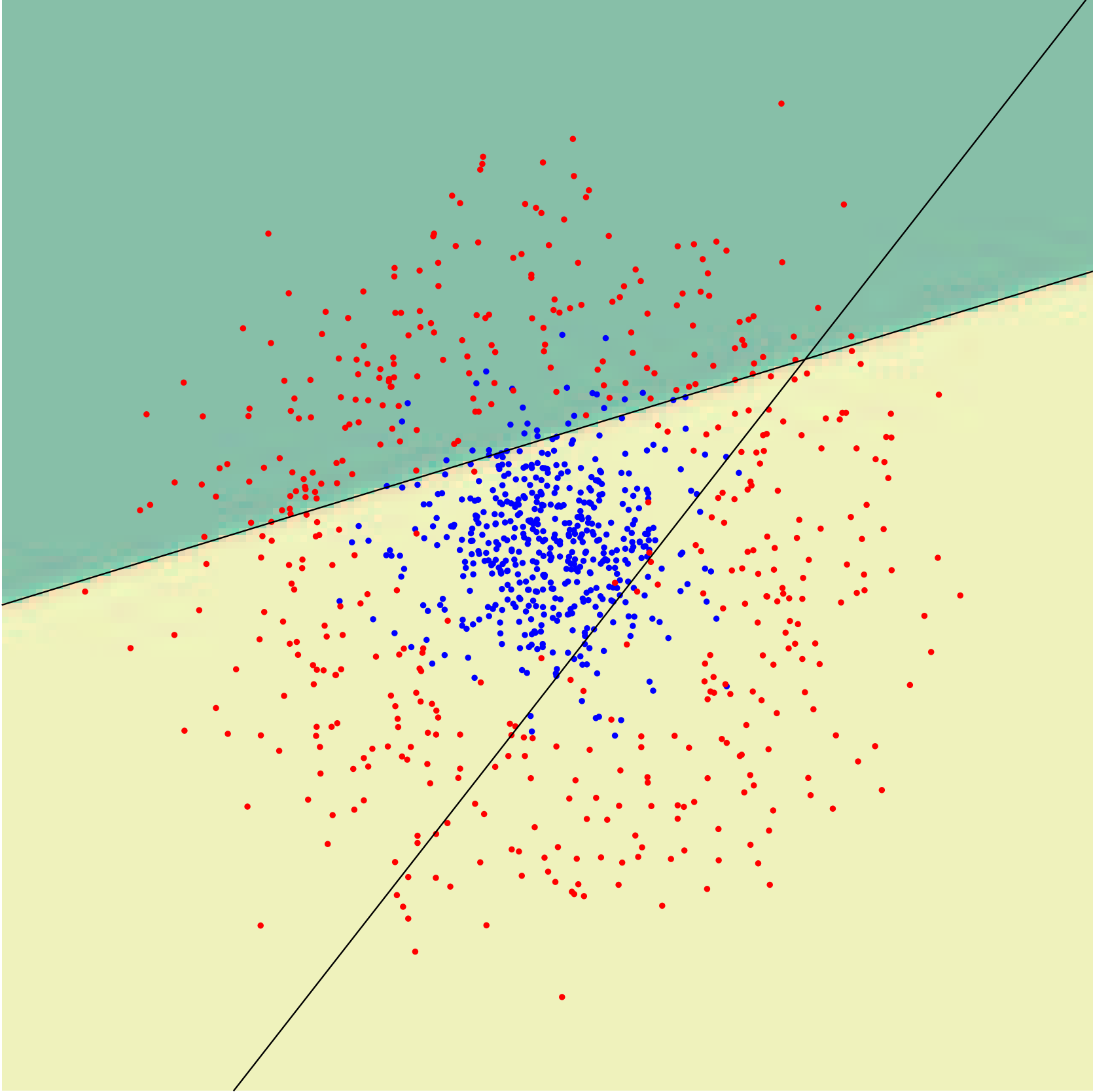


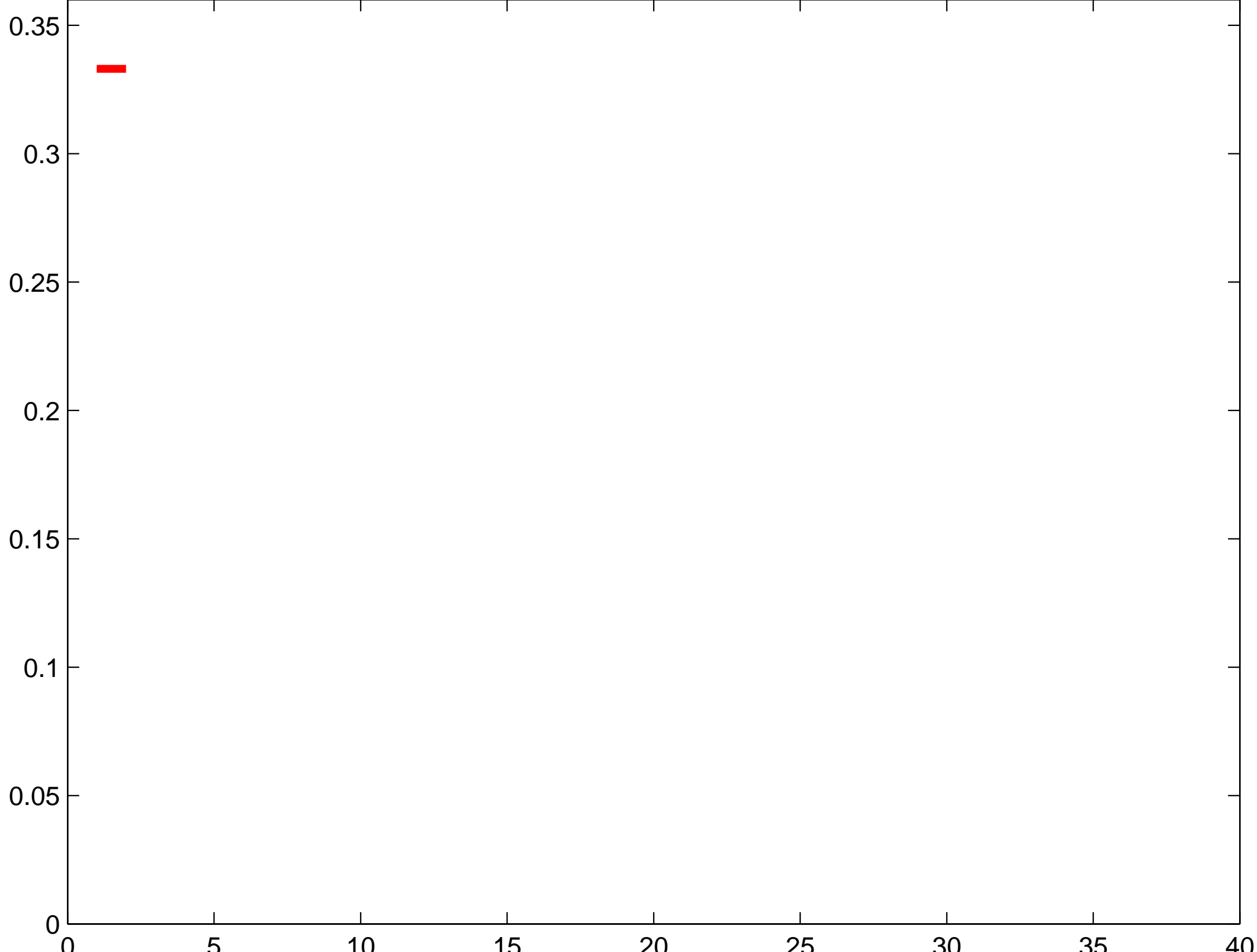


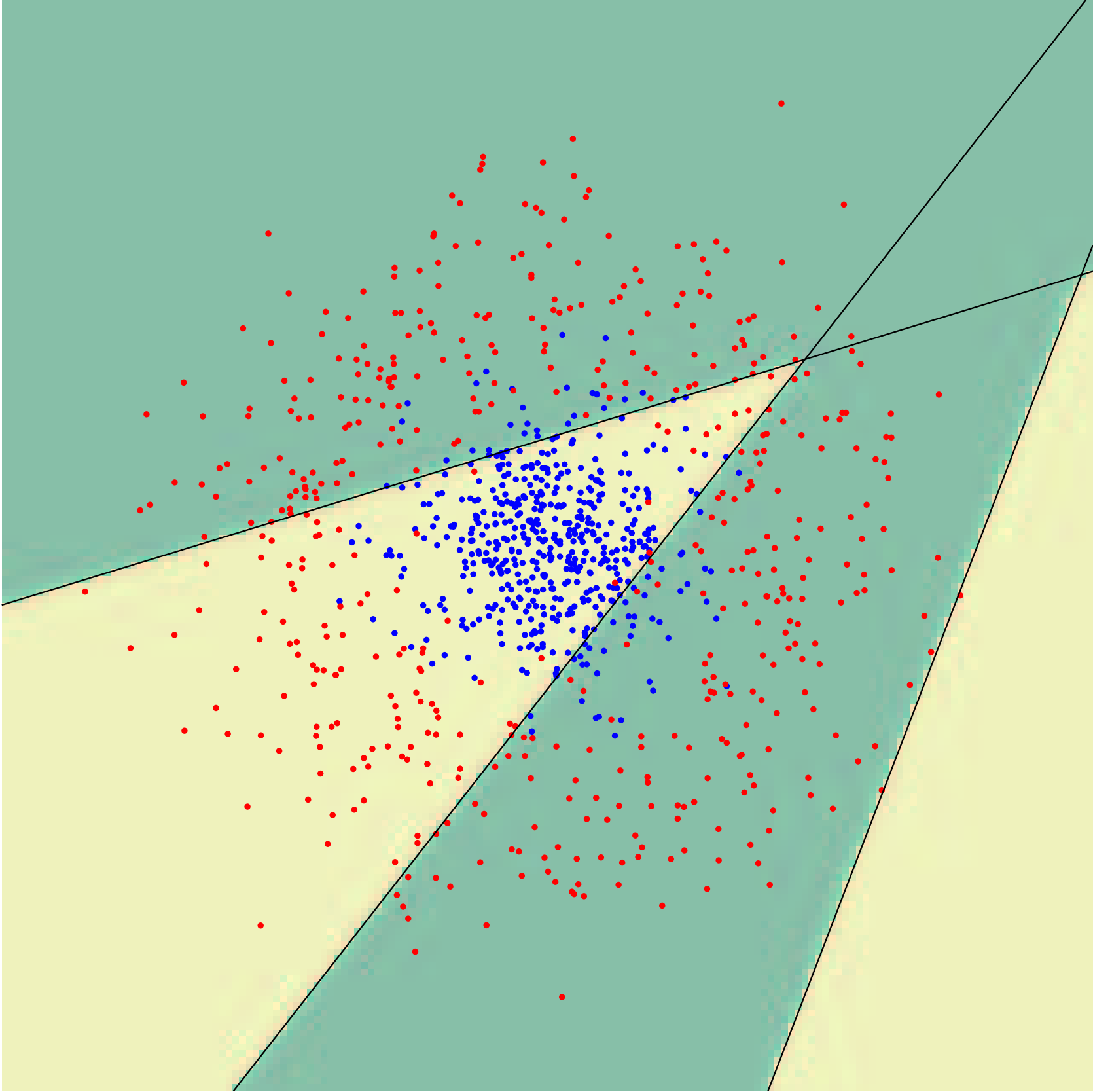


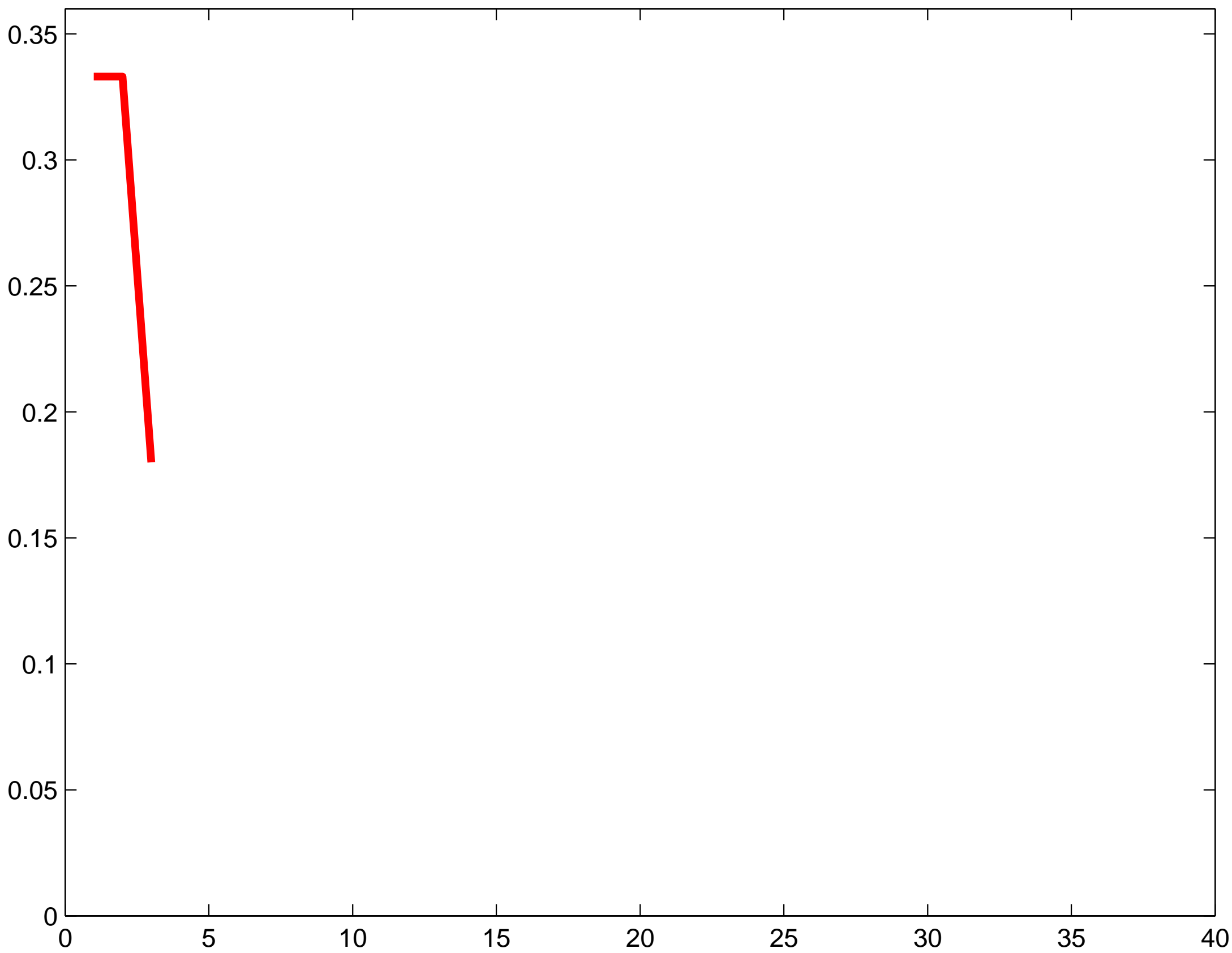


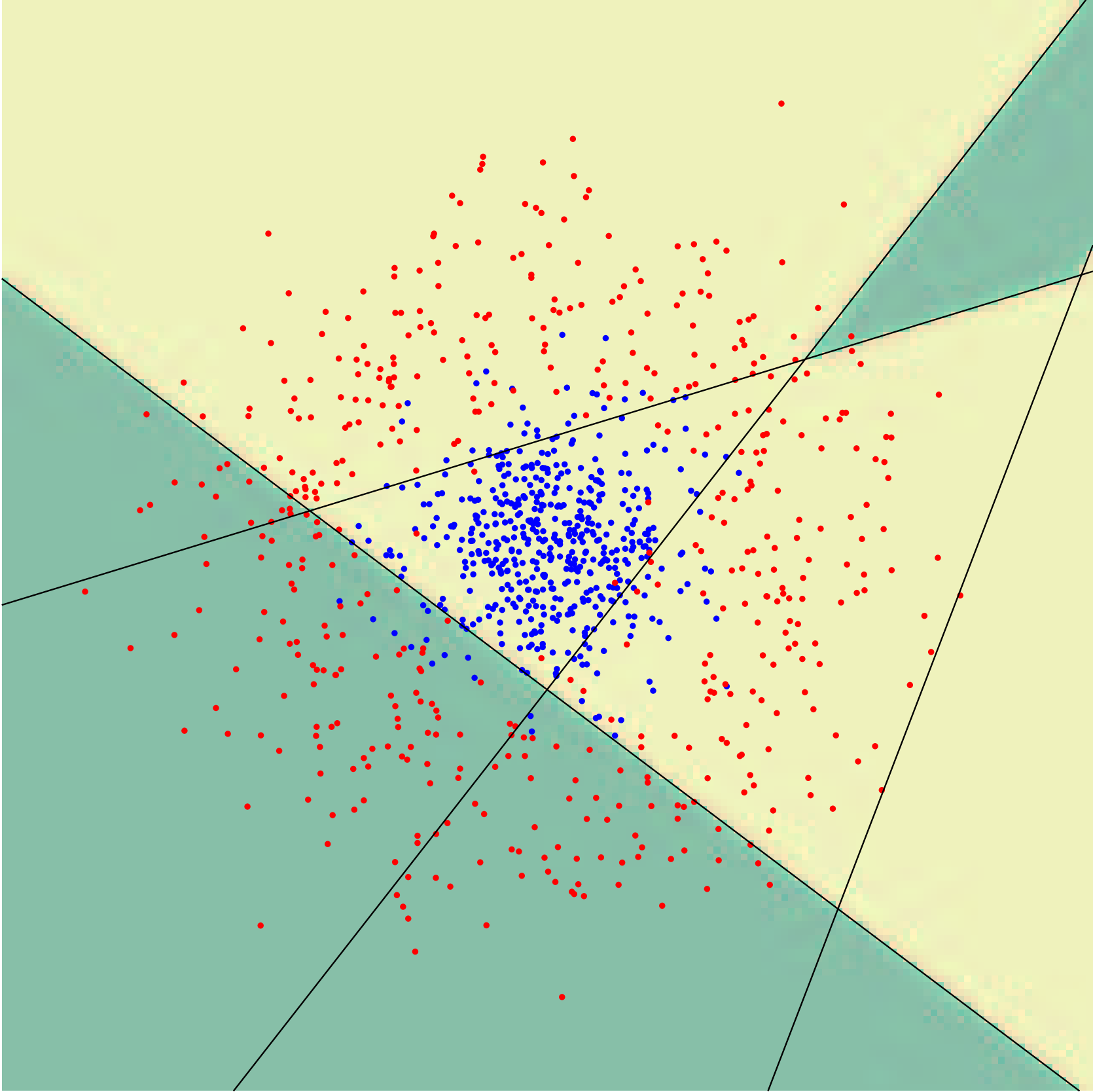


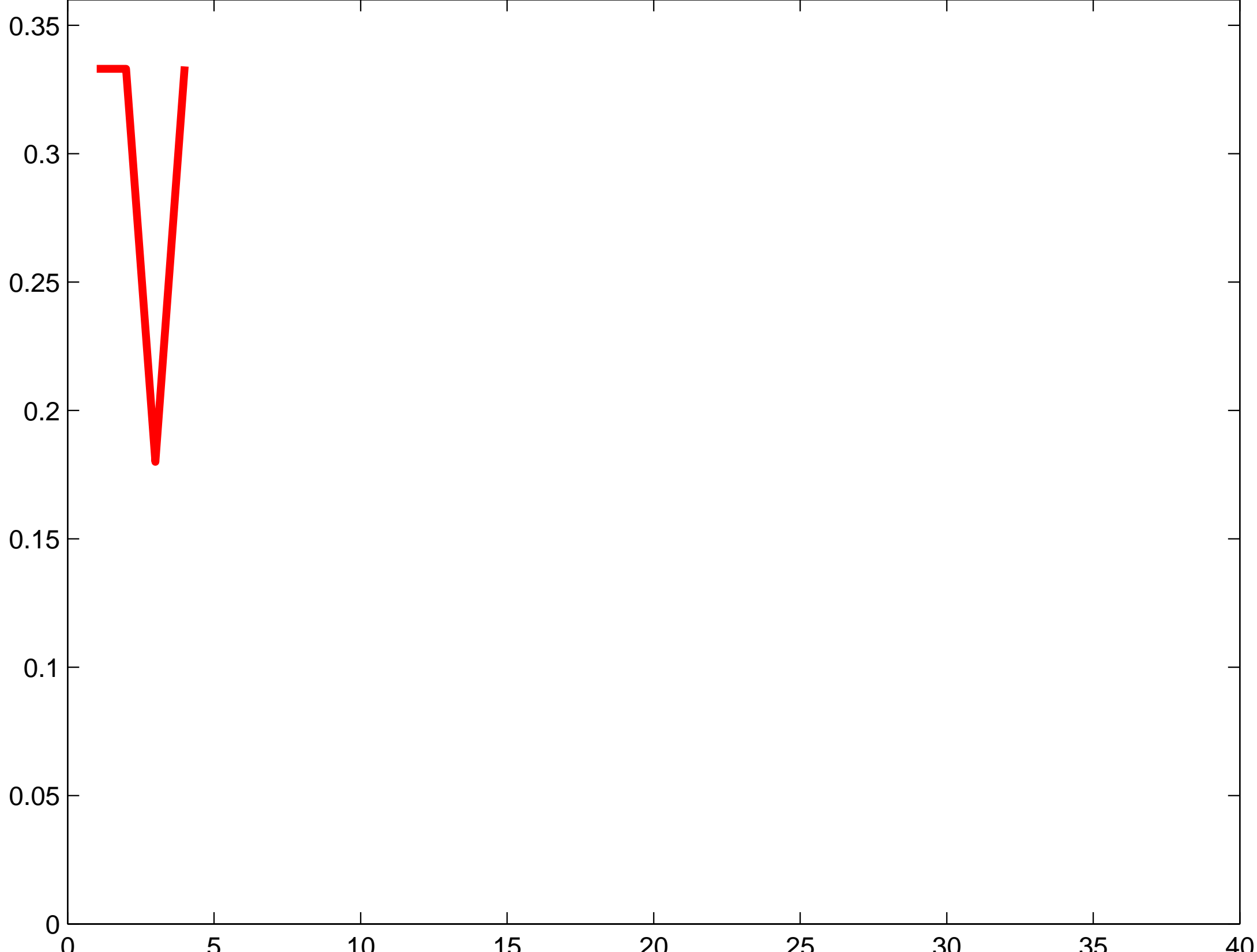


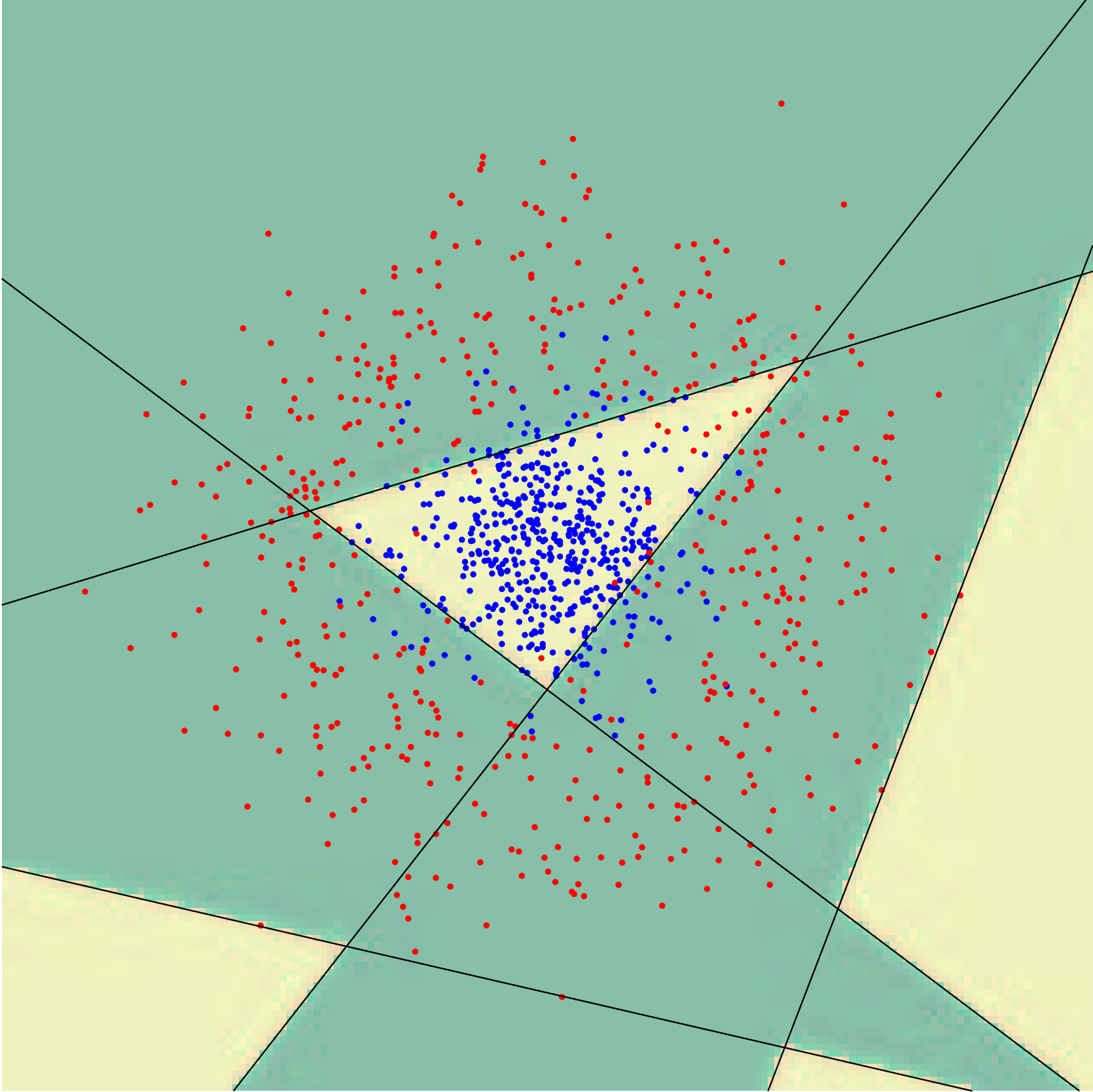


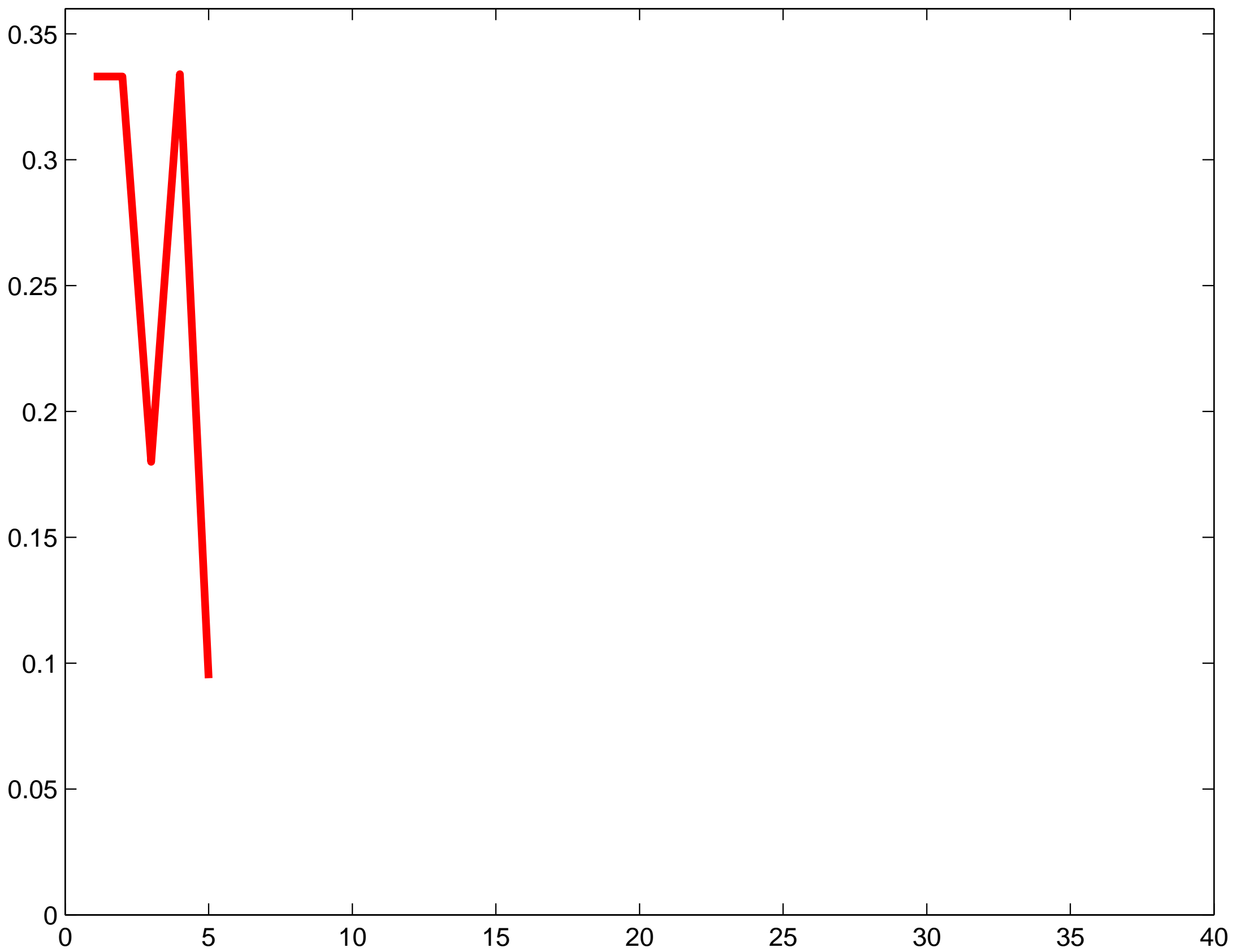


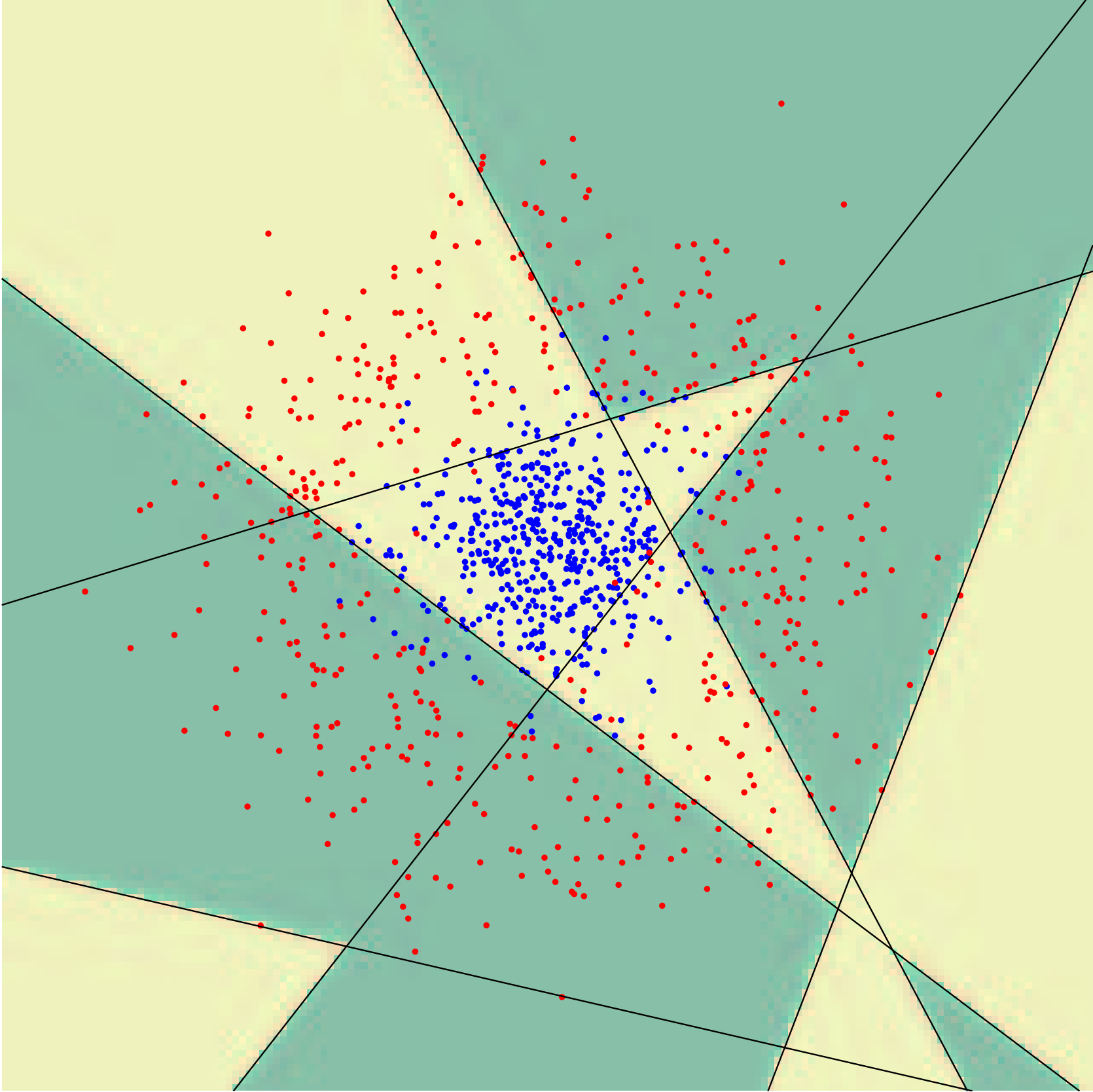


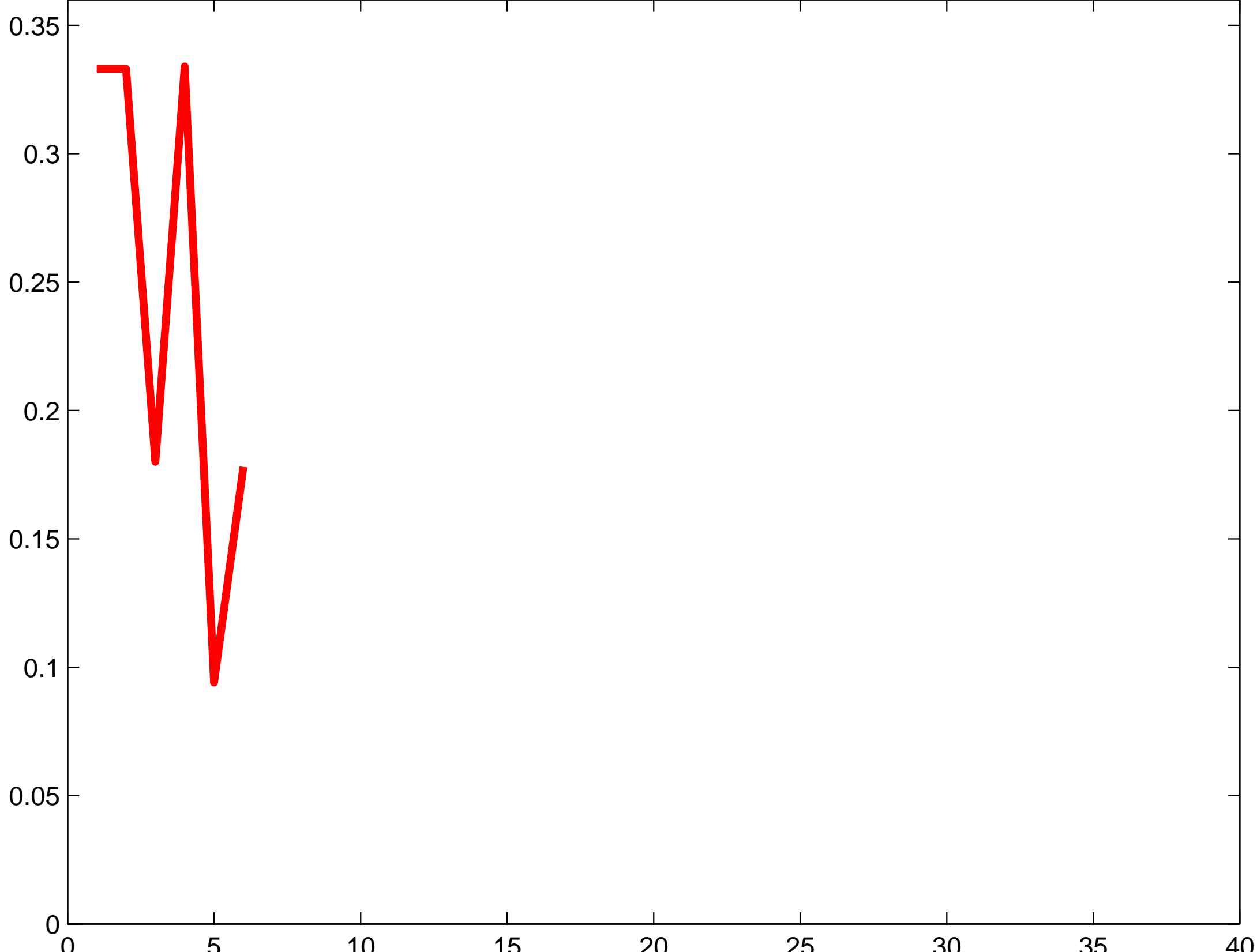


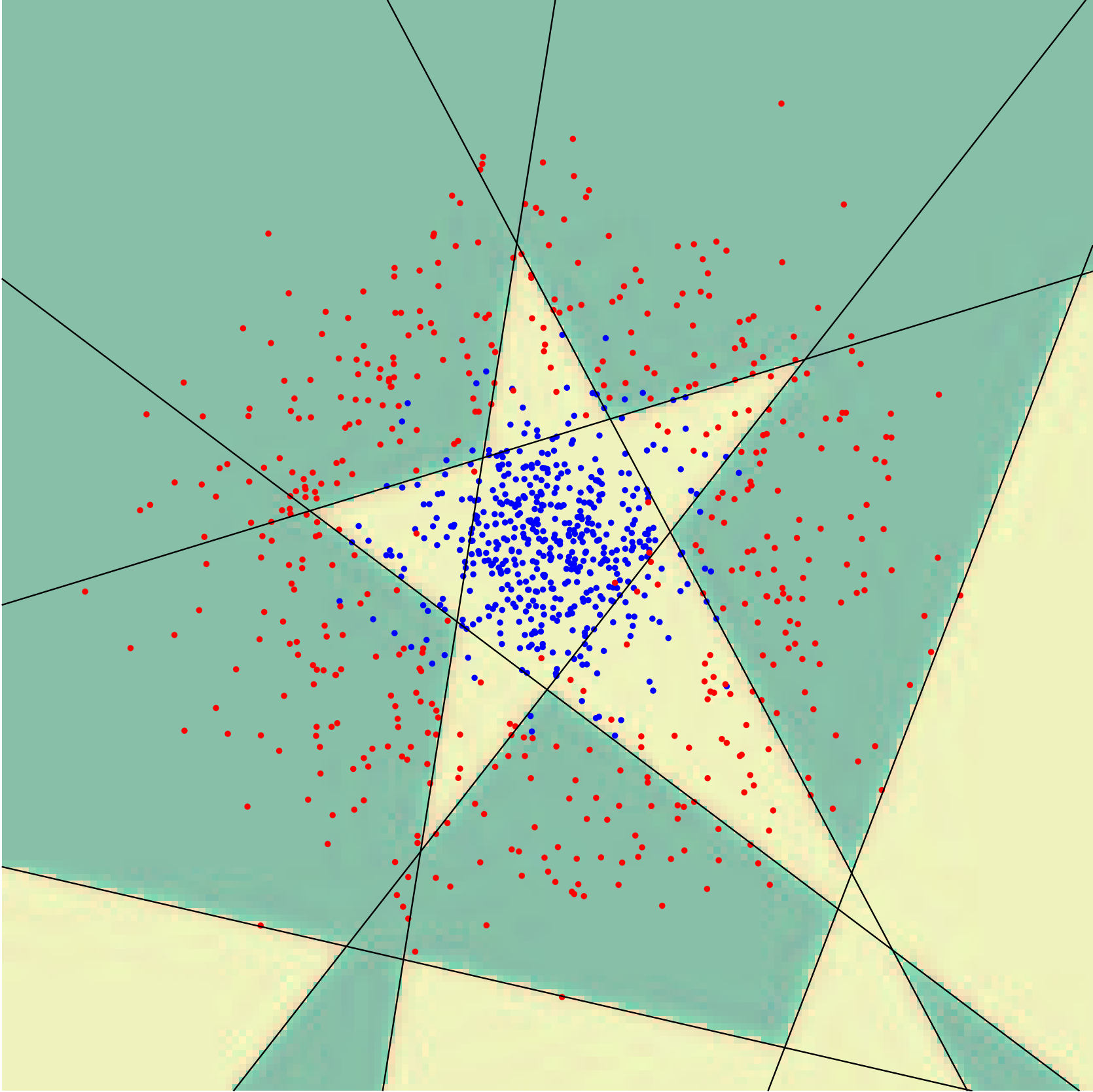


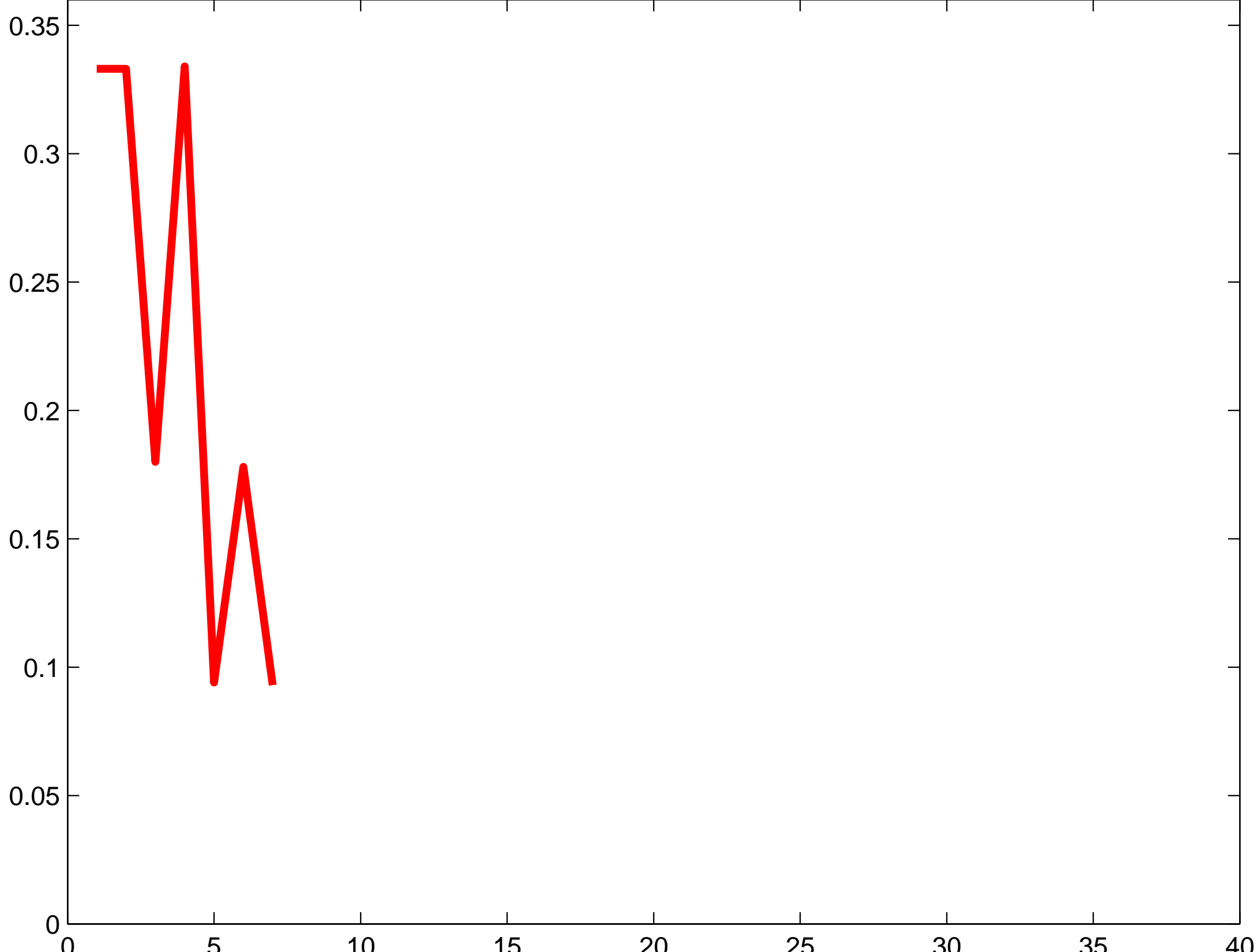


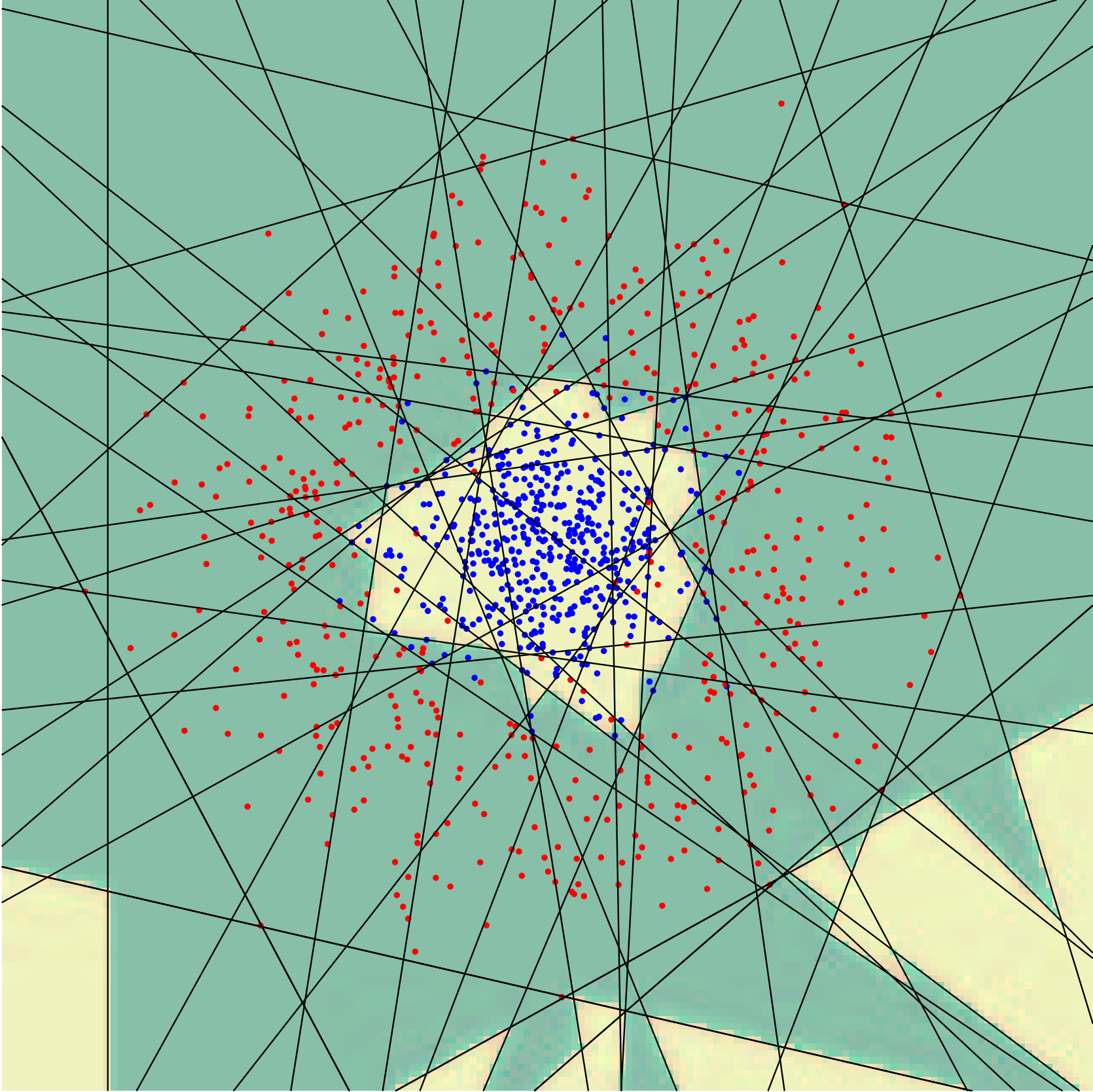


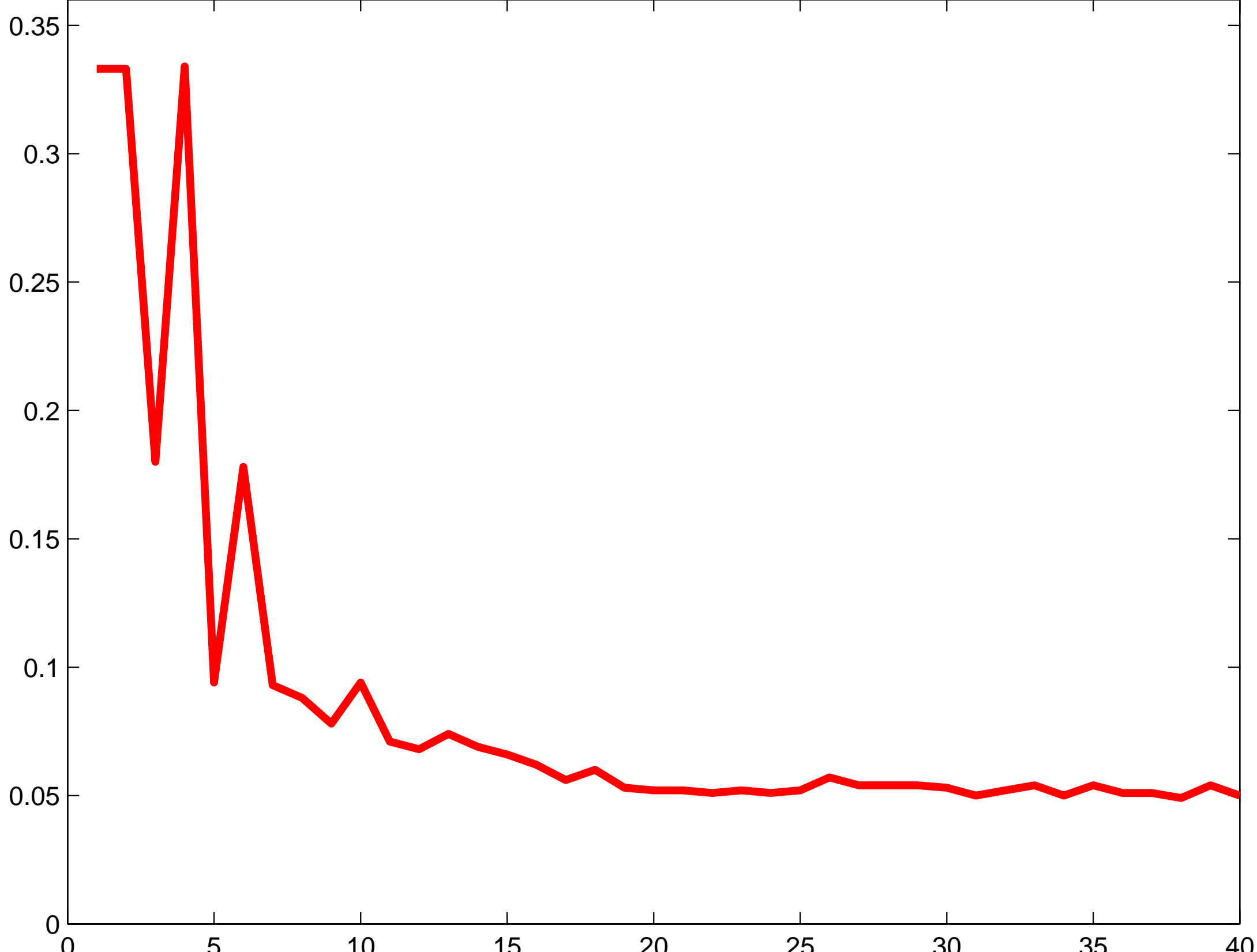


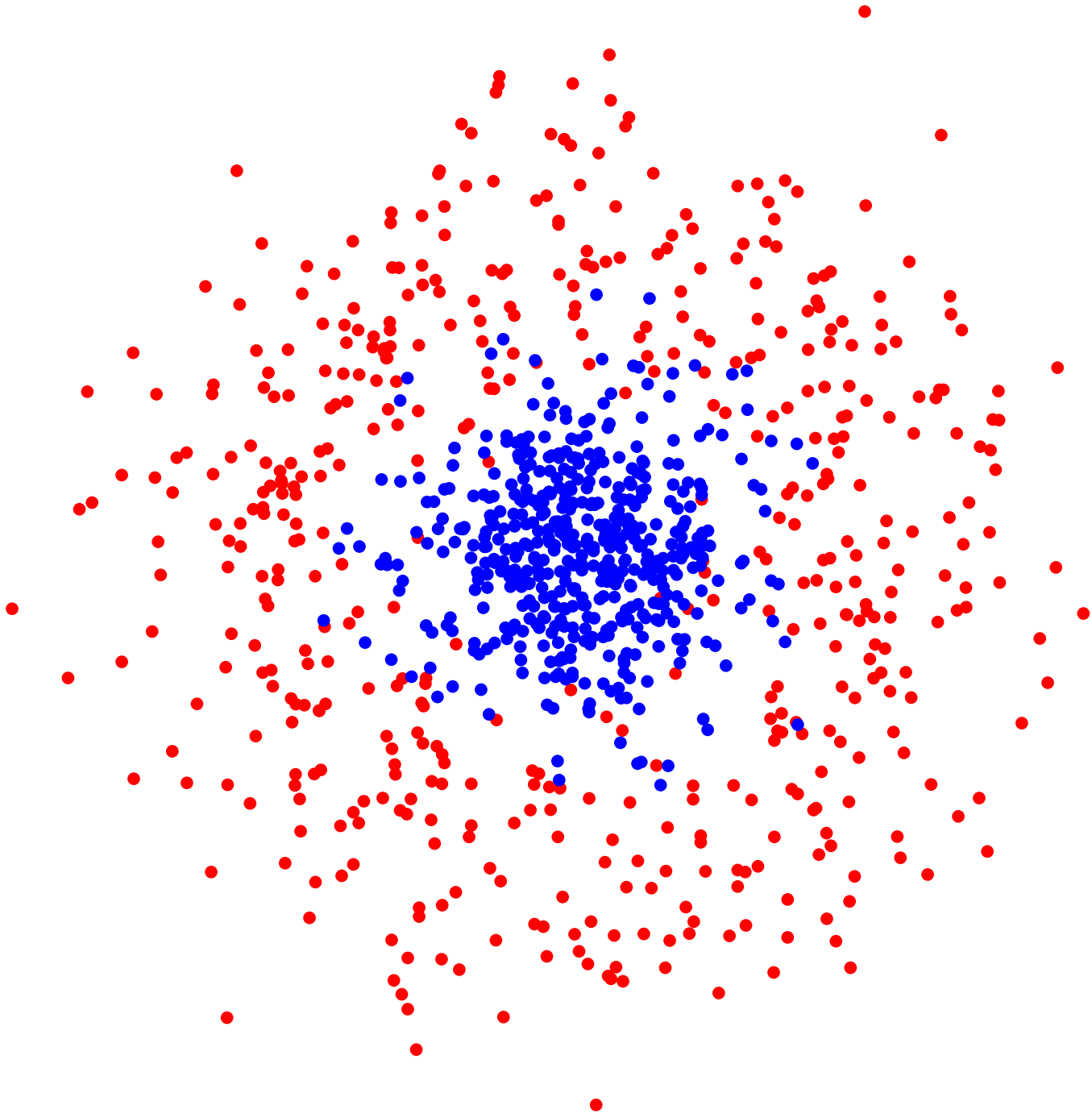


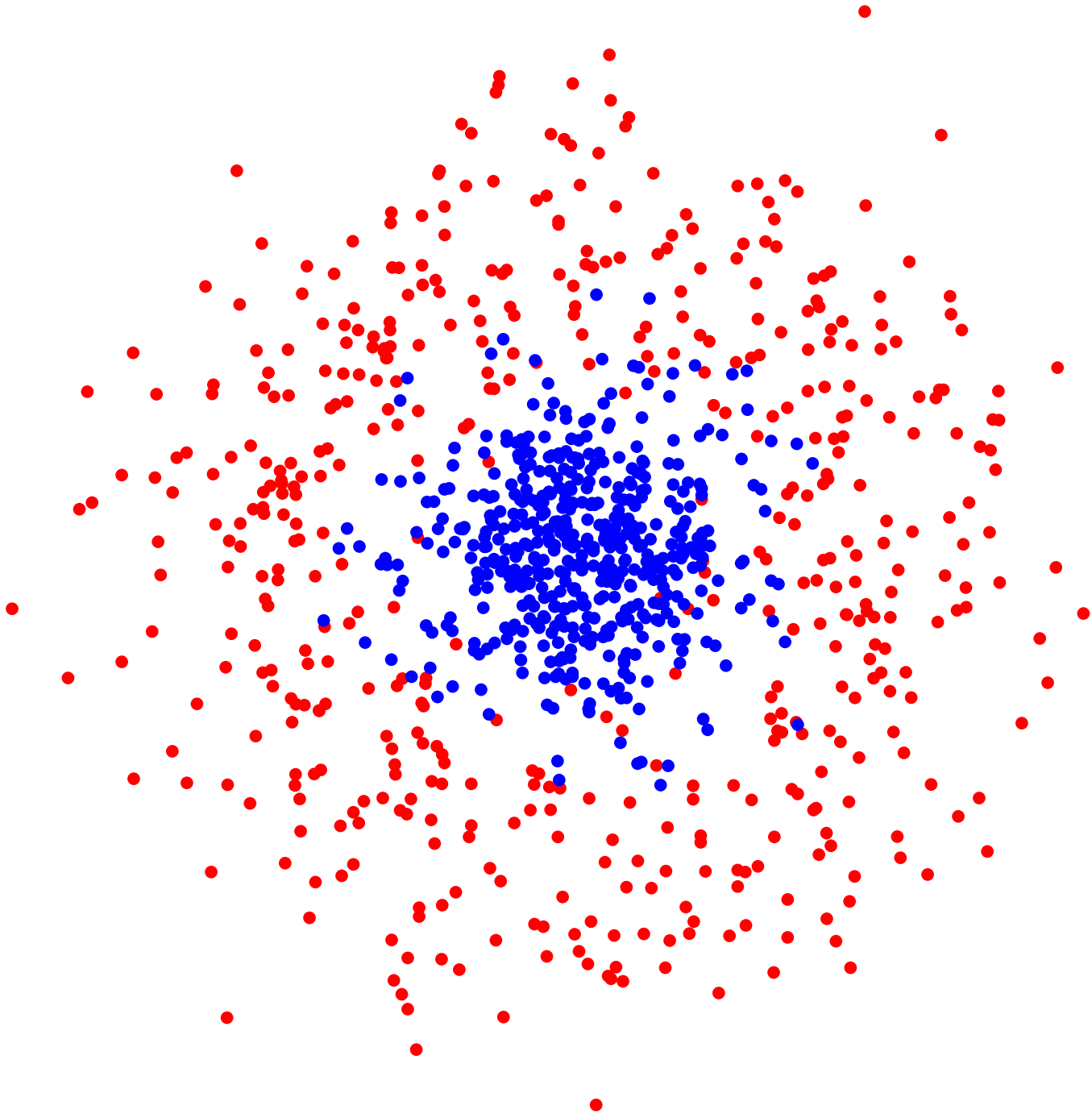


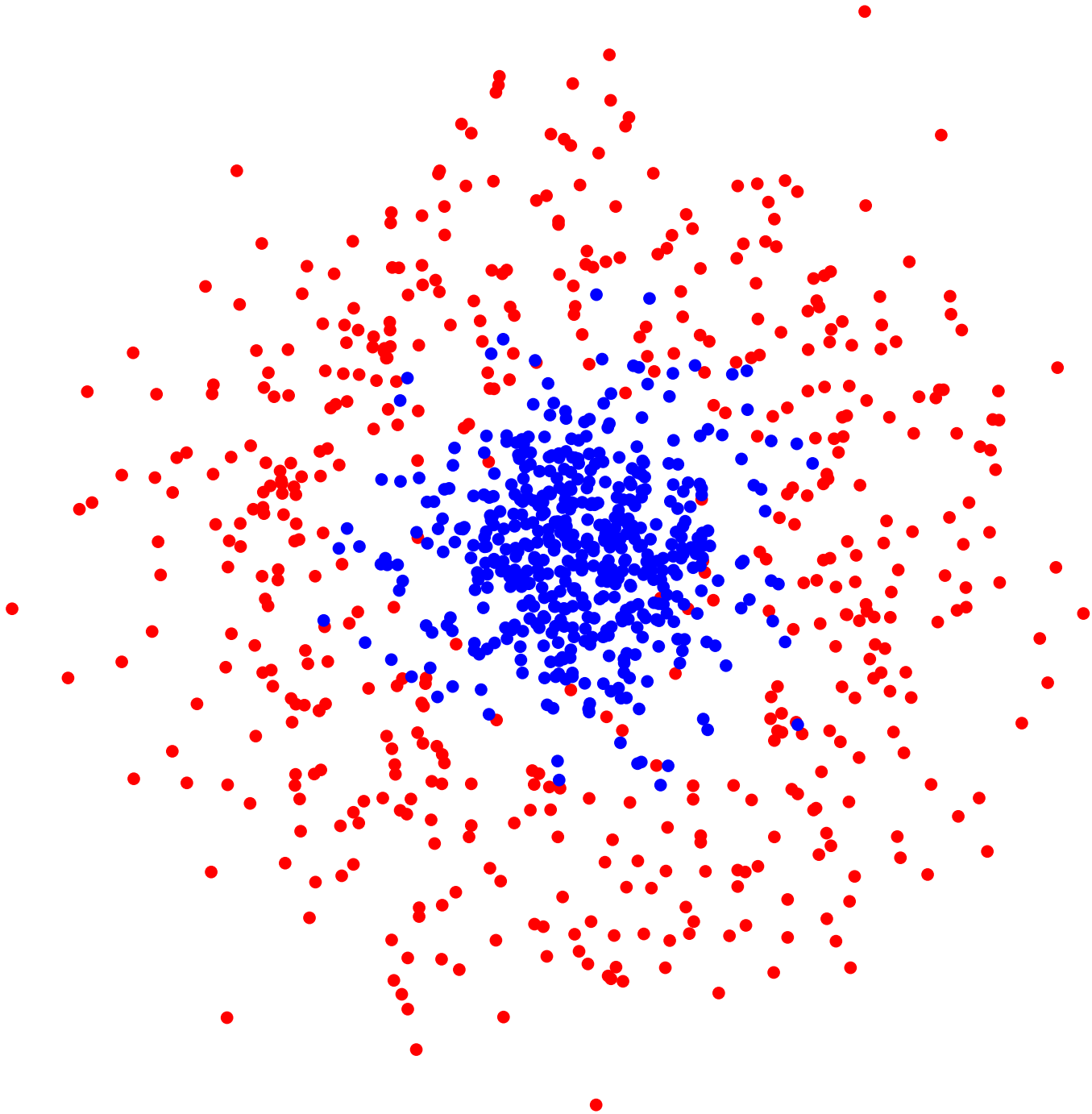


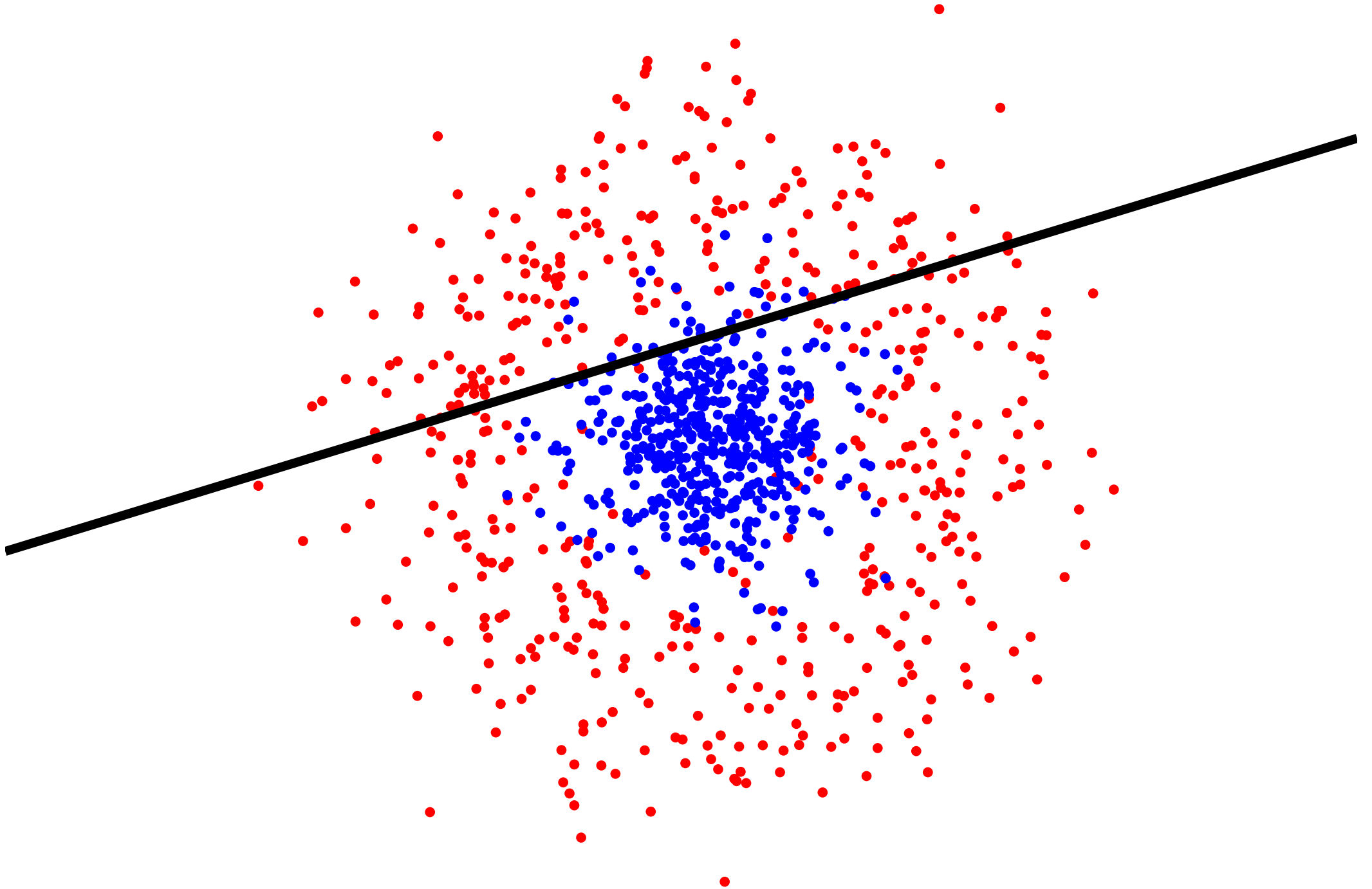


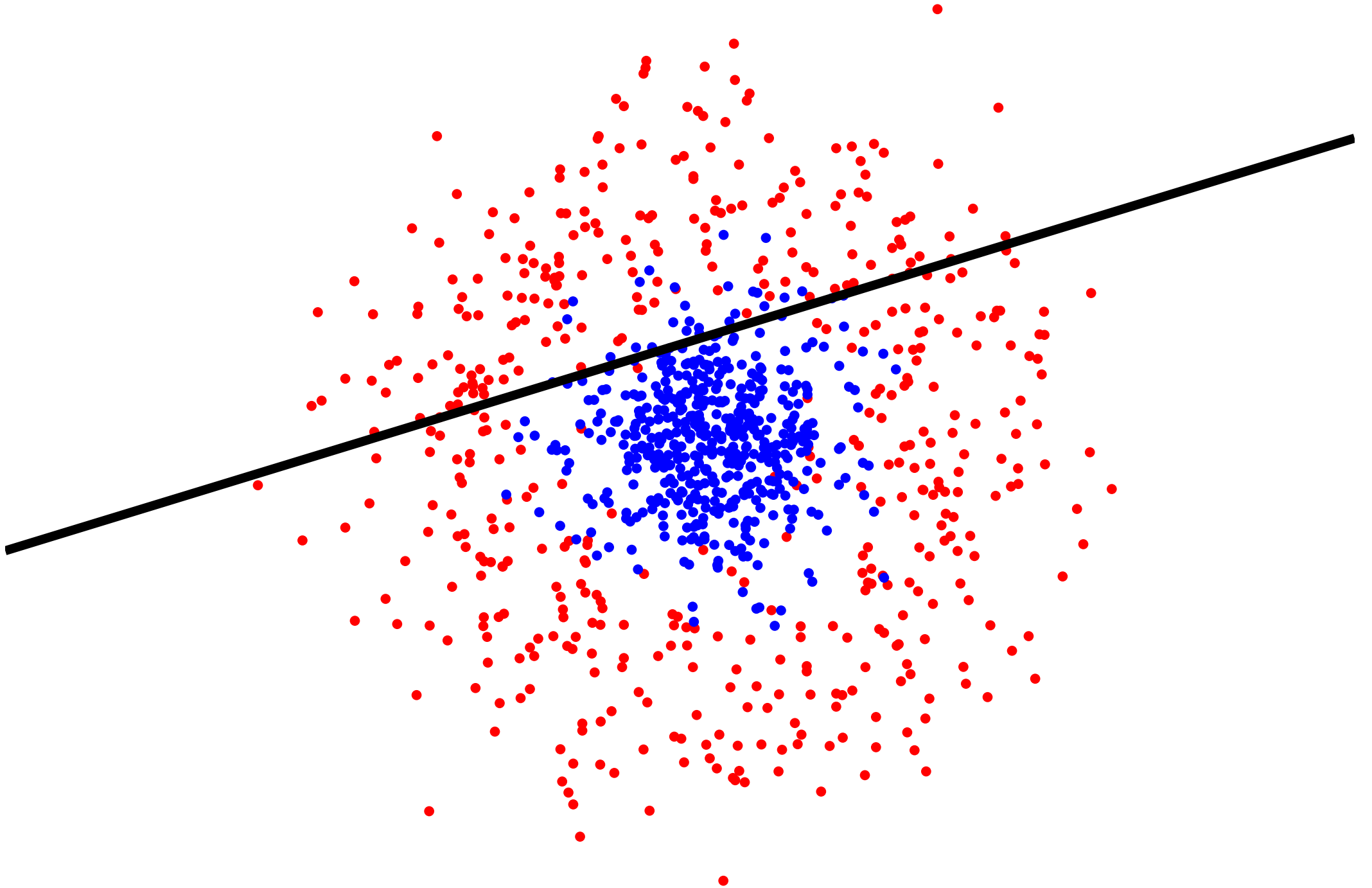


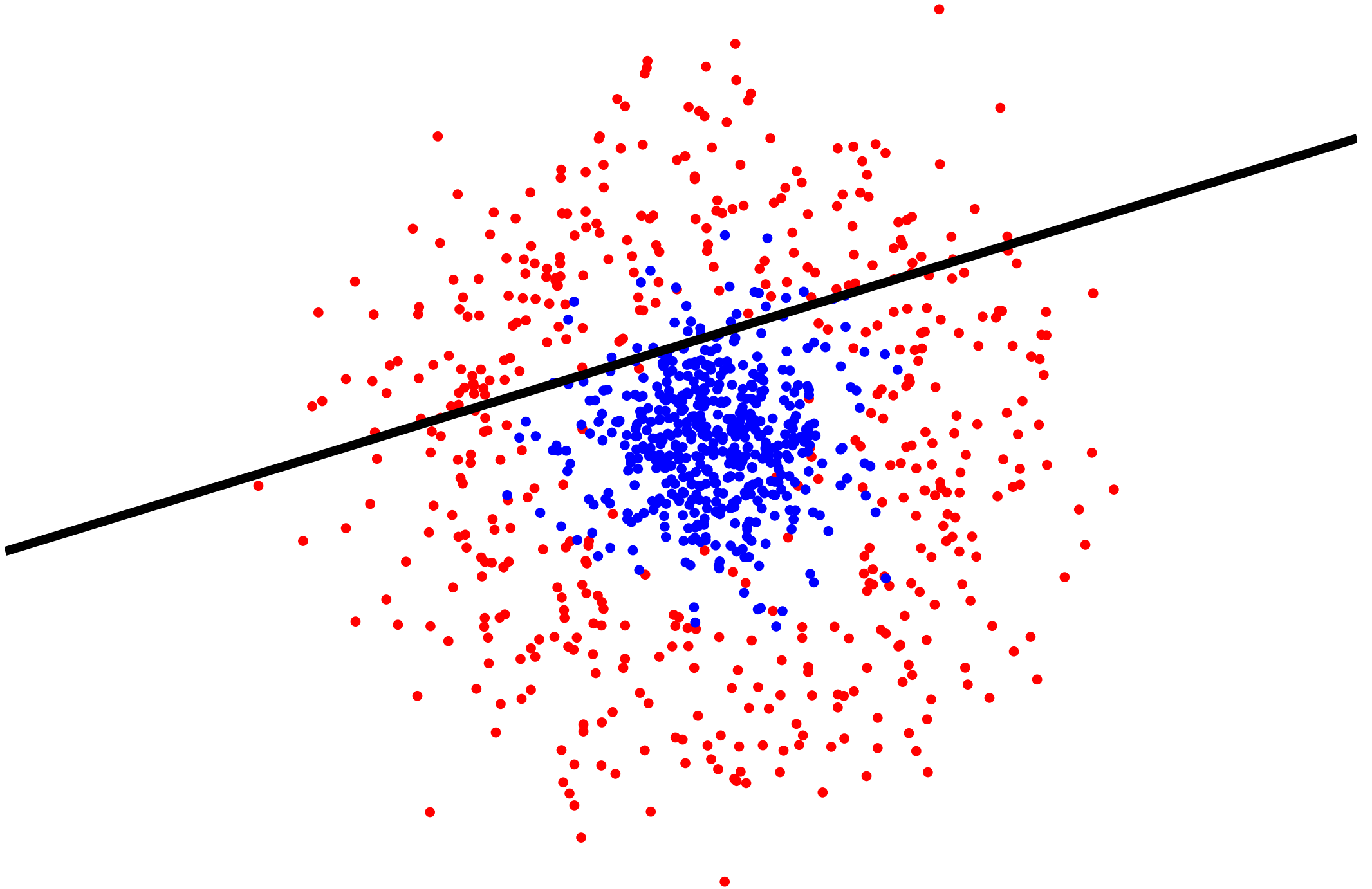


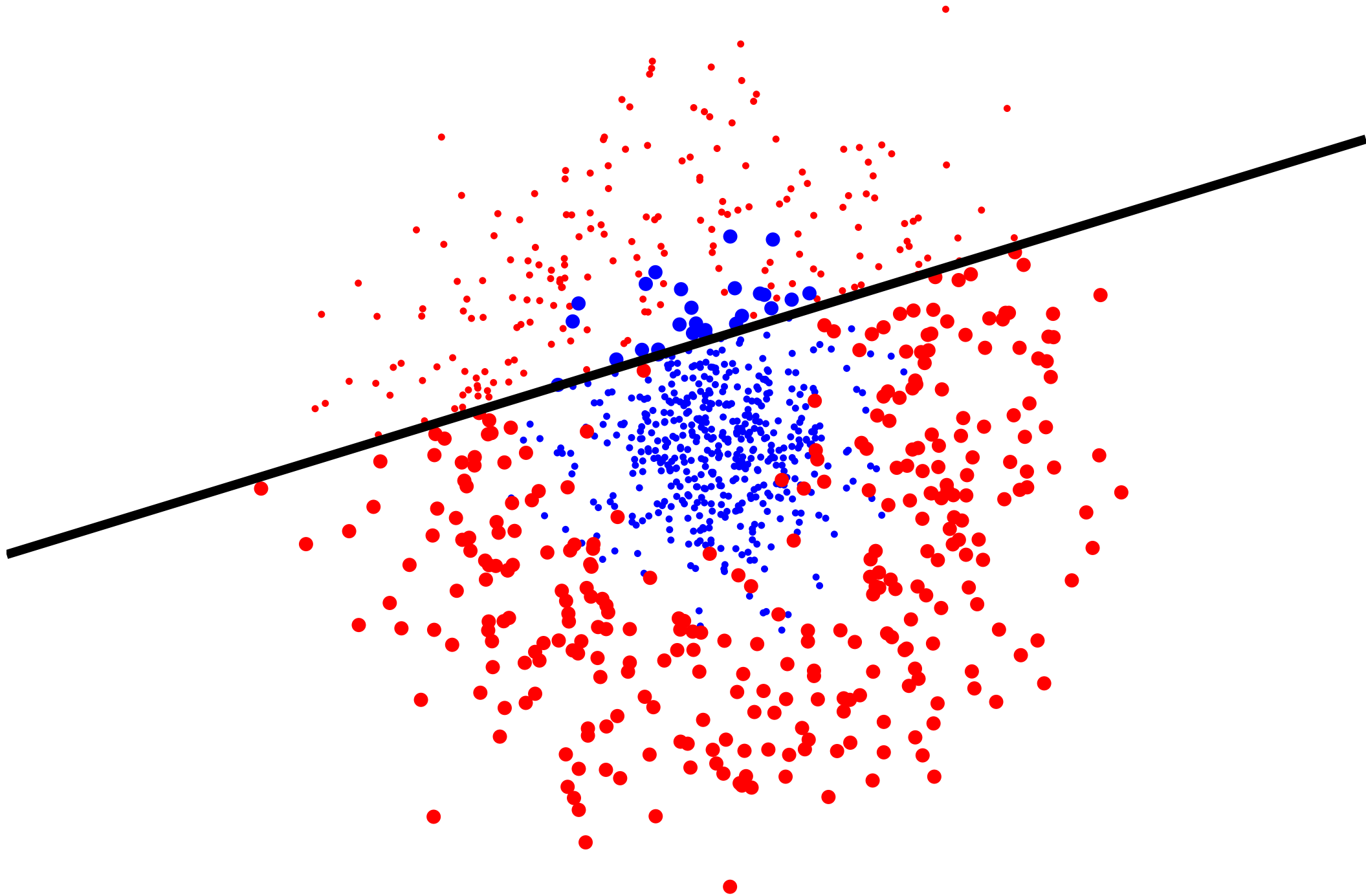


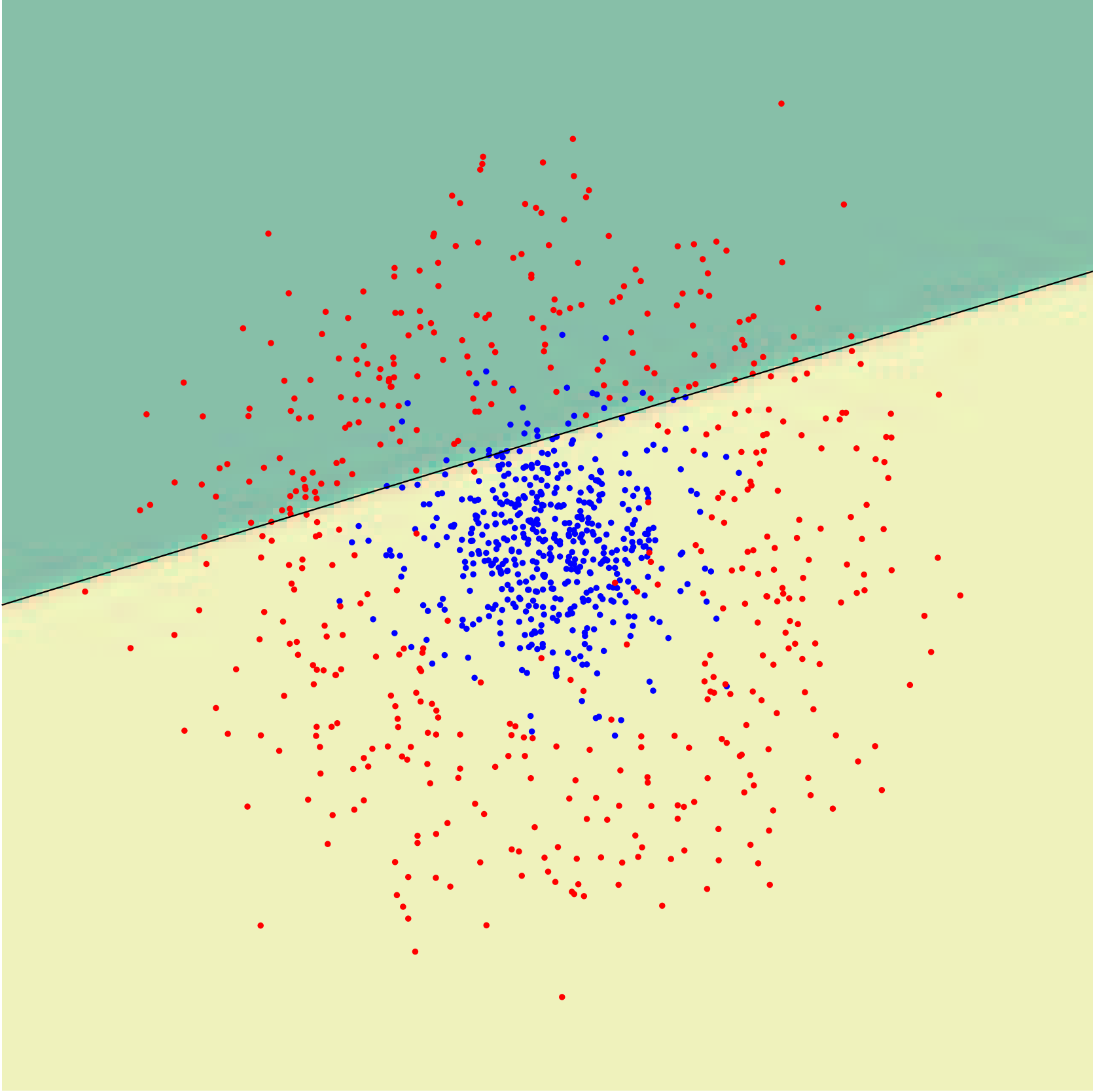


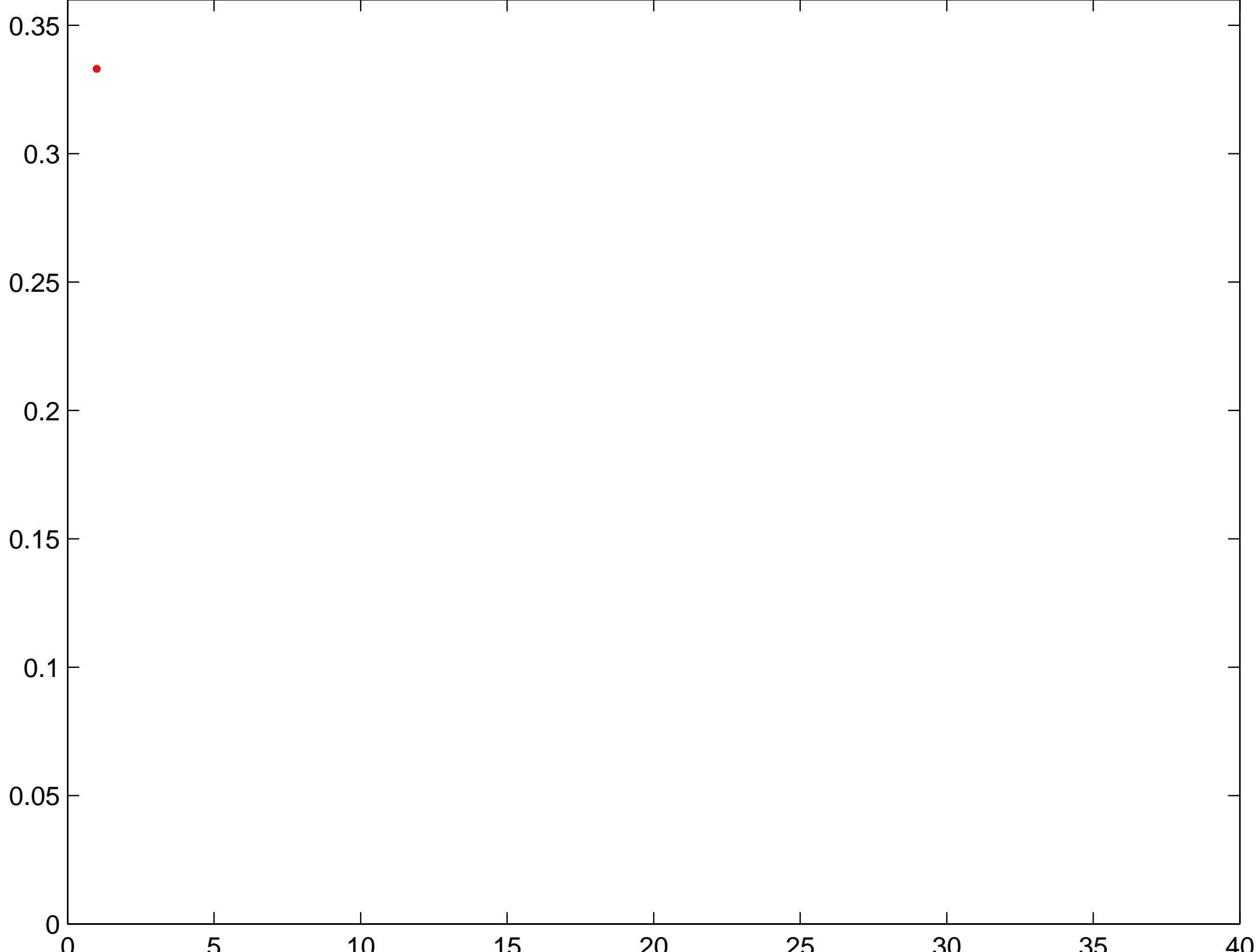


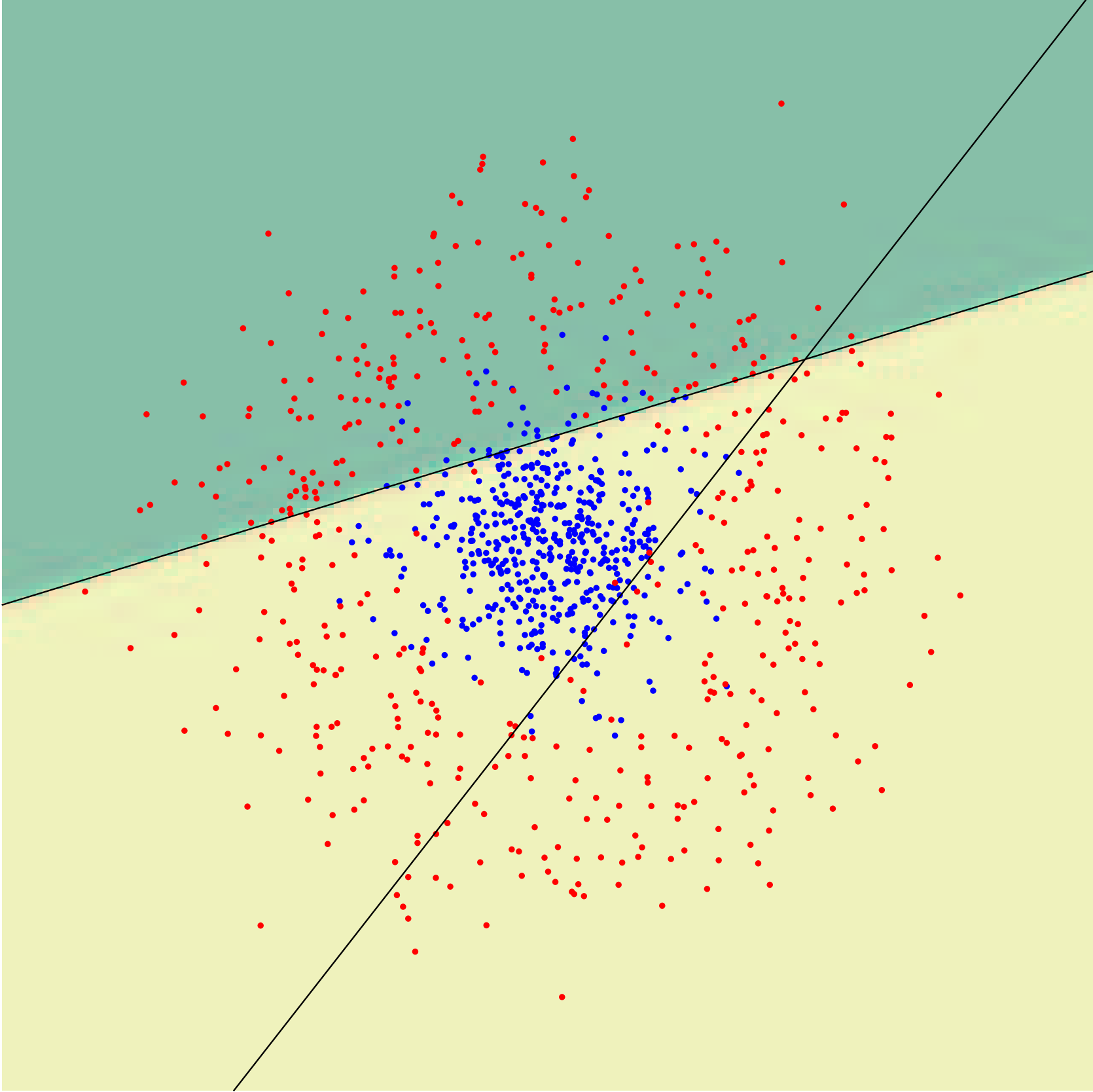


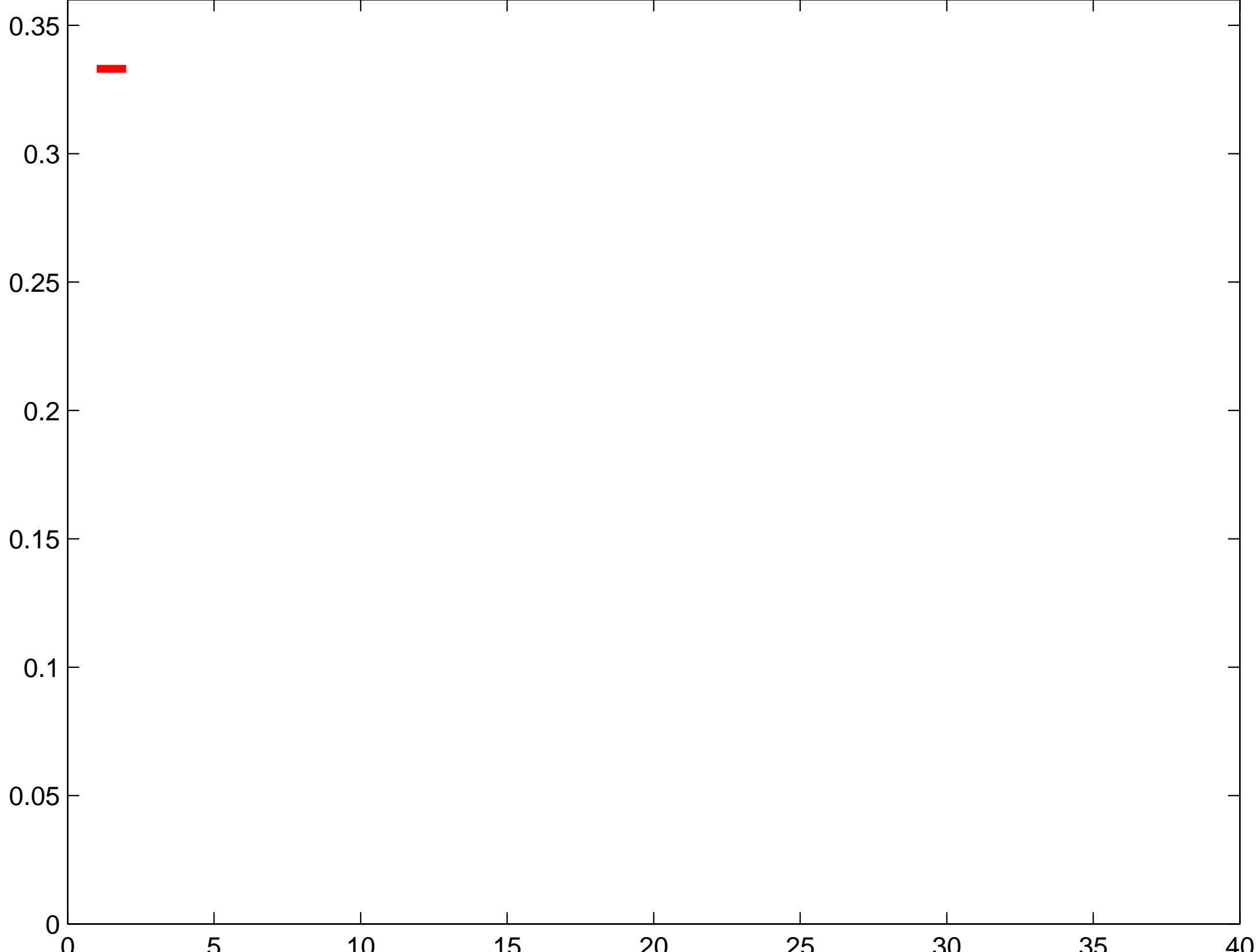


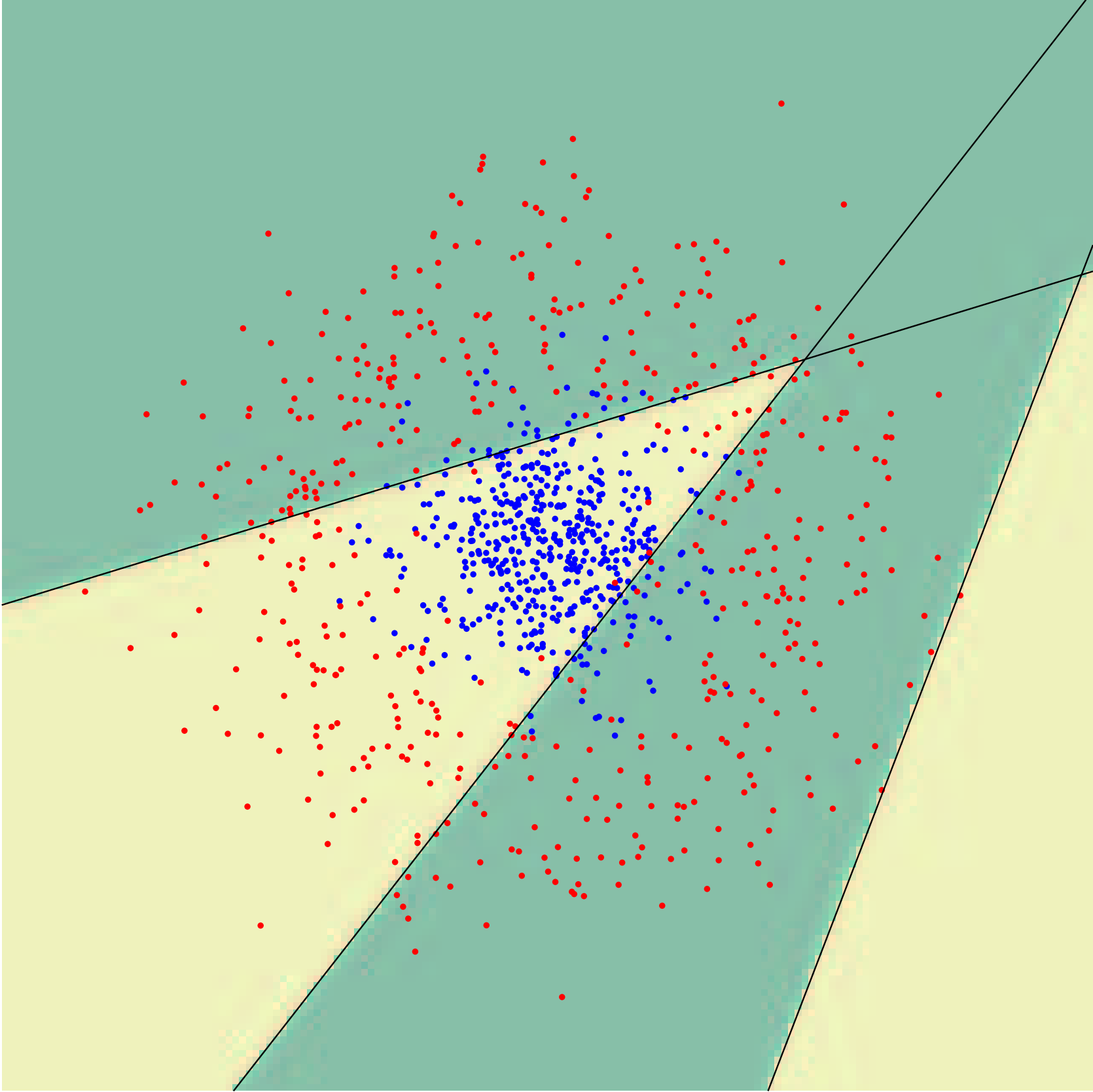


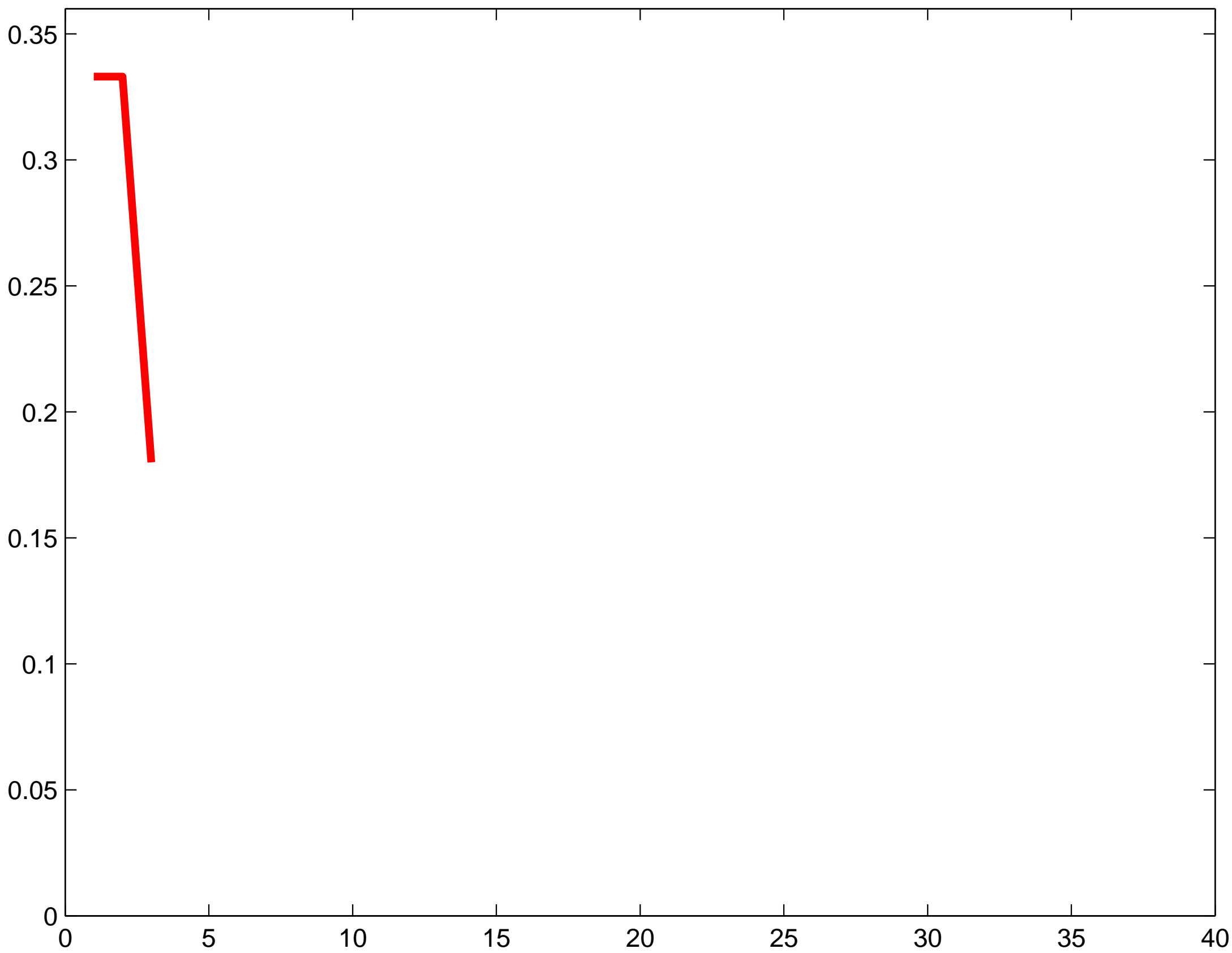


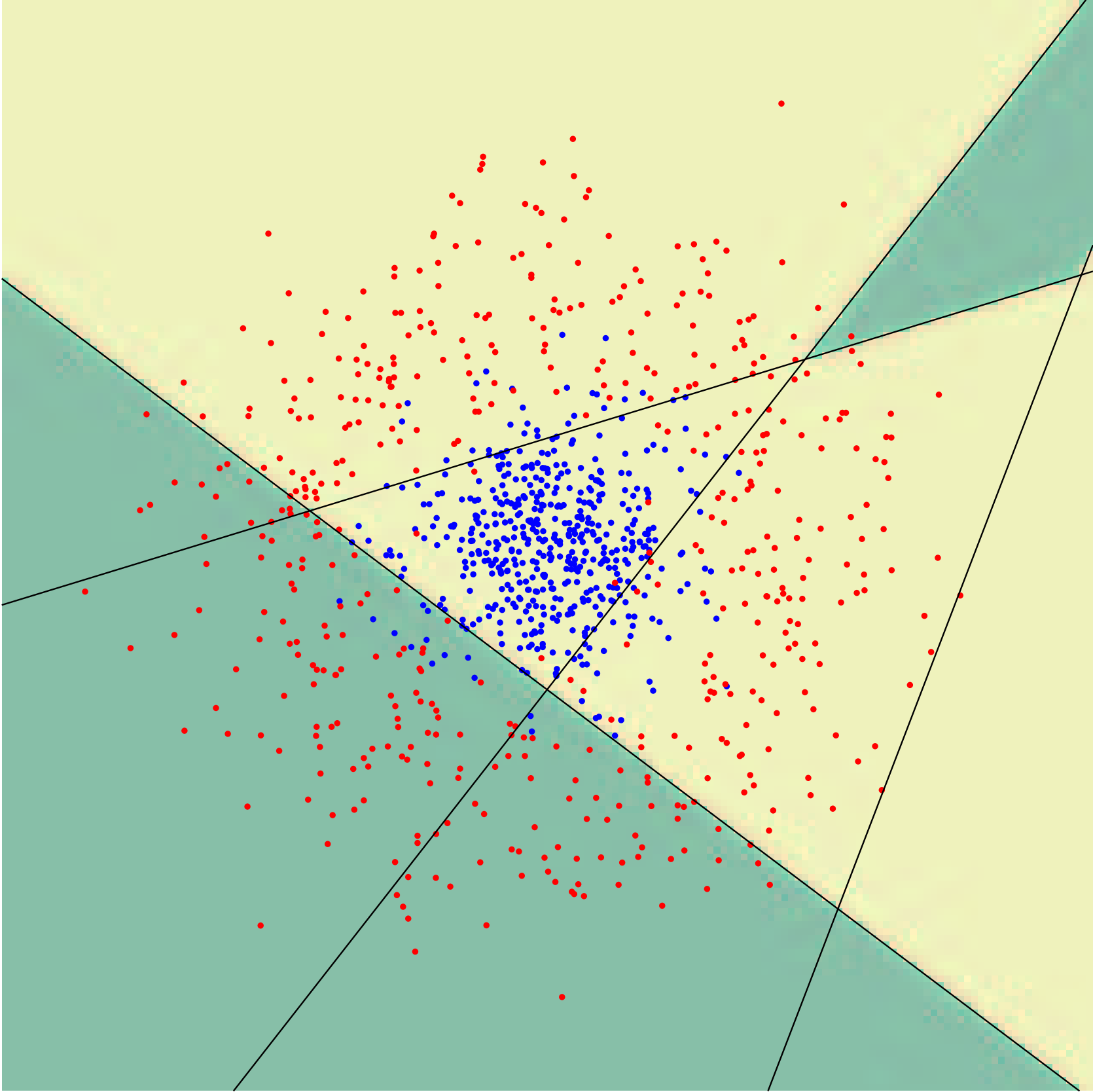


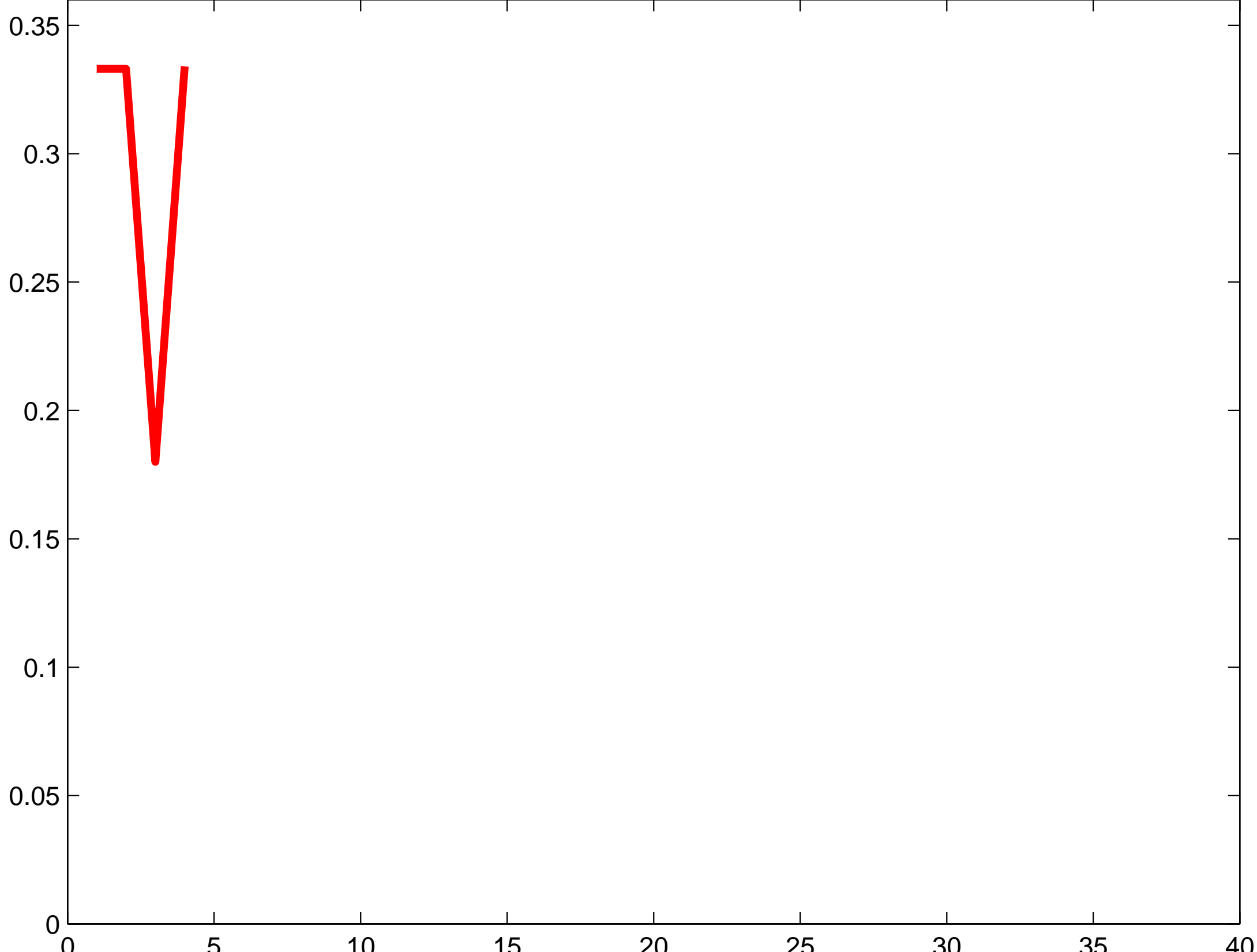


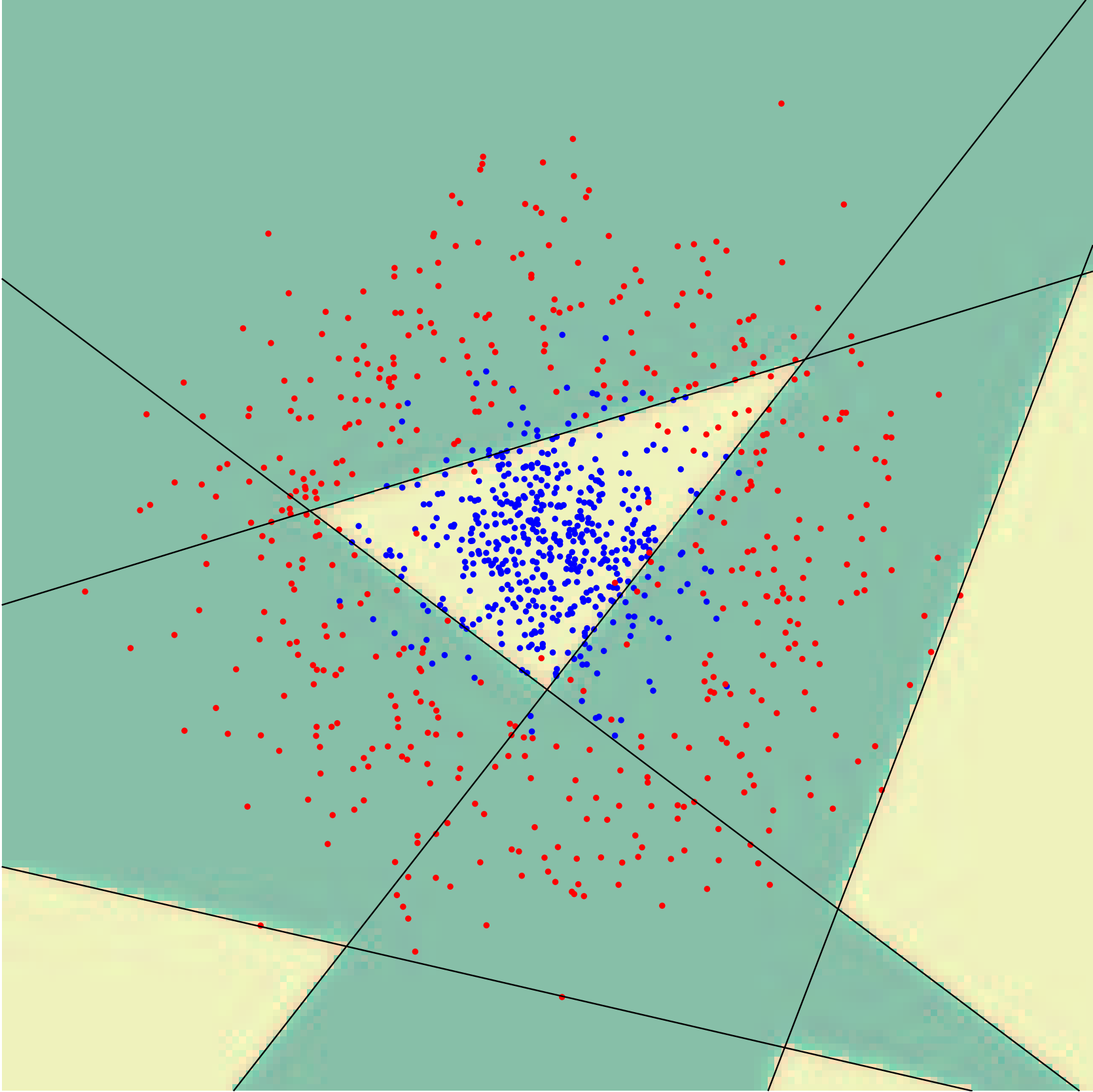


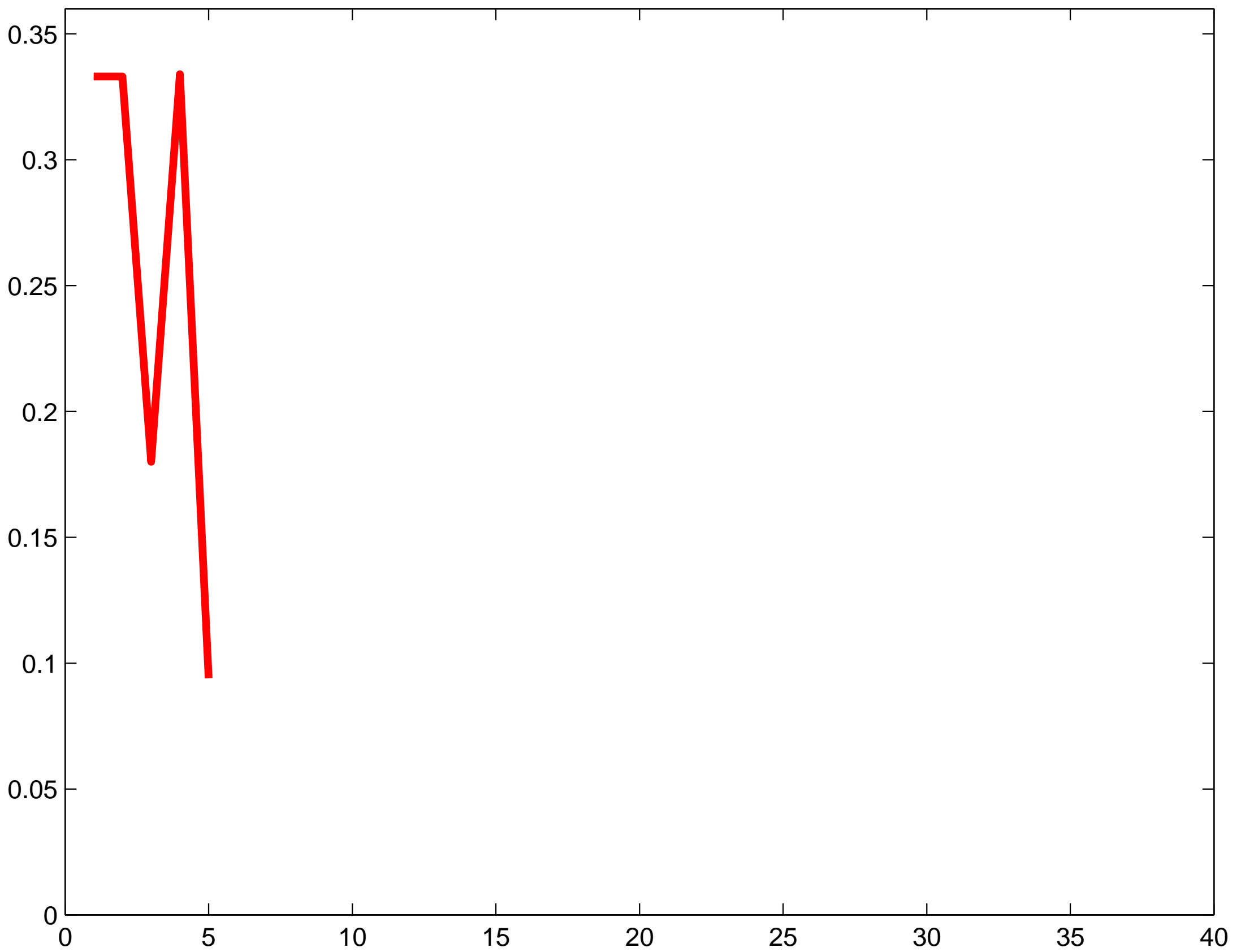


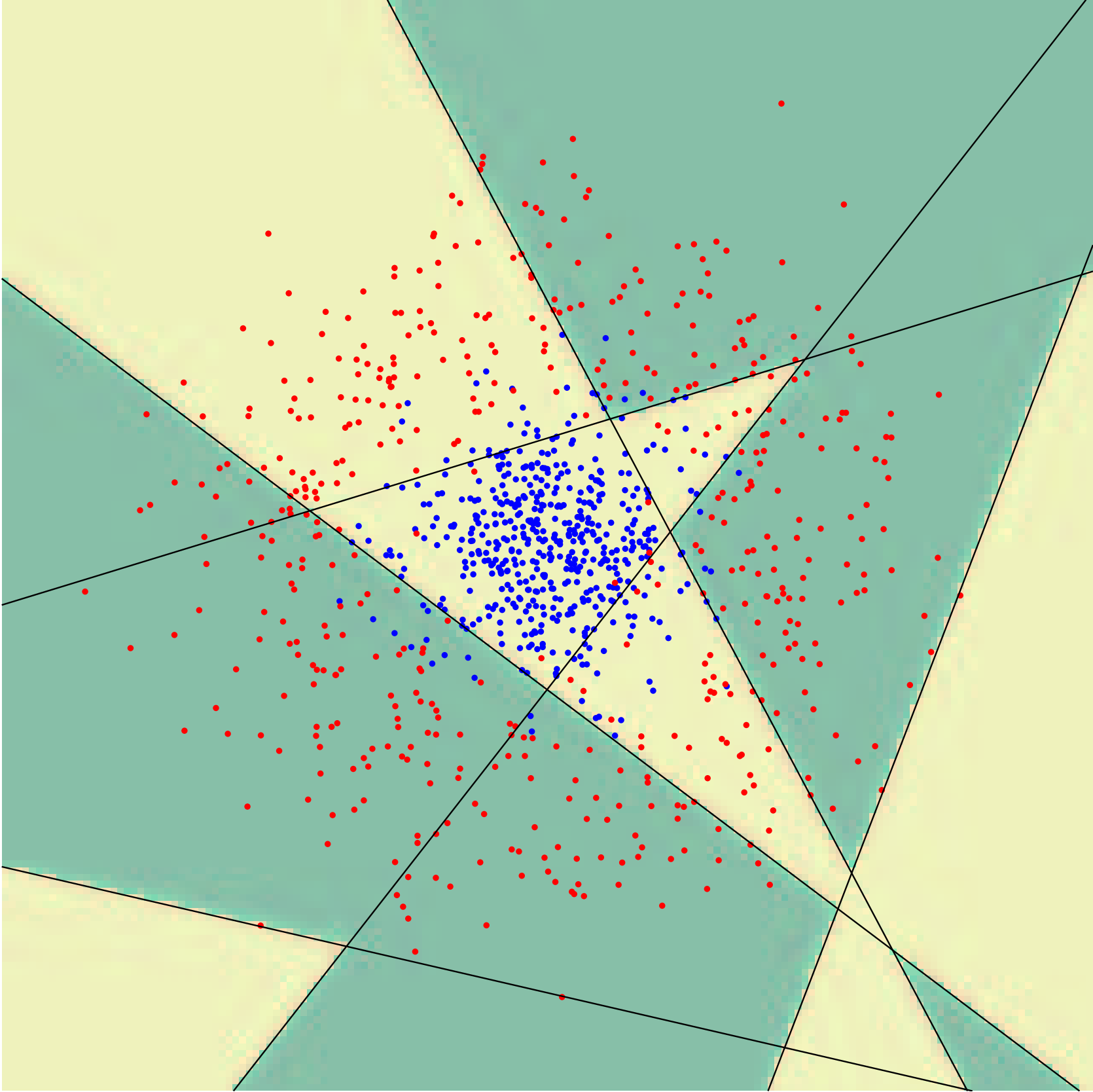


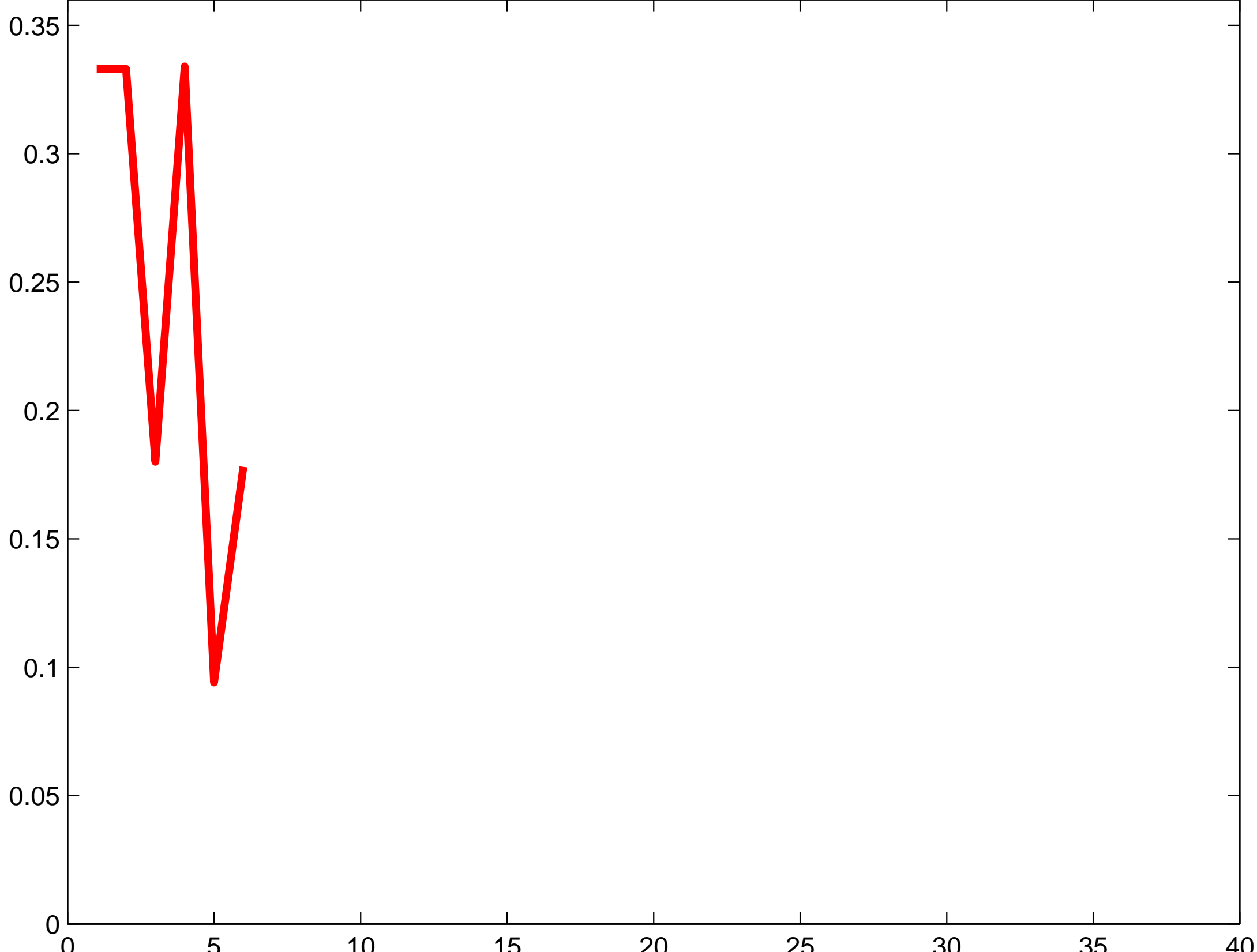


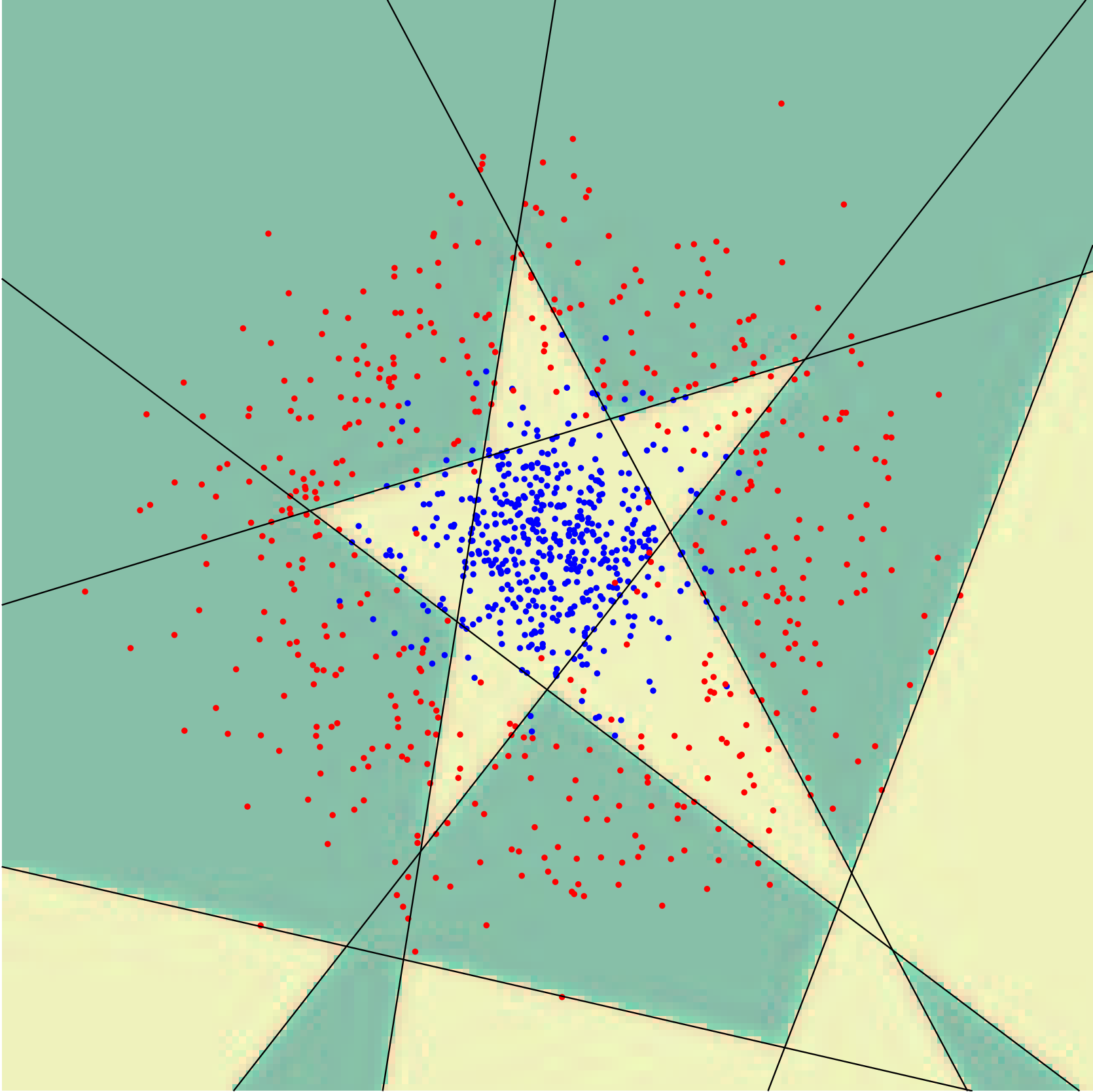


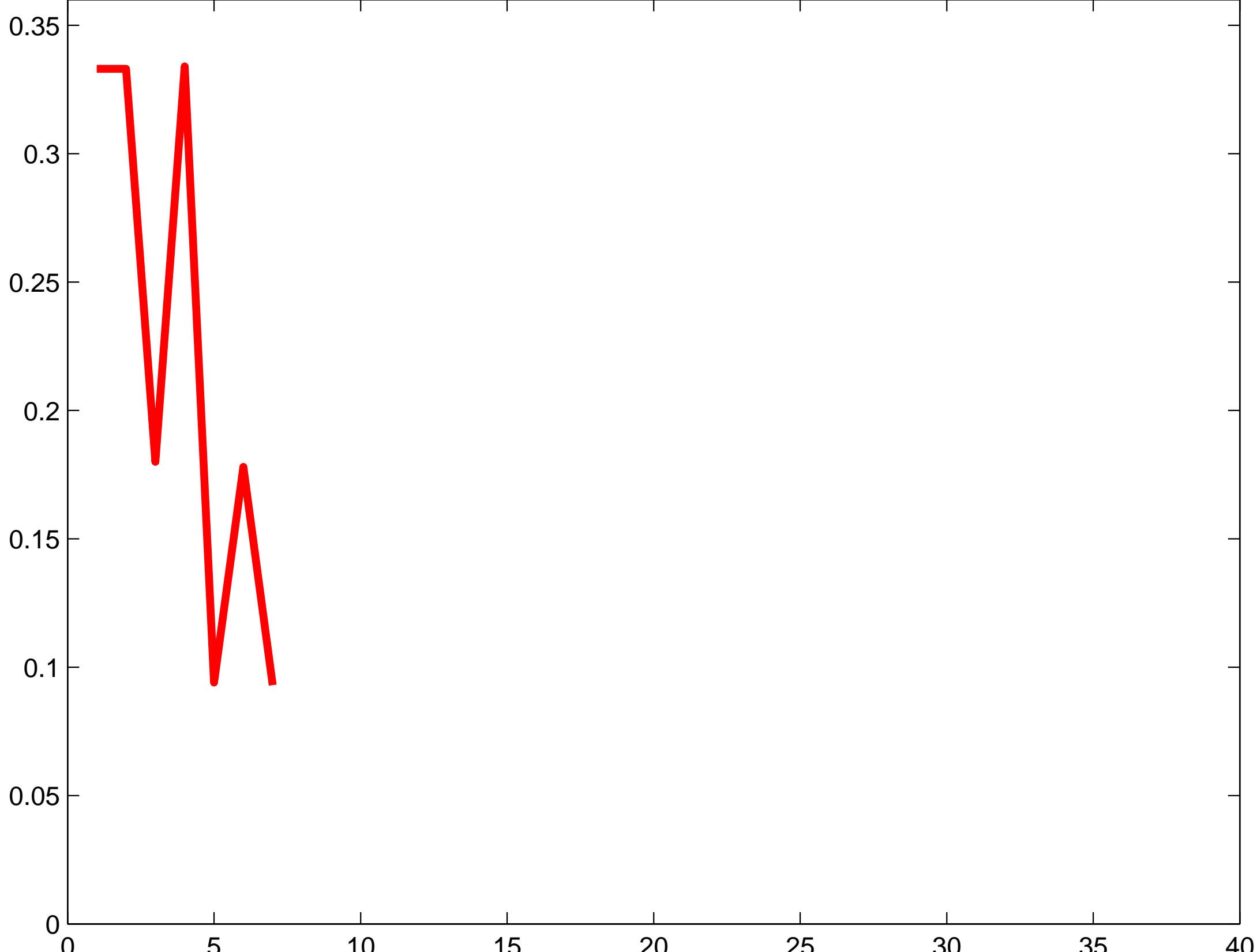


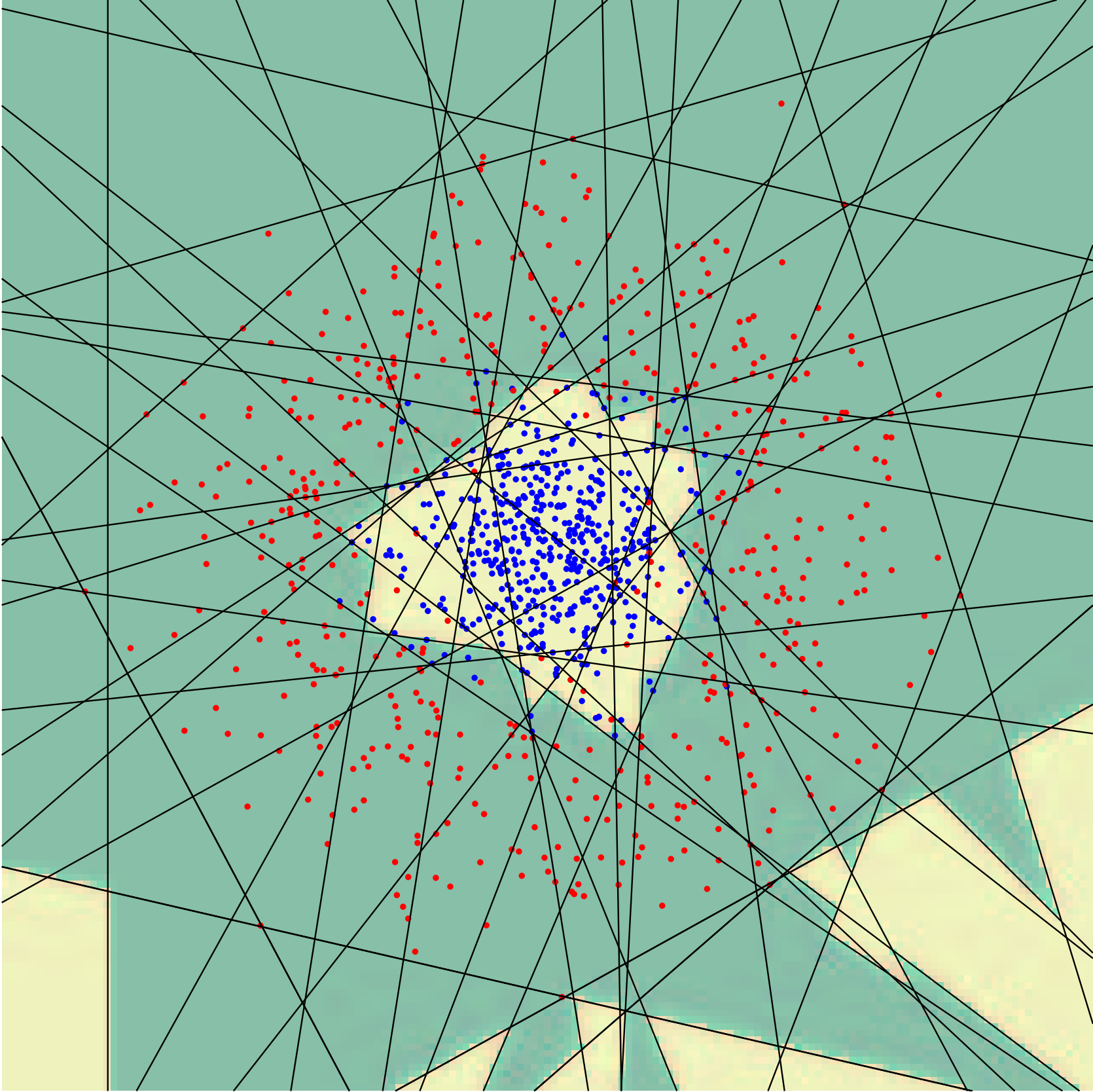


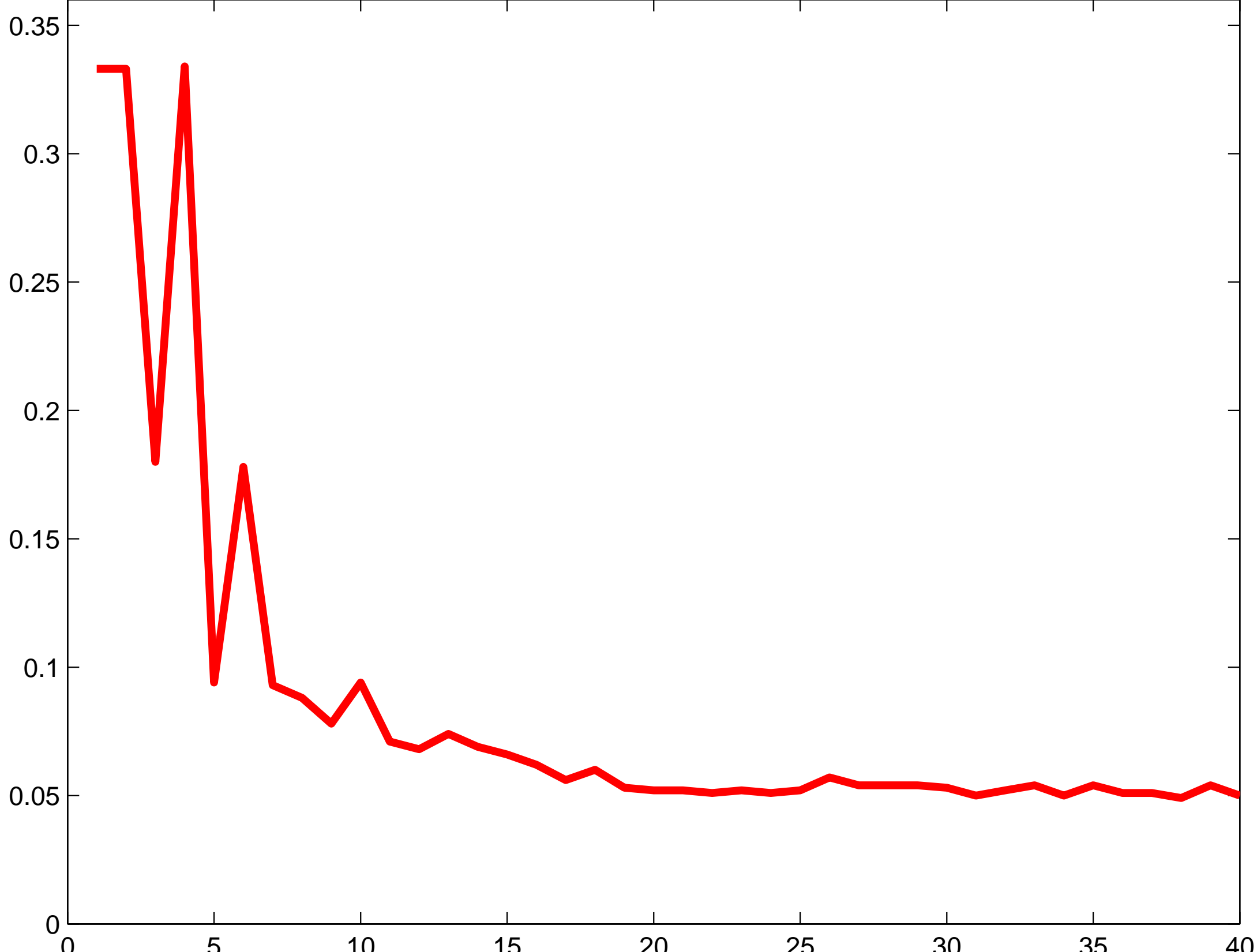




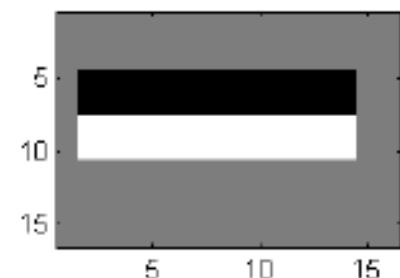




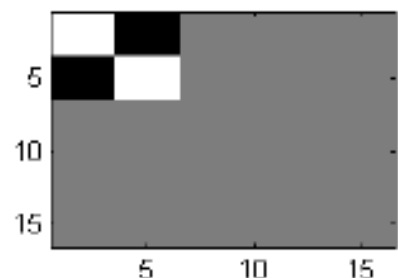




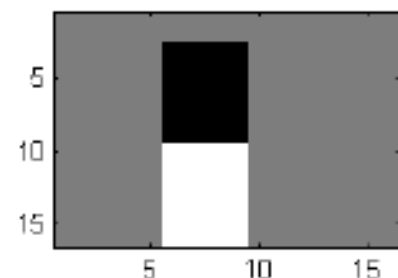
(1, 0.13, 1.93)



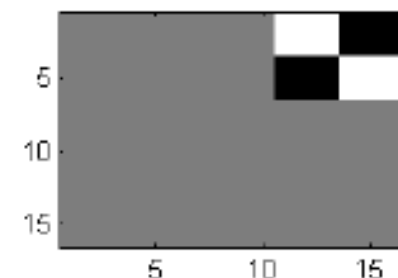
(2, 0.17, 1.59)



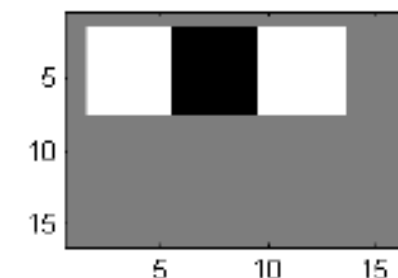
(3, 0.21, 1.30)



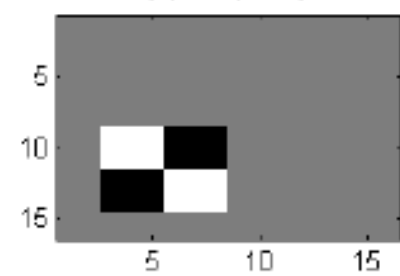
(4, 0.25, 1.12)



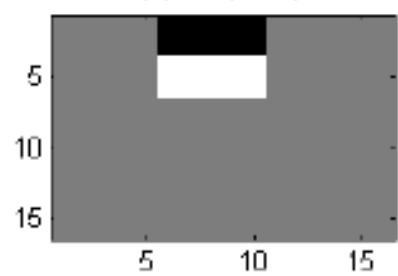
(5, 0.26, 1.05)



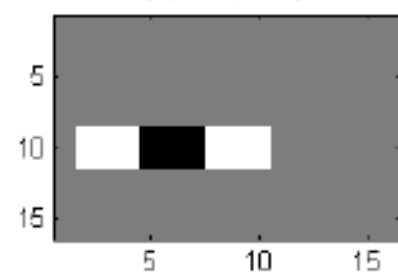
(6, 0.26, 1.04)



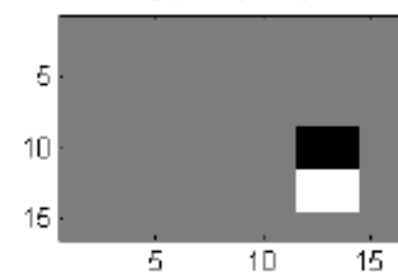
(7, 0.30, 0.82)



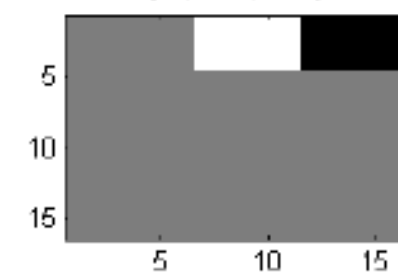
(8, 0.31, 0.79)



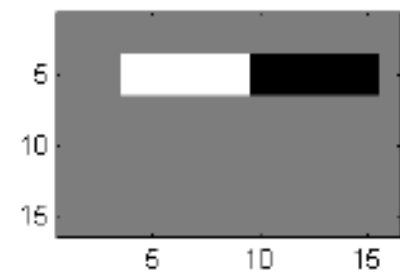
(9, 0.33, 0.72)



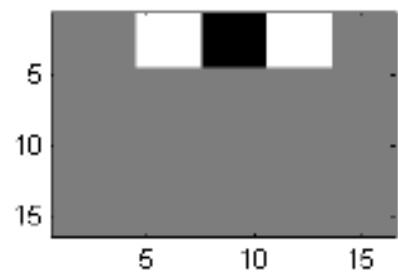
(10, 0.31, 0.81)



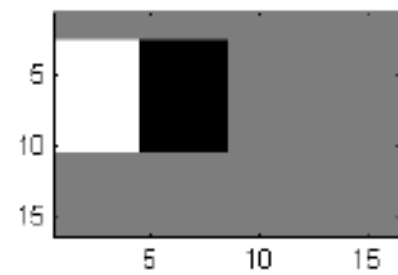
(11, 0.32, 0.78)



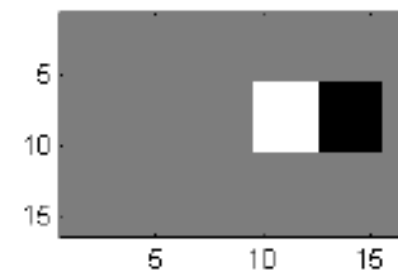
(12, 0.34, 0.65)



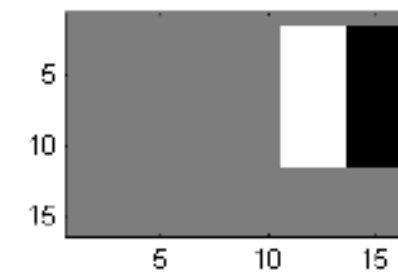
(13, 0.34, 0.64)



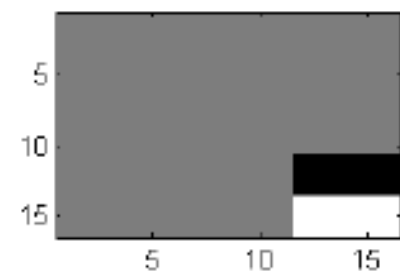
(14, 0.35, 0.60)



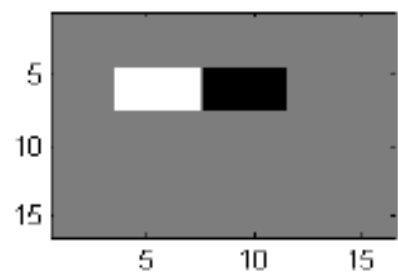
(15, 0.34, 0.67)



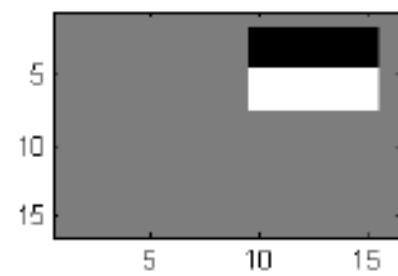
(16, 0.36, 0.59)



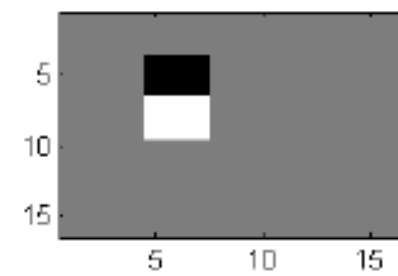
(17, 0.36, 0.57)



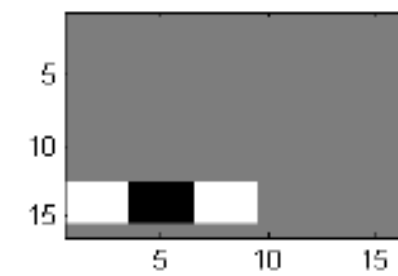
(18, 0.36, 0.56)



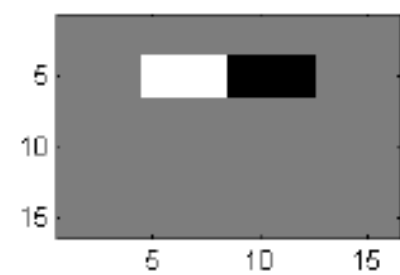
(19, 0.36, 0.57)



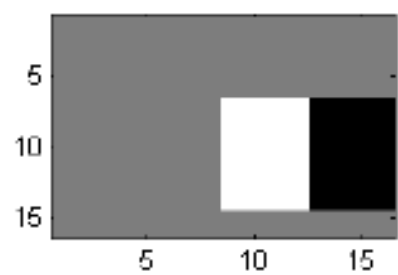
(20, 0.36, 0.56)



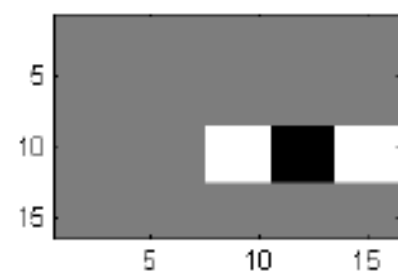
(21, 0.36, 0.56)



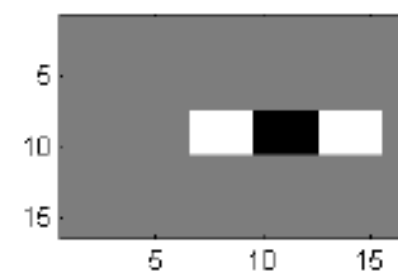
(22, 0.36, 0.55)



(23, 0.37, 0.52)



(24, 0.35, 0.61)



(25, 0.38, 0.49)

