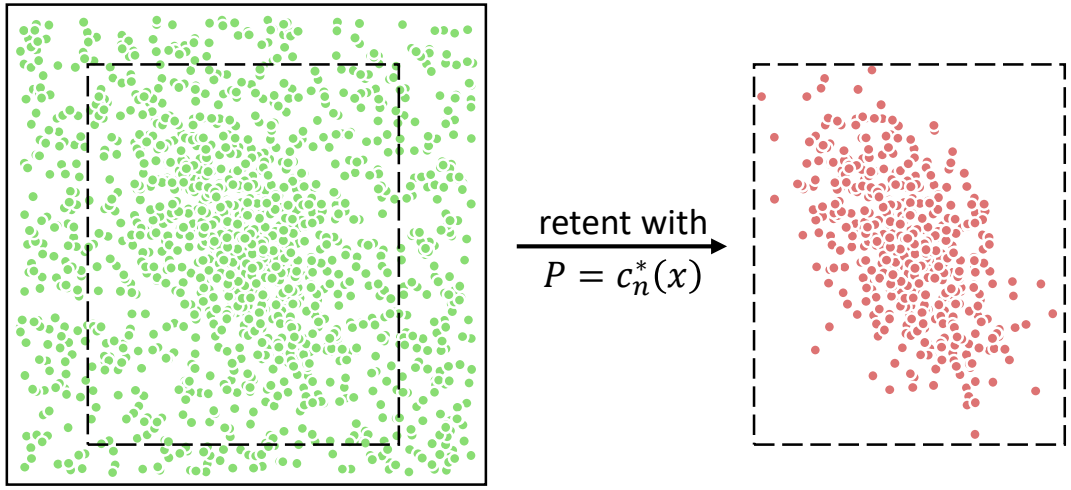
A consistent method for generating synthetic tabular data with a fully connected neural network



Interactive
perceptron visualizer



Perminov A. I. Kovalenko A. P. Turdakov D. Y.

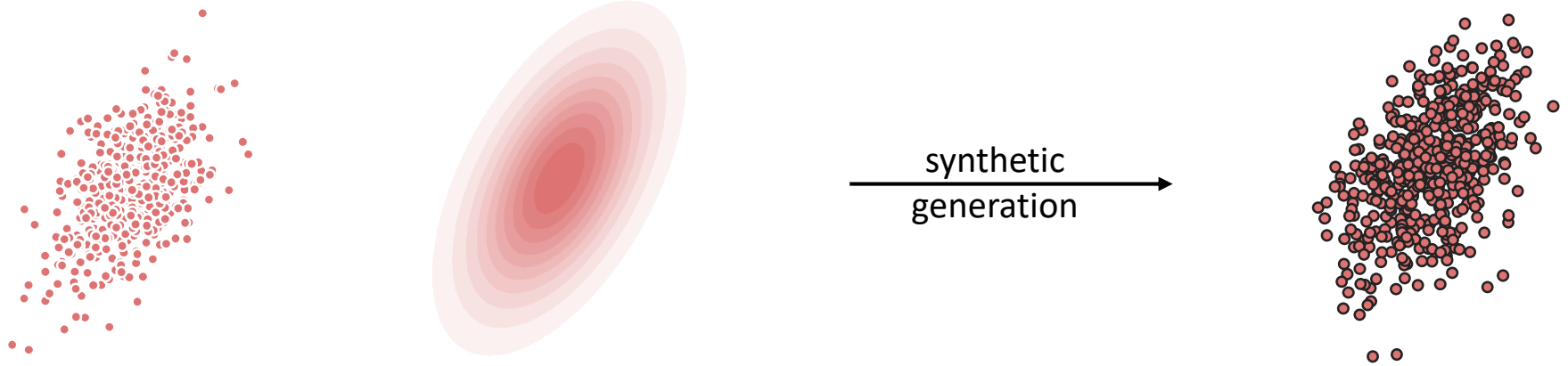
Ivannikov Institute for System Programming
of the Russian Academy of Sciences

ИСП РАН

Synthetic tabular data generation

- $X = \{x_1, x_2, \ldots, x_n\} \subset [0,1]^d$ – dataset from unknown probability distribution with uniform continuously density function $f(x)$;
- $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m\}$ – synthetic dataset.

Goal: construct new dataset $\bar{X}$, that approximates the statistical and structure properties of X while maintaining privacy constraints.

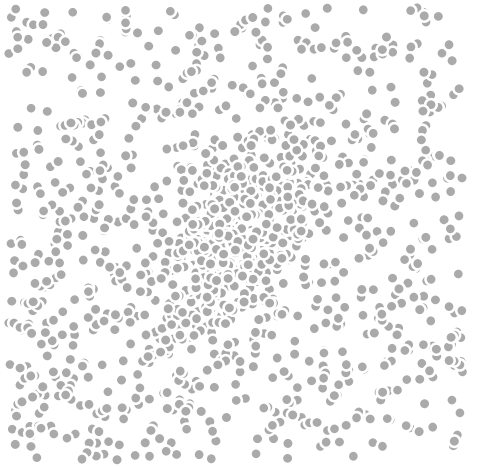


X $f(x)$ $\bar{X}$

Density estimation: approaches

Common approach:

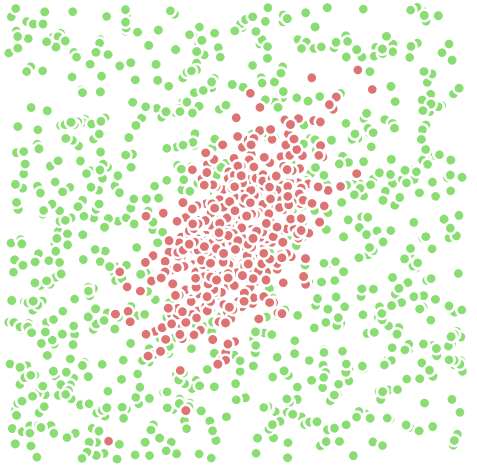
- Nonparametric estimator $\hat{f}(x)$ of $f(x)$: histogram, k-nearest neighbors (kNN), …



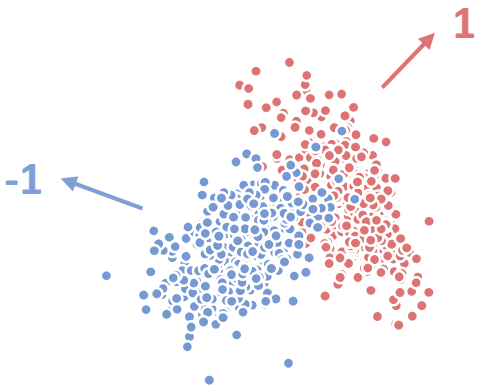
unary classification

Our approach:

- Based on **unary classification**;
- Dense neural network trained to distinguish support of the real data from a background distribution.



Bayesian binary classifier (classic)



- $D = (X, Y)$ – random vector with a certain distribution P;
- $X \subset [0,1]^d$, $Y \in \{\pm 1\}$;
- $f: [0,1]^d \to \{\pm 1\}$ – discriminant function;
- S – support of distribution of vector X.

General **classification** problem:

$$P\big(Y \neq f(X)\big) \to \min_f$$

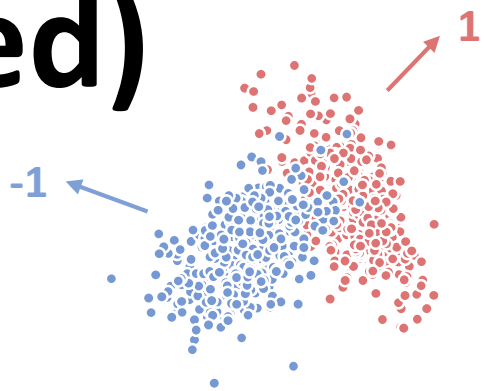
Bayesian classifier $g(x)$ – **optimal discriminant function** of classification problem:

$$s(x) = \begin{cases} 1, & if\ g(x) > 0\ and\ x \in S \\ \boldsymbol{any}\ of\ \pm 1, & if\ g(x) = 0\ or\ x \notin S \\ -1, & if\ g(x) < 0\ and\ x \in S \end{cases}$$

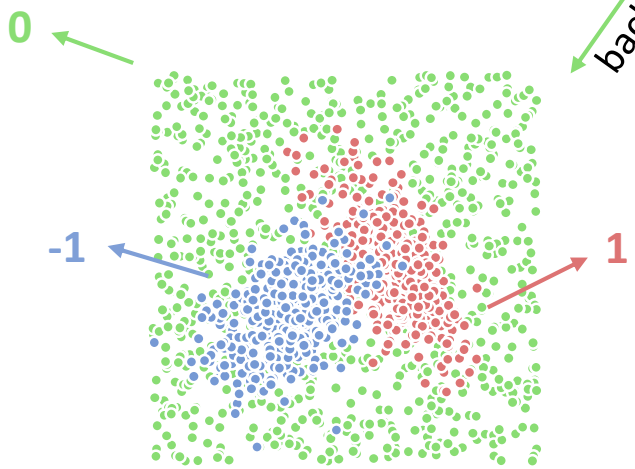
Bayesian binary classifier (modified)

Add m "background" vectors $(\check{X}, \check{Y})$:

- $\check{X} \subset [0,1]^d$ – uniformly distributed noise on the compact domain K;
- $\check{Y} \in \{0\}$ – label of "background" class.



add background



Mixture of two distributions:

- $(X, Y) \in [0,1]^d \times \{\pm 1, 0\}$ – new data-vector;
- $P_\alpha = \alpha P + (1-\alpha)\check{P}, \alpha \in (0,1)$;
- $\check{P}$ – distribution of the background.

Optimal solution $\check{s}(x)$ based on modified dataset **does not worsen** $g(x)$, but $\check{s}(x)$ **equals to 0** if $x \notin S$.

Unary classification: mixture data

- (X, Y) – random variable;
- $X \subset [0,1]^d$ – vector with a mixture density $\alpha f(x) + (1-\alpha)p(x)$;
- $Y = 0$ for background distribution and $Y = 1$ for target class label;
- $f(x)$ – density of target class;
- $p(x)$ – density of the uniform distribution over a compact set K;
- α – weighting coefficient, $\alpha \in [0,1)$;

Posterior probability (or regression function) of Y given X:

$$g(x) = P(Y = 1 | x = x) = E(Y | X = x) = \frac{\alpha f(x)}{\alpha f(x) + (1-\alpha)p(x)}$$

Unary classification: problem solution

- $g(x)$ **known** (wow!):

  x part of distribution if $g(x) > 0$, otherwise reject.

- $g(x)$ **unknown** (typically):

  construct an **approximation** from the given data.

Unary classification: approximation

- $c(x)$ – a continuous function defined on a compact set K;

Mean-squared approximation problem (minimization takes over all $c(x)$):

$$c^*(x) = \operatorname{argmin} \mathrm{E}(\mathrm{c}(\mathrm{x}) - \mathrm{Y})^2 \quad \textbf{(2)}$$

$$E(c(x) - Y)^2 = E(c(x) - g(x) + g(x) - Y)^2$$

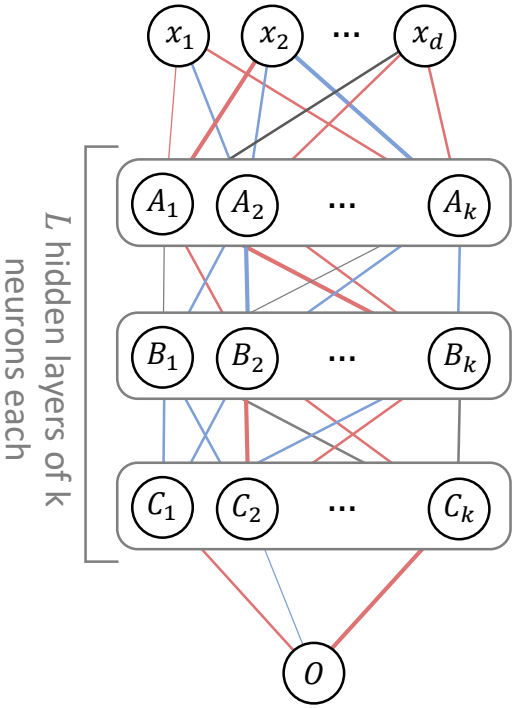
$$E(c(x) - Y)^2 = E\big(c(x) - g(x)\big)^2 + \underbrace{E(g(x) - Y)^2}_{\text{independent of } c(x)}$$

Problem **reduces** to the approximation of the regression function:

$$c^*(x) = \operatorname{argmin} \mathrm{E}(\mathrm{c}(\mathrm{x}) - g(x))^2 \quad \textbf{(4)}$$

Unary classification: perceptron

- $c(x)$ – multilayer **perceptron** with L hidden layers of k neurons each, one neuron in the output layer and piecewise-linear activation function (Abs, $ReLU$, …).



Universal approximation theorem: for any $\epsilon > 0$ there exist values of k and L such that for any $x \in K$:

$$\sup|c(x) - g(x)| < \epsilon$$

Thus, an ϵ-approximated solution of **(2)** **theoretically exists**.

Cybenko G. Approximation by superpositions of a sigmoidal function (1989)

Unary classification: perceptron

- $\{X_i, 1\}_{i=1}^{n}$ – labeled set of n observations from **target** density $f(x)$;
- $\{X_j, 0\}_{j=n+1}^{n+m}$ – m added **background** samples, $m = n \cdot \frac{1-\alpha}{\alpha}$;
- $C(L, k)$ – class of **MLP** with activation function $\boldsymbol{Abs}$ and given $\boldsymbol{L}$, $\boldsymbol{k}$;

Optimization problem (minimization takes over all $c_n(x) \in C(L, k)$):

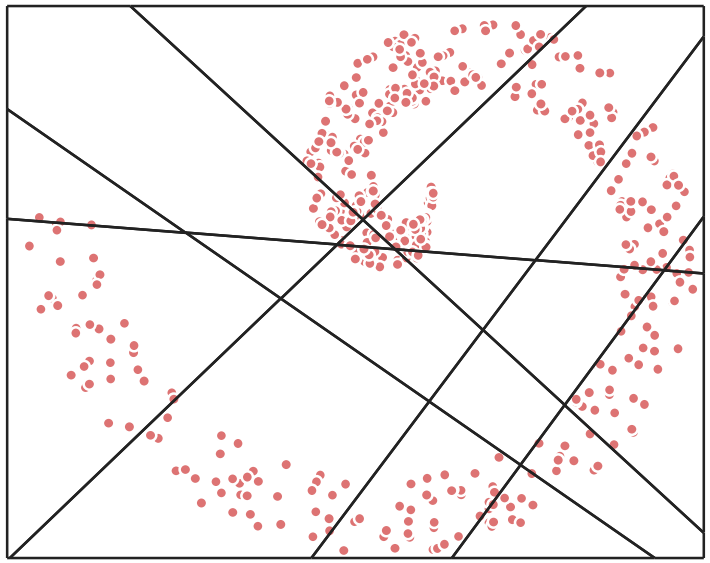
$$\sum_{i=1}^{n+m} (c_n(X_i) - Y_i)^2 \to min \qquad (6)$$

Let $c_n^*(x)$ – is solution of (6), called **neural regression function**.

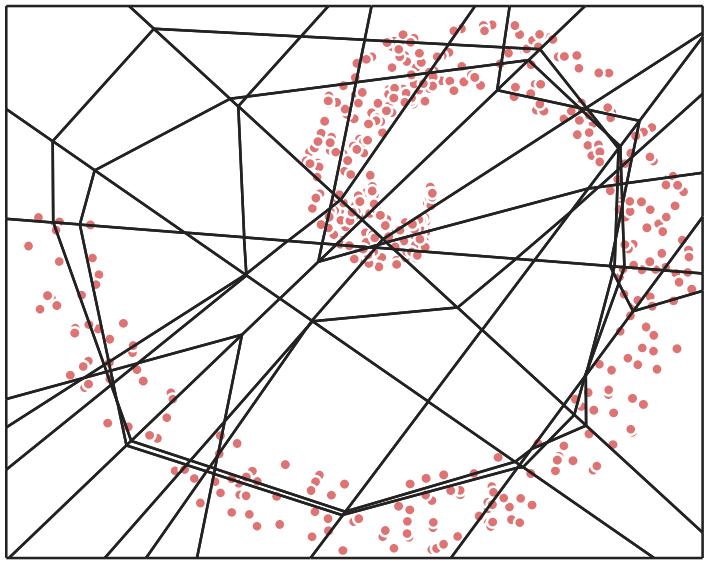
Unary classification: adaptive histogram

- $c_n^*(x)$ constructs a **hierarchical partition** of K into N disjoint cells: $K = \{K_1, K_2, \dots, K_N\}$

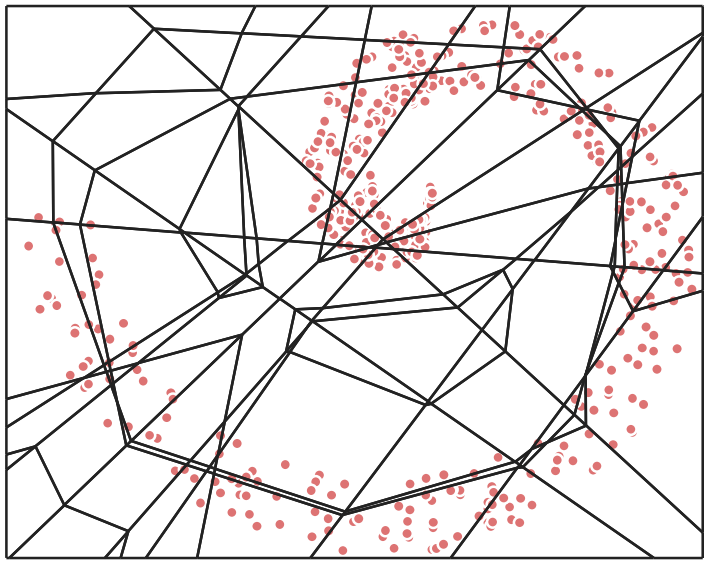
Partition example of some MLP with $L = 2$, $k = 6$ and Abs activation:



Cells after first hidden layer:
18 cells



Cells after second hidden layer:
121 cells



Cells after output neuron:
148 cells

Unary classification: adaptive histogram

- $h_n(x)$ – **piecewise constant** function of histogram regression, defined by minimizing:

$$\sum_{i=1}^{n+m} (h_n(X_i) - Y_i)^2 \to min \qquad (7)$$

Minimization takes over all **piecewise constant** functions defined on cells K_r of K.

- Let $X \in K_r \to$ within each cell K_r the problem **(7)** reduces to

$$n_1(X) \cdot (h_n(X) - 1)^2 + n_0(X) \cdot (h_n(X) - 0)^2 \to min$$

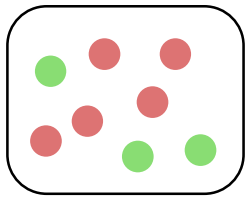
- $n_j(X) = \sum_{i=1}^{n+m} I_{\{X_i \in K_r, Y_i = j\}}$;

Unary classification: adaptive histogram

Differentiating with respect to $h_n(x)$ yields the solution of **(7)**:

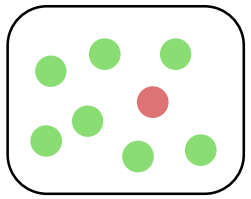
$$h_n^*(x) = \frac{n_1(X)}{n_1(X) + n_0(X)} = \frac{f_n(X)}{f_n(X) + \frac{1-\alpha}{\alpha} p_n(X)}$$

- $f_n(X) = \frac{n_1(X)}{n \cdot V(K_r)}$ – histogram density $f(x)$ estimates at cell K_r;
- $p_n(X) = \frac{n_0(X)}{n \cdot V(K_r)}$ – histogram uniform density $p(x)$ estimates at cell K_r;
- $V(K_r)$ – measure of the cell K_r.



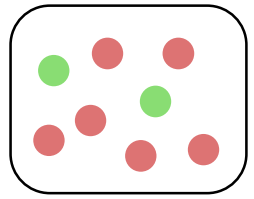
$n_1 = 5, n_0 = 3$

$h_n^*(X) = \frac{5}{5+3} = 0.625$



$n_1 = 1, n_0 = 7$

$h_n^*(X) = \frac{1}{1+7} = 0.125$



$n_1 = 6, n_0 = 2$

$h_n^*(X) = \frac{6}{6+2} = 0.75$

Unary classification: consistency

- **Cell shrinking**: diameter decreases while keeping enough points per cell;
- **Practical**: for $d = 10$, $k = 10$, $L = 2$, $N > 10^4$ – millions of background points are needed;
- **Filled cells** (both target and background points present): $h_n^*(X) \approx c_n^*(X)$;
- **Background-only cells**: $h_n^*(X) = 0$, $c_n^*(X)$ is interpolated (continuity);
- **High-density regions**: $c_n^*(X) \gg 0$;
- **Low-density regions**: $c_n^*(X) \to 0$;

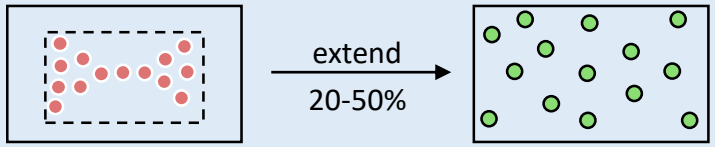
Sufficiently large n and appropriately chosen k and L, the neural regression $c_n^*(X)$ **may be interpreted** as a consistent estimator of $g(X)$.

Devroye L. Nonparametric density estimates with improved. performance on given sets of densities (1989)

Synthetic data generation: methodology

Background data generation

- **Determine compact**: $K \subset [0,1]^d$ – an axis-aligned hyper rectangle extending 20-50% beyond the real data in each dimension);
- **Margin effect**: ensures enough negative samples for the classifier to distinguish data support and helps push the decision function toward the zero plane.
- **Point set**: $B = \{b_1, b_2, \dots b_n\}$ are sampled uniformly from K with $|B| = |X|$ $(\alpha = 0.5)$



Classifier training

- **Classifier**: MLP $c_n(x)$: $[0,1]^d \rightarrow [0,1]$;
- **Dataset**: $X \cup B$ with binary labels:

$$c_n(x) = \begin{cases} 1, x \in X \\ 0, x \in B \end{cases}$$

- **Loss function**: mean squared error

$$L = \sum_{x \in X} \big(1 - c_n(x)\big)^2 + \sum_{b \in B} \big(0 - c_n(b)\big)^2$$

- **Dynamic background**: points B are generated at each epoch;

Why MSE?

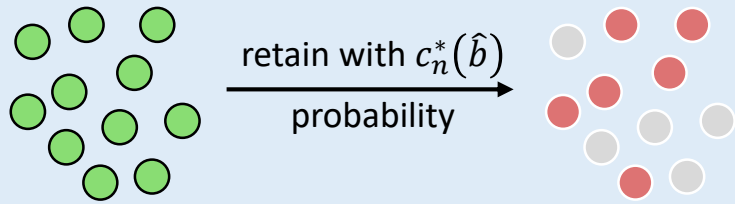
- Smoother approximation of posterior probabilities without requiring sigmoid;
- Supports histogram based interpretation.

Synthetic data sampling

- **Candidate points**: sample points $\hat{B} = \{\hat{b}_1, \hat{b}_2, \dots\}$ uniformly from K.
- **Prediction**: get value of $c_n^*(\hat{b})$ for each point $\hat{b} \in \hat{B}$
- **Retention rule**: point $\hat{b}$ is kept with probability $c_n^*(\hat{b})$:

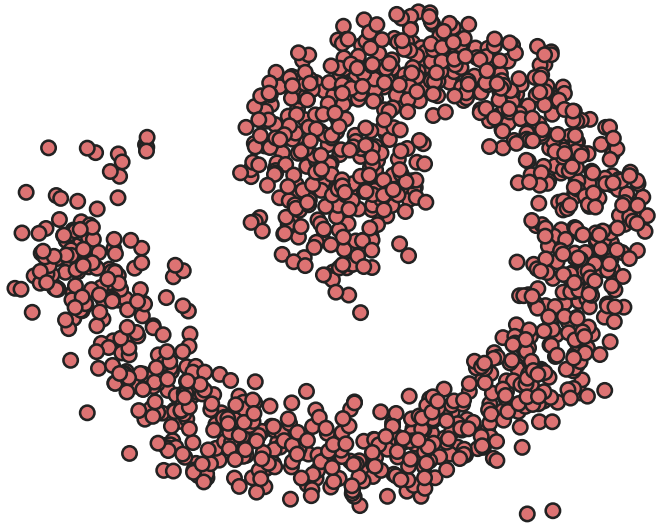
$$\hat{X} = \{\hat{b} \in \hat{B} \mid \xi < c_n^*(\hat{b})\}$$

ξ – is a random variable from uniform distribution $U(0,1)$

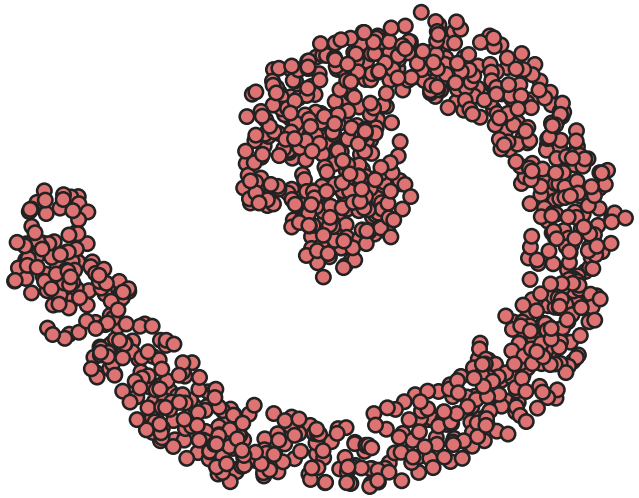


Trust threshold

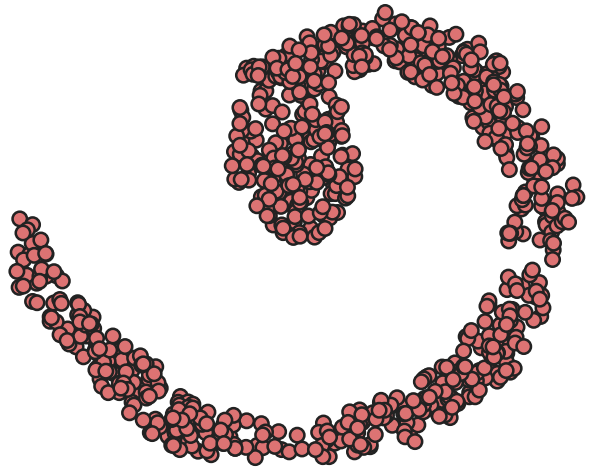
- **Threshold $\boldsymbol{\beta} \in [\mathbf{0}, \mathbf{1})$**: defines regions where the classifier output is unreliable;
- **Smooth decision surface**: $c_n^*(x)$ can not sharply drop to zero outside the data support;
- **Untrusted regions**: occur where $c_n^*(x) < \beta$ (exclude such points from $\hat{X}$);
- **Variance**: increasing the threshold reduces the variance of the synthetic data.



$\beta = 0$

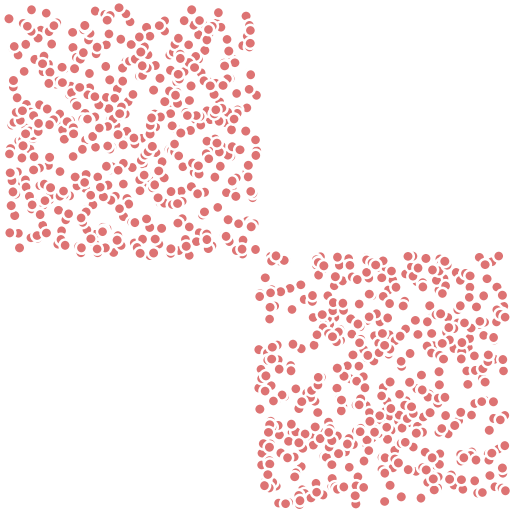


$\beta = 0.3$

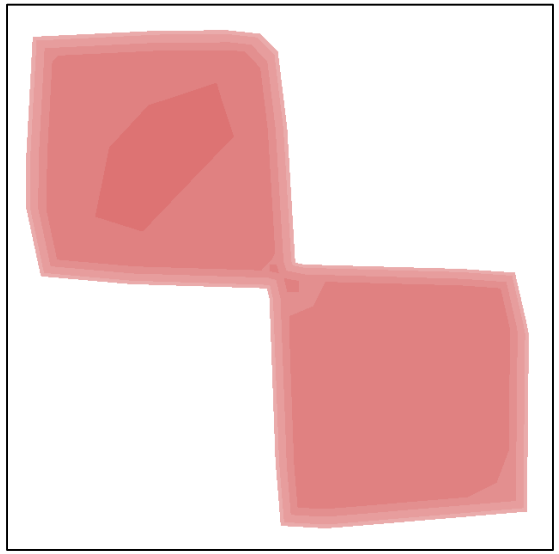


$\beta = 0.6$

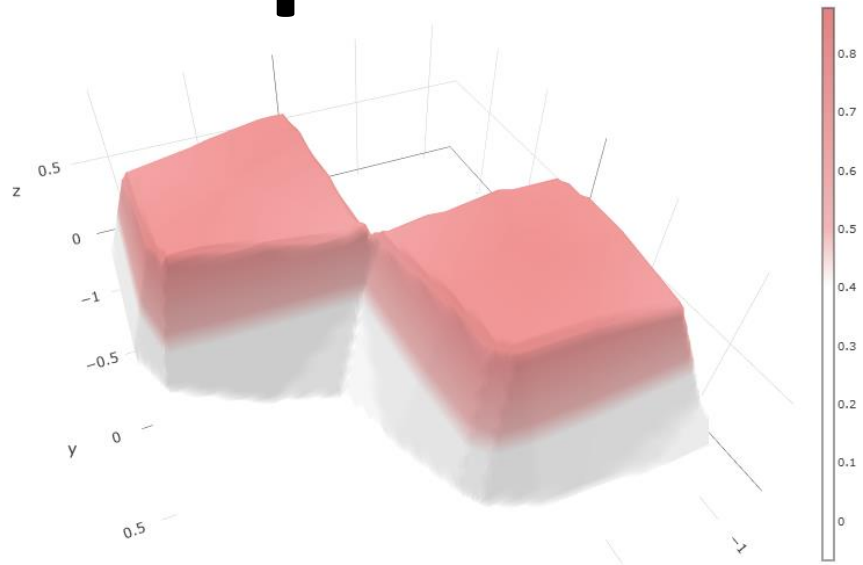
Synthetic data generation: example 1



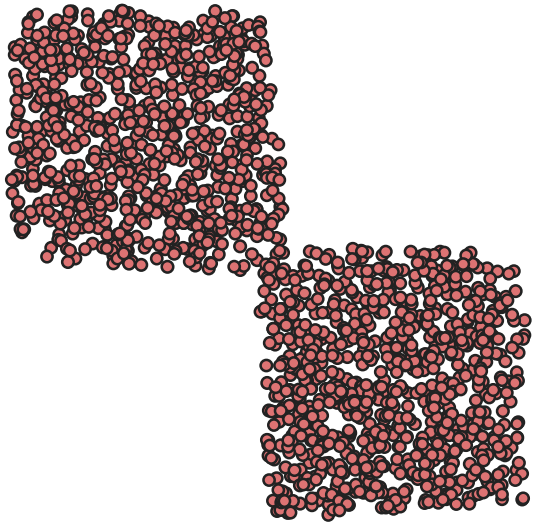
Original (train) dataset



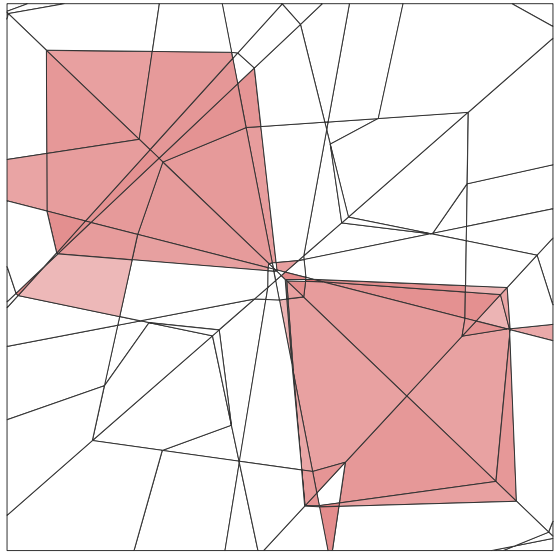
Output of $c_n^*(x)$



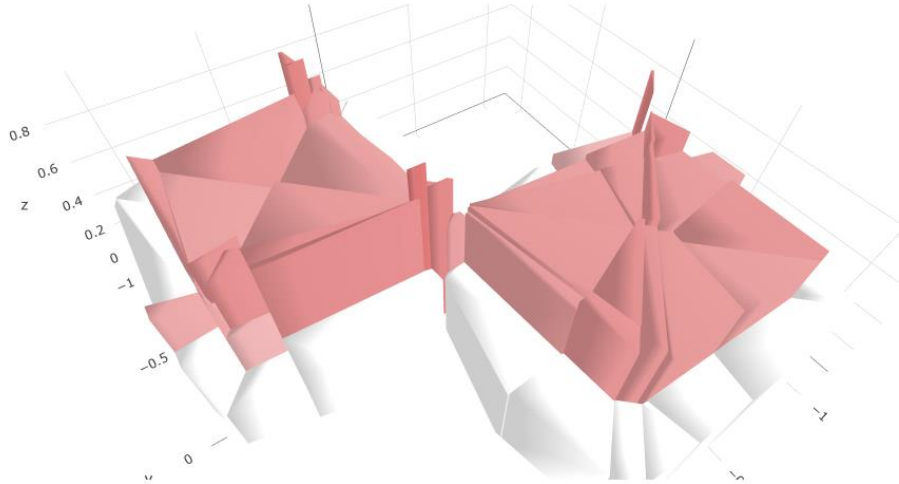
Output of $c_n^*(x)$ in 3D



Reproduced (synthetic) dataset

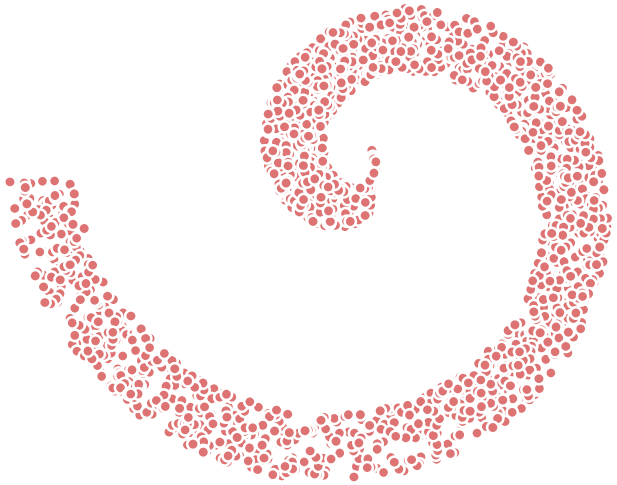


Output of $h_n^*(x)$

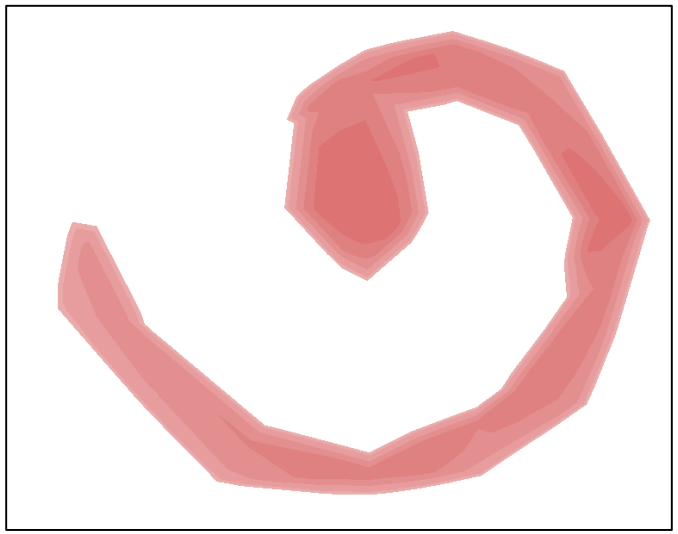


Output of $h_n^*(x)$ in 3D

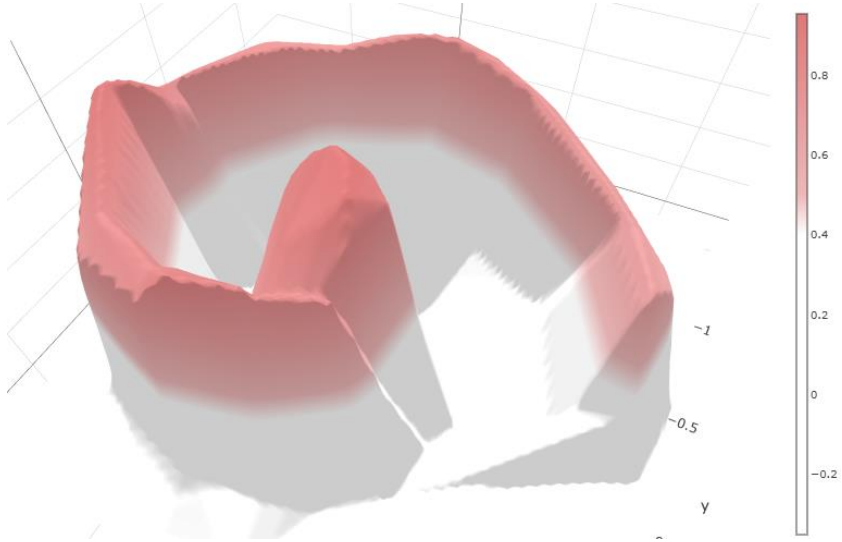
Synthetic data generation: example 2



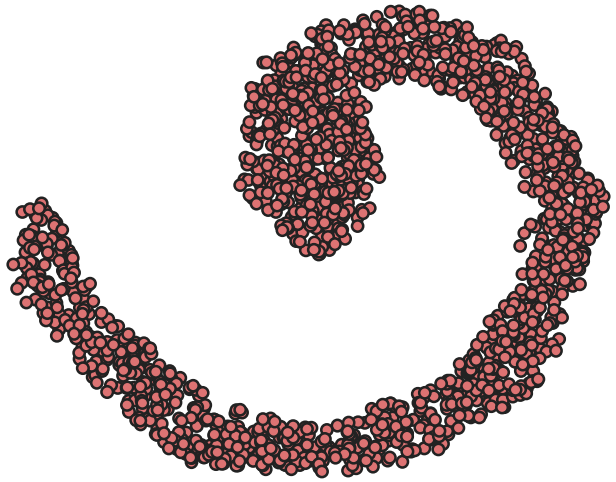
Original (train) dataset



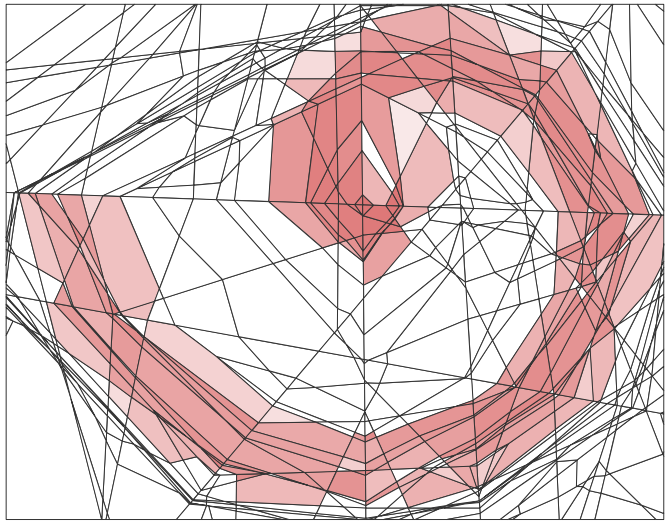
Output of $c_n^*(x)$



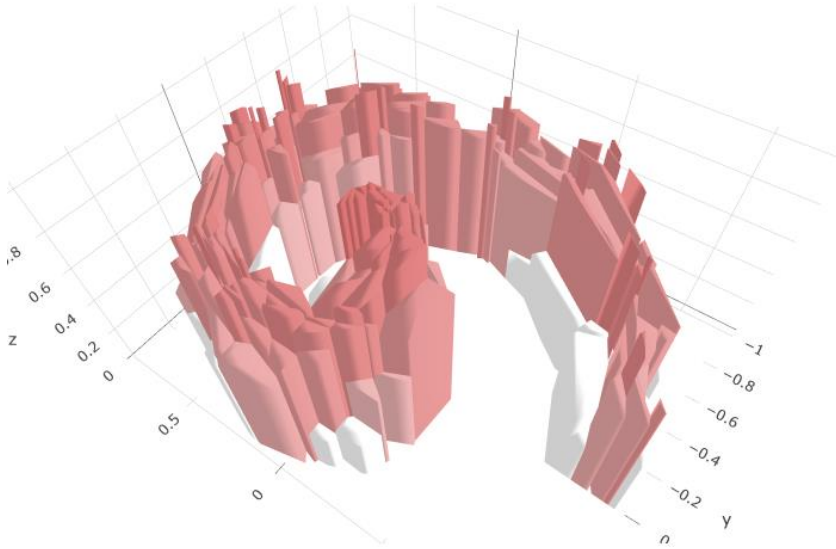
Output of $c_n^*(x)$ in 3D



Reproduced (synthetic) dataset



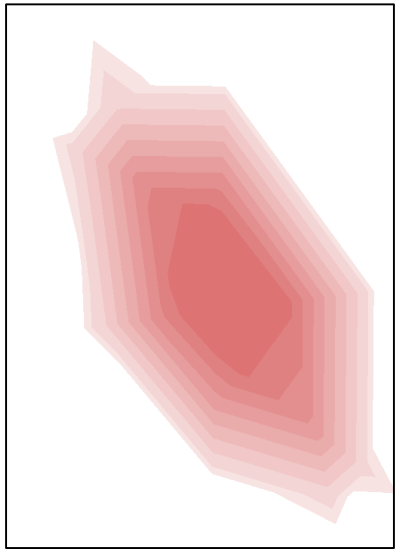
Output of $h_n^*(x)$



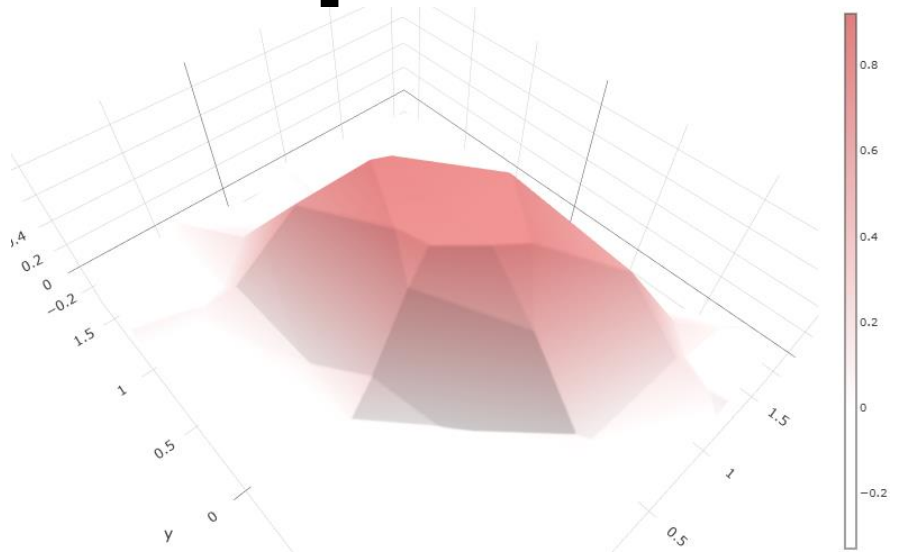
Output of $h_n^*(x)$ in 3D

Synthetic data generation: example 3

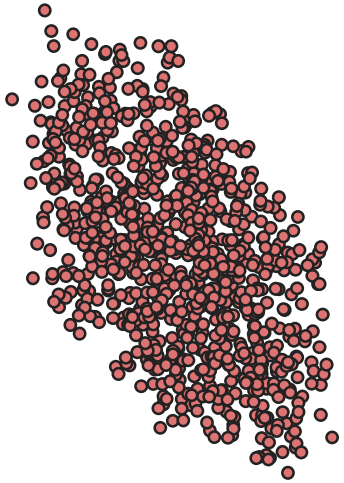
Original (train) dataset



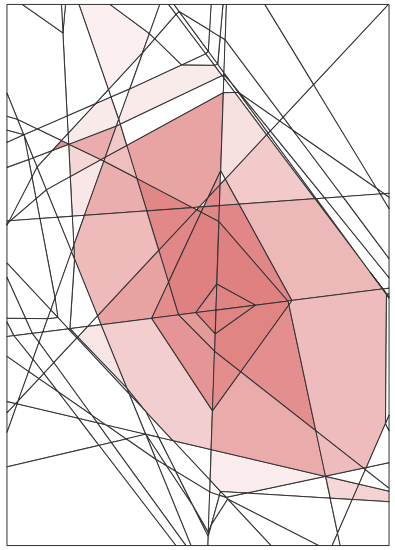
Output of $c_n^*(x)$



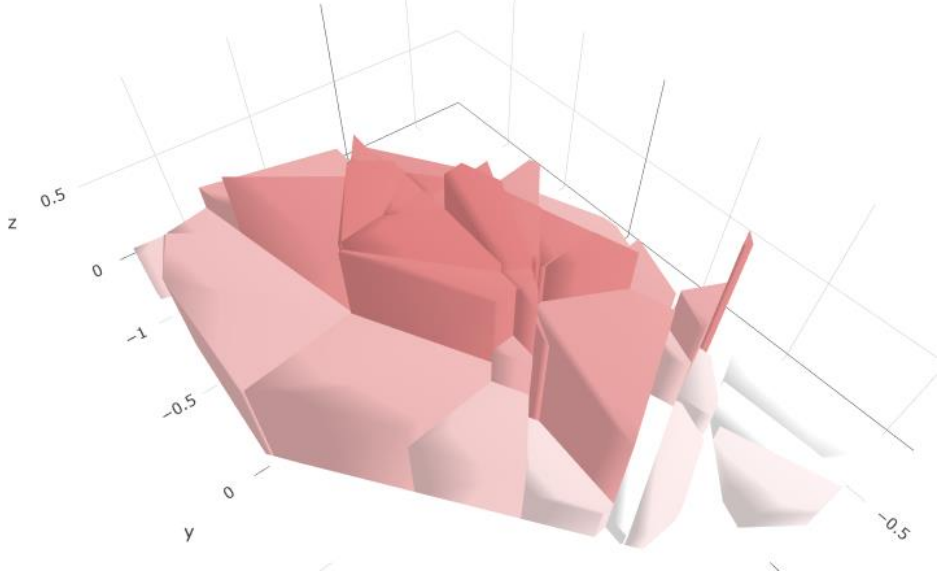
Output of $c_n^*(x)$ in 3D



Reproduced (synthetic) dataset



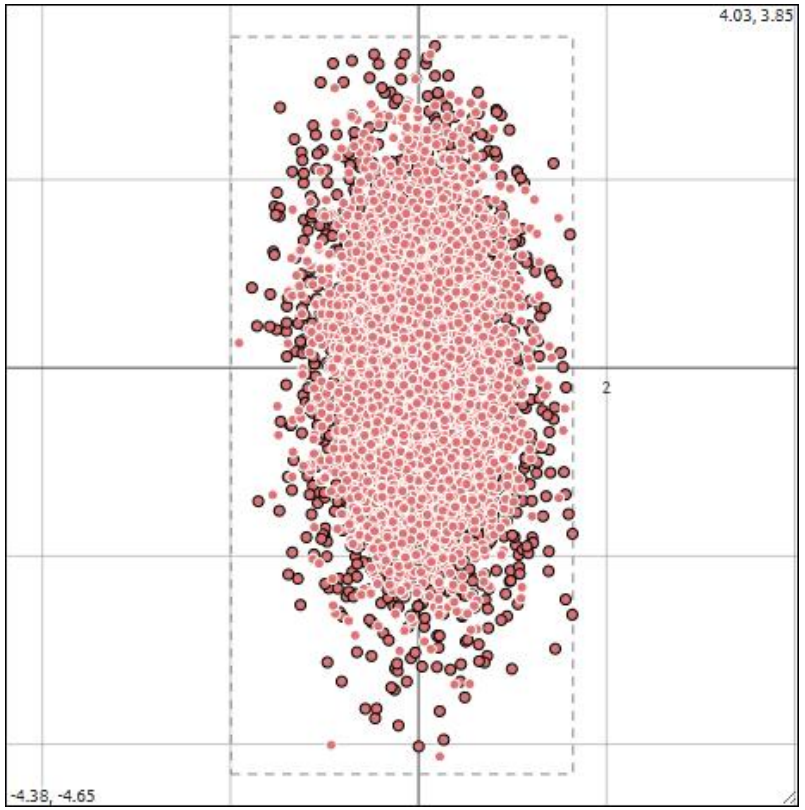
Output of $h_n^*(x)$



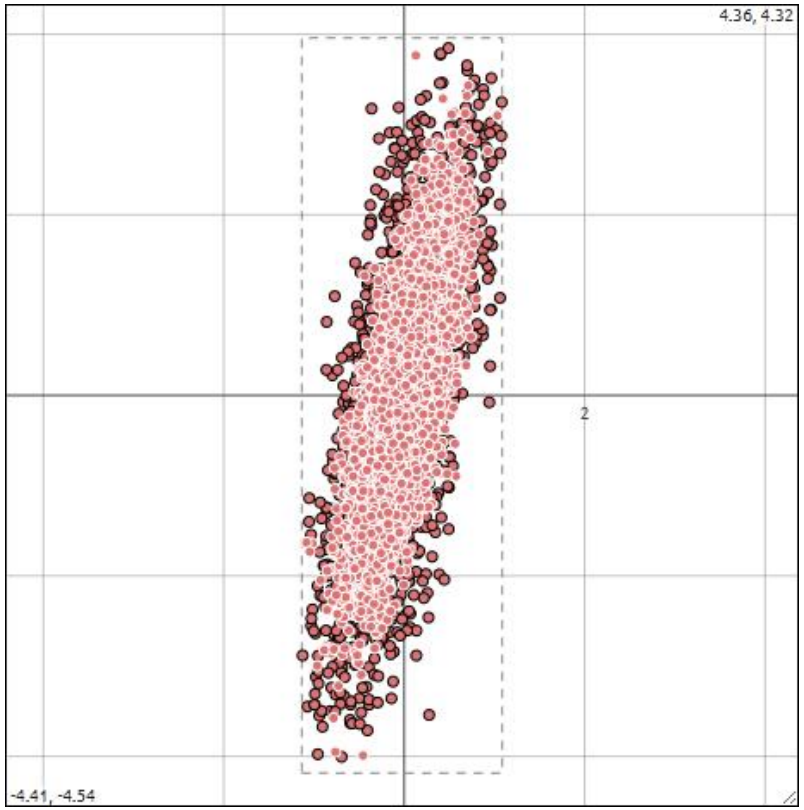
Output of $h_n^*(x)$ in 3D

Synthetic data generation: example 4

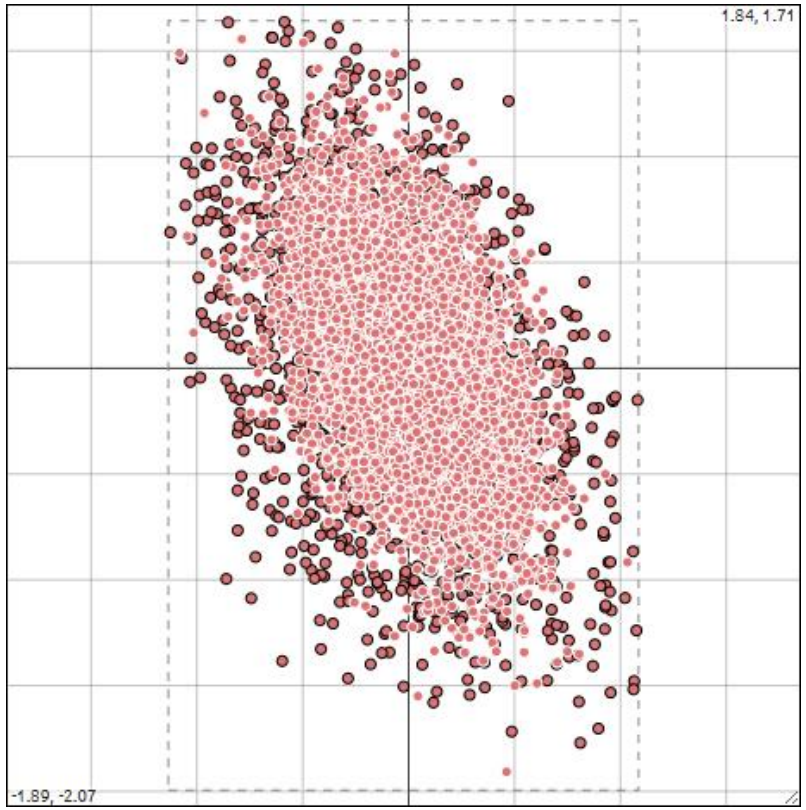
- **Real dataset X**: 10-dimensional Gaussian distribution.
- **Synthetic dataset $\widehat{X}$**: greater dispersion than in the two-dimensional case.



x_1/x_2 projection



x_1/x_3 projection

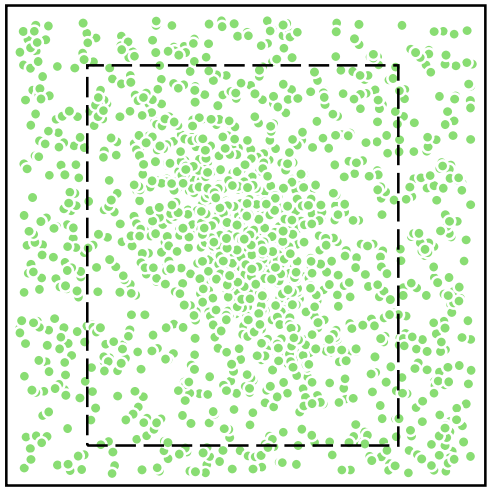


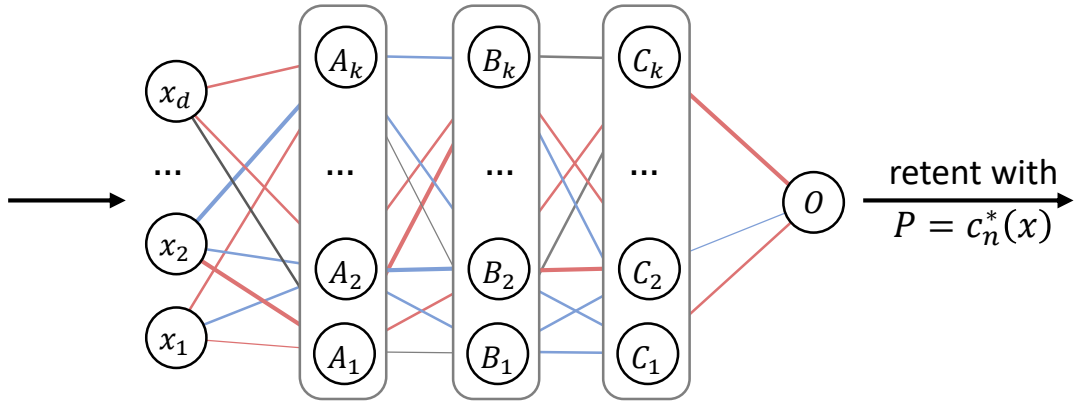
x_2/x_{10} projection

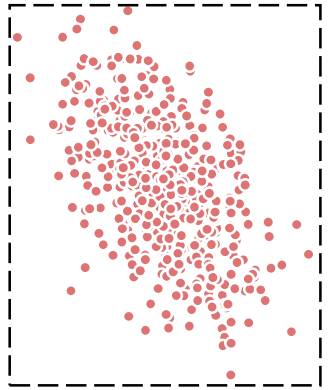
Future work

- **High-dimensional analysis:** studying method behavior in higher dimensions;
- **Training impact:** examining how perceptron training affects final quality;
- **Optimization strategies:** improving stability and efficiency of training;
- **Comparisons:** benchmarking against alternative density estimation methods.

Interactive perceptron visualizer







Conclusion

- **Structured approach**: generates high-fidelity synthetic data while preserving *statistical* and *structured* properties.
- **Adaptive shaping**: classifier output guides the synthetic distribution.
- **Perceptron flexibility**: different architectures capture complex density patterns.
- **Confidence threshold**: balances fidelity and sample sufficiency.

Interactive perceptron visualizer

Challenges

- High-dimensional background sampling remains difficult.
- Perceptron depth / width affects density granularity.
- Over-segmentation risk in complex models.