

Model Selection from Statistical Point of View

Stanislav Kolenikov
skolenik@unc.edu

October 24, 2002
Presentation at COMP 290 class

A Tale of Two Cities - I

Statistics and Computer Science: two separate worlds???

Presentation format:

Computer Science: multicolor PowerPoint slides

Statistics: L^AT_EX originated PDF-files

A Tale of Two Cities - II

Convergence:

Computer Science: a reasonable solution is found in a finite number of steps

Statistics: the variability decreases as the sample size increases to ∞ .

A Tale of Two Cities - III

Data:

Statistics: $10^1 - 10^5$

happy with variation of $O(\sqrt{N})$

continuous variables

Computer Science: $10^4 - 10^7$

operations $O(N \log N)$ is desirable

categorical variables

A Tale of Two Cities - IV

Aim of the Analysis:

Computer Science: find the solution given the hardware constraints
(memory, real time . . .)

Statistics: assess the generalization properties of the procedure and the
sampling variation of the answer

Outline of the Talk

1. Classification and prediction as functional approximation problems
2. Kernel and nearest neighbors methods
3. Linear methods and the maximum likelihood
4. Bridging the gap between the two
5. Model selection
6. Computational concerns

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, NY.

Functional approximation - I

Data:

- the inputs / predictors / independent variables X ;
- the outcome / response / explanatory variables Y .

Goal: to figure out what Y should be for a given constellation of X 's.

We are happy if our prediction is accurate, and we are :(if not.

Functional approximation - II

For the data Y and the prediction $f(X)$, define the *loss function*:

$$L(Y, f(X)) \geq 0, \quad L(\cdot, \cdot) = 0 \text{ iff } Y = f(X). \quad (1)$$

The aim then is to minimize the expected loss $E L(Y, f(X))$ where the expectation is over the stochastic elements in Y 's (and sometimes in X 's, too).

This is the framework of *statistical decision theory*.

Functional approximation - III

Classification: the domain of Y and $f(\cdot)$ is the set of class labels, $Y \in \{\mathcal{G}_1, \dots, \mathcal{G}_K\}$. The loss is given by a $K \times K$ matrix with the zero diagonale: the price of misclassification errors.

An often used specification is the *zero-one loss*:

$$L(Y, f(X)) = I[Y \neq f(X)]. \quad (2)$$

Another specification is the *maximum likelihood*, or *entropy*, loss:

$$L(Y, f(X)) = 2 \sum_{k=1}^K I(Y \in \mathcal{G}_k) \log \hat{p}_k(X) \quad (3)$$

where $\hat{p}_k(X)$ is the predicted probability that Y falls into the k -th class for given X .

Functional approximation - IV

Prediction of a continuous response: the domain of Y and $f(\cdot)$ is usually \mathbb{R} , or a subinterval of it.

Loss specifications:

quadratic, squared error, L_2 :

$$L(Y, f(X)) = (Y - f(X))^2 \quad (4)$$

absolute error, L_1 :

$$L(Y, f(X)) = |Y - f(X)| \quad (5)$$

Functional approximation - V

Let us look at the prediction error in more detail (equation (7.8) of HTF):

$$\begin{aligned}
 Y &= f(X) + \epsilon, \quad \mathbf{E} \epsilon = 0, \quad \text{Var} \epsilon = \sigma_\epsilon^2 \\
 \mathbf{E}[(Y - \hat{f}(x_0))^2 | X = x_0] &= \\
 &= \mathbf{E}[(f(x_0) - \mathbf{E}\hat{f}(x_0) + \mathbf{E}\hat{f}(x_0) - \hat{f}(x_0) + \epsilon)^2 | X = x_0] = \\
 &= \mathbf{E}(f(x_0) - \mathbf{E}\hat{f}(x_0))^2 + \mathbf{E}[\hat{f}(x_0) - \mathbf{E}\hat{f}(x_0)]^2 + \sigma_\epsilon^2 = \\
 &= \text{Bias}^2[\hat{f}(x_0)] + \text{Var}[\hat{f}(x_0)] + \text{Irreducible error} \quad (6)
 \end{aligned}$$

As we shall see later, by increasing the complexity of the model (as statisticians say, “devote more degrees of freedom to estimation”), one can get lower bias at the expense of higher variance. Is there a trade-off between the two? [Fig. 2.11 from HTF](#).

Nearest neighbors - I

An easy and obvious way to locally approximate a function at a given point is to look at the (response of the) neighbors.

In the prediction context,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \quad (7)$$

where $N_k(x)$ is the neighborhood of x that contains exactly k neighbors (k -nearest neighbors fit).

In the classification context, if there is a clear dominance of one of the classes in the neighborhood of an observation x , then it is likely that the observation itself would belong to that class, too. Thus the classification rule is the majority voting among the members of $N_k(x)$.

Nearest neighbors - II

Here is the example of classification according to the nearest neighbor approach.

Figures: [raw data](#); [15-nearest neighbor classifier](#), [1-nearest neighbor classifier](#) (*Voronoi tessellations*) — a clear overfit...

The boundary provided by the nearest neighbors methods has low bias, but a good deal of variance. Applying (6) to this method, we get

$$\mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] = \left[f(x_0) - \frac{1}{k} \sum_{x_i \in N_k(x_0)} f(x_i) \right]^2 + \sigma_\epsilon^2/k + \sigma_\epsilon^2 \quad (8)$$

Kernel methods - III

The generalization of the local approximation:

$$\hat{f}(x_0) = \frac{\sum_{i=1}^N K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^N K_\lambda(x_0, x_i)} \quad (9)$$

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right) \quad (10)$$

The function $h_\lambda(x_0)$ is the *bandwidth/window size* near x_0 . The function $D(\cdot)$ is referred to as the *kernel function*. It should peak near zero, have the support of $[-1, 1]$, and integrate to one. An example is *Epanechnikov kernel*:

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2), & |t| < 1 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

Kernel methods - IV

Equation (9) is known as *Nadaraya-Watson kernel weighted average*, and the expression in its denominator is *Rosenblatt-Parzen kernel density estimate*.

There are also variants of the kernel methods that assume local linearity rather than local constancy (local regression, *lowess* smoother — see [linear regression](#) to be discussed rather soon). That's something like first-order Taylor series expansion of the (unknown) $f(\cdot)$ that we want to approximate, as opposed to the zero-order approximation in (9).

Kernel methods - V

The nearest neighbors and other local / kernel methods do not rely on particular assumptions (except probably for the metric of the space). They might be adopted to pretty much any situation, and thus would have low bias although high variance.

However, they rather quickly start facing difficulties as the dimensionality of the problem grows (*curse of dimensionality*). The problem is that in high dimensions, there are few observations' that are really local to a given data point.

Fig. 2.6 of HTF: to capture $\nu\%$ of data for our local estimate, we need to stretch the neighborhood over $\approx \nu^{1/p}\%$ of the range in each of the p coordinates. That is a problem irrespective of the method! Trees and other KDD methods are bound to have problems of that kind, too.

Kernel methods - VI

- Highly adaptive: can achieve low bias, but then variance is high
- Need model performance evaluation and model selection
- # operations: computing N distances ($O(Np)$), sorting the observations by distance ($O(N \log N)$?) $\Rightarrow O(N^2 p \log N)$; or computing the kernel functions ($O(N)$) $\Rightarrow O(N^2 p)$
- Slow convergence (as bad as $O(N^{1/p})$),
- Dimensionality curse
- Need to have all of the data set in the memory

Linear models - I

The linear prediction of the output is based on the linear form of the relation to X :

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^p X_j \hat{\beta}_j = X^T \hat{\beta} \quad (12)$$

The approximation for the expected quadratic loss / training error is the residual sum of squares:

$$RSS(\beta) = \sum_{i=1}^N (y_i - x_i^T \beta)^2 \quad (13)$$

The value of β that minimizes the RSS is the *least squares* (or OLS) estimator of β . The explicit matrix formula is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (14)$$

Linear models - II

An example from HTF: 2D, two classes, linear boundary obtained by the OLS regression of the 0-1 class indicators. See [Fig. 2.1 from HTF](#).

What is the error that the linear model makes? (6) can be now re-written as:

$$\mathbb{E}[(Y - X^T \hat{\beta})^2 | X = x_0] = [f(x_0) - x_0^T \mathbb{E} \hat{\beta}]^2 + \|\mathbf{h}(x_0)\| \sigma_\epsilon^2 + \sigma_\epsilon^2 \quad (15)$$

where $\mathbf{h}(x_0) = x_0^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$. The in-sample / training error is found to be

$$\frac{1}{N} \sum_{i=1}^N [f(x_i) - x_i^T \mathbb{E} \hat{\beta}]^2 + \frac{p}{N} \sigma_\epsilon^2 + \sigma_\epsilon^2 \quad (16)$$

Linear models - III

Another loss function popular for the classification problems: [entropy](#).

For the zero-one case, it boils down to

$$\begin{aligned} L(Y, f(X)) &= Y \log \Pr(Y = 1|X) + (1 - Y) \log \Pr(Y = 0|X) \\ &= Y \log(\text{odds ratio}) + \text{const} \end{aligned} \quad (17)$$

If $\log(\text{odds ratio})$ is taken to be linear in X , then we result in *logistic regression*, closely related to *linear discrimination* (LDA). The boundary obtained by (17) is linear in X 's — the same as in Fig. 2.1.

Figures: [linear discrimination](#), [linear discrimination \$\pm 25\%\$](#) .

Linear models - IV

Properties of the linear models:

- low variance
- unknown / potentially high bias
- rigid assumptions about the data generating models
- need to have all of the data set in the memory
- computationally fast, though sometimes require iterative procedures
- if no model misspecification, there are some “natural” performance measures

Linear models - V

How to increase the complexity of a linear model? We can add higher order terms as the explanatory variables:

quadratic : x_1^2, x_2^2, x_1x_2 ;

cubic : $x_1^3, x_1^2x_2, x_1x_2^2, x_2^3$;

Figures: [discrimination with quadratic terms](#), [discrimination with quadratic terms \$\pm 25\%\$](#) ; [discrimination with cubic terms](#), [discrimination with cubic terms \$\pm 25\%\$](#) .

Other examples from Prof. Steve J. Marron: [parallel clouds](#); [tilted cross](#); [perpendicular clouds](#); [donut](#).

Or, we can use various other *basis expansion* models, such as splines, or neural networks.

(What is the impact of dimensionality?)

Linear models - VI

1D splines basis are the *B-splines*: the cubic splines, with non-zero weights near *knots*.

$$B(u) = \begin{cases} \frac{3|u|^3 - 6u^2 + 4}{6}, & -1 \leq u \leq 1, \\ \frac{(2 - |u|)^3}{6}, & 1 < |u| \leq 2, \\ 0, & 2 < |u|. \end{cases} \quad (18)$$

$$\phi_{temp}(t) = \alpha_0 + \sum_{k=1}^K \alpha_k \delta_k(t), \quad t \in [0, T], \quad \delta_k(t) = B\left(\frac{K}{T} \left(t - \frac{Tk}{K}\right)\right) \quad (19)$$

Figures: [discrimination with the product B-splines](#), [discrimination with the product B-splines \$\pm 25\%\$](#) . The grid of knots is $\{-1.4, 0.8, 3.0\} \times \{-1.1, 0.45, 2.0\}$

Linear models - VII

2D splines basis is the *thin-plate* splines. We again need the knots to center our basis function at. Those are denoted as $\mathbf{x}^{(j)}$ in the following formula:

$$f(\mathbf{x}) = \beta_{x_1} x_1 + \beta_{x_2} x_2 + \sum_{j=1}^J \beta_j \psi(x_1 - x_1^{(j)}, x_2 - x_2^{(j)})$$
$$\psi(x, y) = \frac{r \log r}{16\pi}; \quad r = \sqrt{x_1^2 + x_2^2} \quad (20)$$

Figures: [discrimination with thin-plates splines](#), [discrimination with thin-plate splines \$\pm 25\%\$](#) . The grid of knots is $\{-1.4, 0.8, 3.0\} \times \{-1.1, 0.45, 2.0\}$

Bayes classifier

If the process that generates the data is known (as is the case with the simulated data), then one can come up with the *Bayes classifier*, or *Bayes decision rule*, that suggest classifying an observation into the class with the highest probability:

$$\hat{Y}(X) = \arg \max_{k=1, \dots, K} \Pr[Y \in \mathcal{G}_k | X] \quad (21)$$

The error rate of the Bayes classifier is called the *Bayes rate*. It serves as the lower bound of the test error.

[Bayes decision boundary](#) in our simulated data example.

Model selection - I

As we have seen, for each prediction / classification method, there is a spectrum of models that differ in their complexity level α . Naturally, we would want to select the model that performs best — say in terms of the [expected loss](#), or some approximation to it:

$$\mathbb{E} L(Y, f(X)) \approx \frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} L(y_i - f(x_i)) \quad (22)$$

where \mathcal{S}_t is the sample used to assess the performance of the model.

If $\mathcal{S}_t = \mathcal{S}$, then we obtain the *training error* (denoted as $\overline{\text{err}}$). This is likely to be overly optimistic about the performance of the model, so ideally we would like to have extra data from the same population. If \mathcal{S}_t is such data, then we obtain the *test error*.

Simulated data example: [misclassification curves for various methods and parameter combinations](#).

Model selection - II

As a rule, the training error computed from the same data on which the learning was based is not a very good measure of the performance of the classification / prediction model for the new data. By how much is the misclassification rate understated? Suppose we can observe the new data Y^{new} at each of the training points $x_i, i = 1, \dots, N$, and compute the error in the new sample, Err_{in} . Then the excessive optimism about the misclassification rate can be expressed as

$$\text{op} = \text{Err}_{\text{in}} - \overline{\text{err}} = \frac{2}{N} \sum_{i=1}^N \text{Cov}(\hat{y}_i, y_i) \quad (23)$$

For the linear models, the last sum turns out to be $d(\alpha)\sigma_\epsilon^2$ where $d(\alpha)$ is the number of parameters (inputs in linear regression, # basis expansion terms, etc.).

Model selection - III

A popular model selection measure is *Akaike information criterion*, AIC, and what it does is the correction to the appropriate error rate for linear models

$$\text{AIC} = \overline{\text{err}} + \frac{2d(\alpha)}{N} \sigma_\epsilon^2 \propto -2 \log LKHD + \frac{2d(\alpha)}{N} \quad (24)$$

In fact, the number of parameters for the linear models is a special case of the more general definition:

$$d(\alpha) = \text{tr} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}} \quad (25)$$

Another popular measure is *Bayesian information criterion* (BIC):

$$\text{BIC} = -2 \log LKHD + d(\alpha) \log N \sigma_\epsilon^2 \quad (26)$$

Model selection - IV

In practice, we do not know the distributions involved. But what we can do is to base our analysis on the empirical distributions in the spirit of (22). In terms of that formula, one can split the data into the training part $\mathcal{S} \setminus \mathcal{S}_t$ and the test part \mathcal{S}_t . This is the essence of *cross-validation*.

Various options for cross-validation:

- split the original data into the training subsample, cross-validation subsample, and sometimes test subsample to assess the performance of the model selected by the cross-validation (say 2:1:1);
- split all the data into chunks of 10%; put aside one chunk, train the model(s), assess the performance with the 10% cross-validation subsample; repeat 10 times for all subsamples; add the results up; choose the best performing model.
- *leave-one-out/jackknife* method: put aside one observation at a time.

Cross-validation - V

Yet... not without a fault.

Cross-validation estimates the error rate *for a wrong sample size*. In difficult learning situations, leaving 10% of data aside may lead to noticeable biases in the results. The cross-validation estimate of the prediction error is biased upward; *per se*, CV tends to undersmooth the data.

Hypothetical example: [Fig. 7.8 of HTF](#).

Simulated data: [CV estimate of the prediction error](#).

Cross-validation - VI

An *ad hoc* fix for the undersmoothing problem: find the most parsimonious model that has the (CV-estimated) misclassification rate at most 1 standard error above the minimal one.

Once again: [CV estimate of the prediction error](#).

It looks like the model with $k = 27$ gives the best results (d.f. = 7.4).

The resulting classification maps: [5 nearest neighbor classification](#), [5 nearest neighbor classification \$\pm 25\%\$](#) . [15 nearest neighbor classification](#), [15 nearest neighbor classification \$\pm 25\%\$](#) . [27 nearest neighbor classification](#), [27 nearest neighbor classification \$\pm 25\%\$](#) .

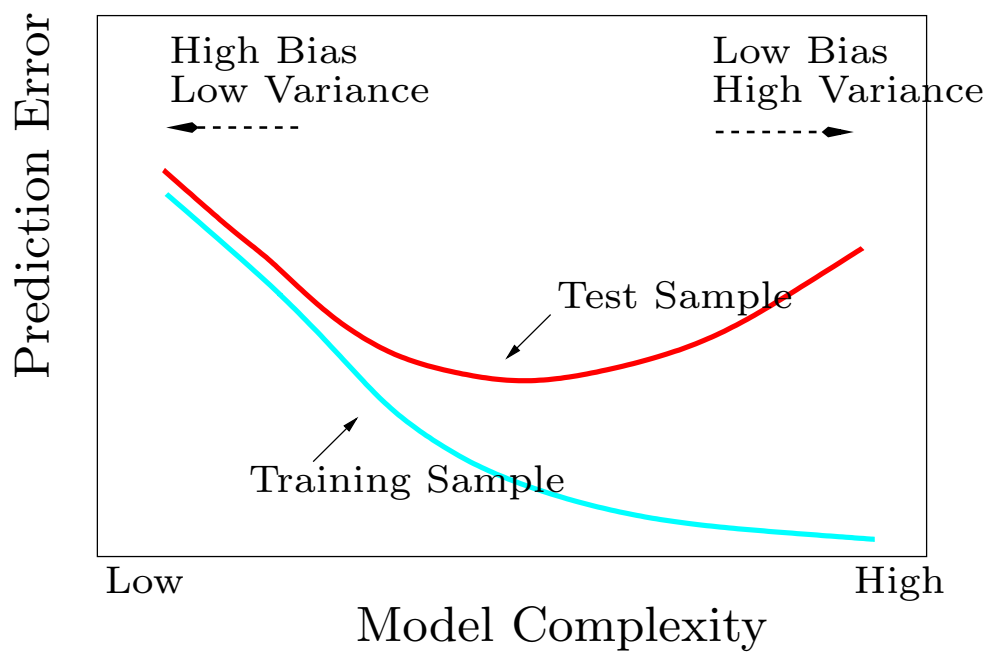
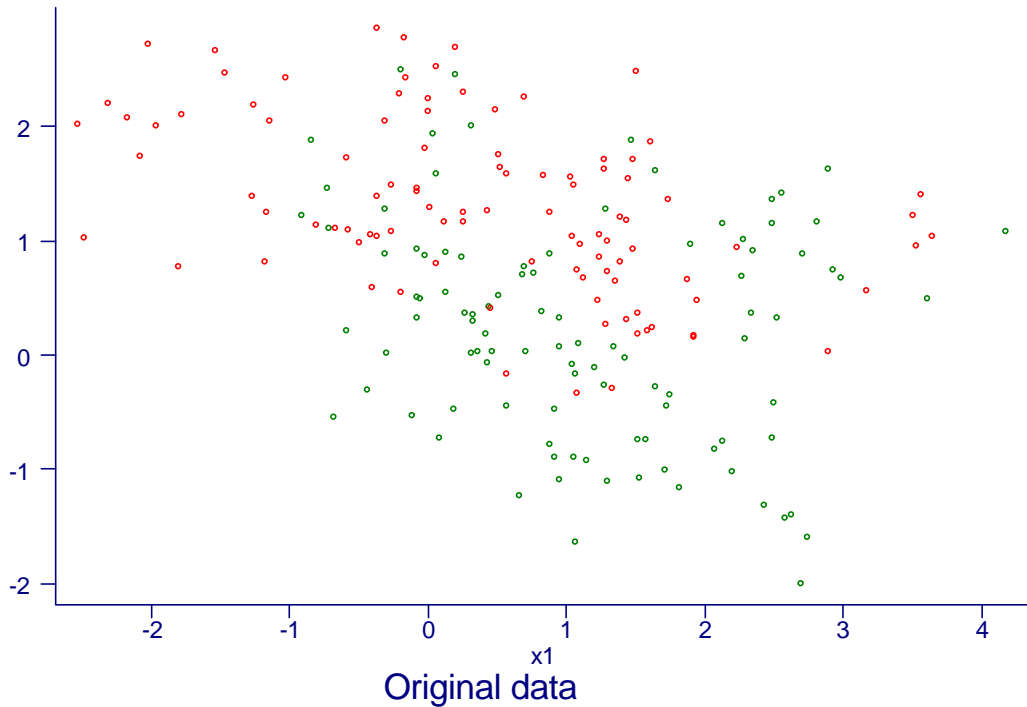


Figure 2.11: *Test and training error as a function of model complexity.*

Original data: class 0

Original data: class 1



15-Nearest Neighbor Classifier

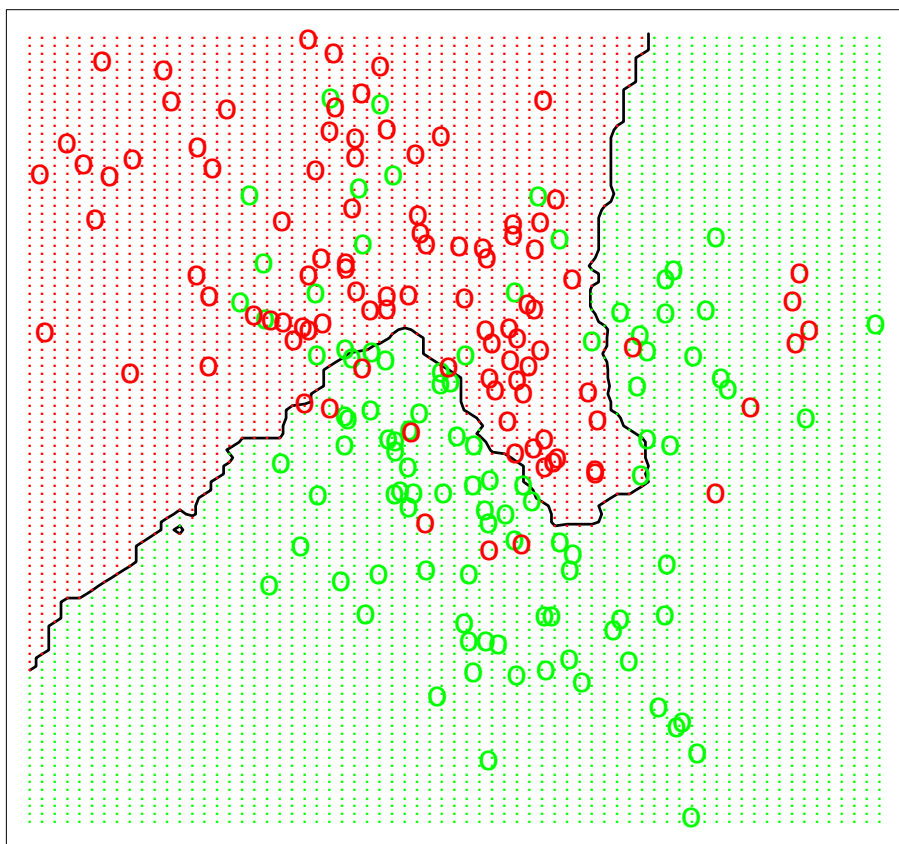


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

1-Nearest Neighbor Classifier

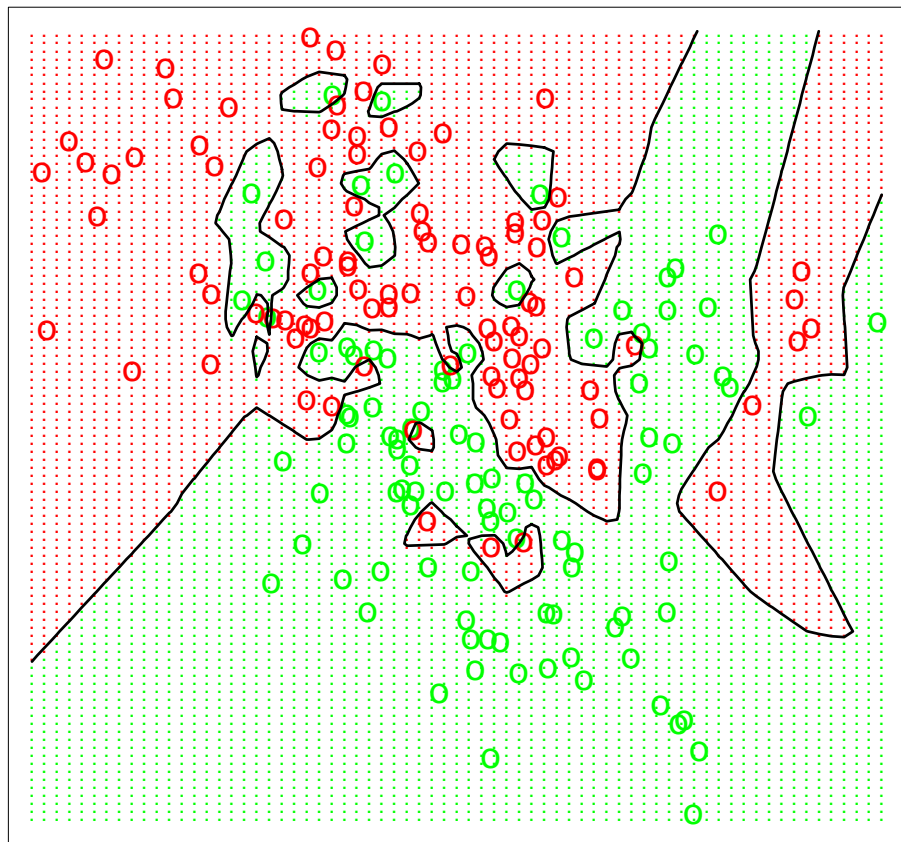


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1), and then predicted by 1-nearest-neighbor classification.*

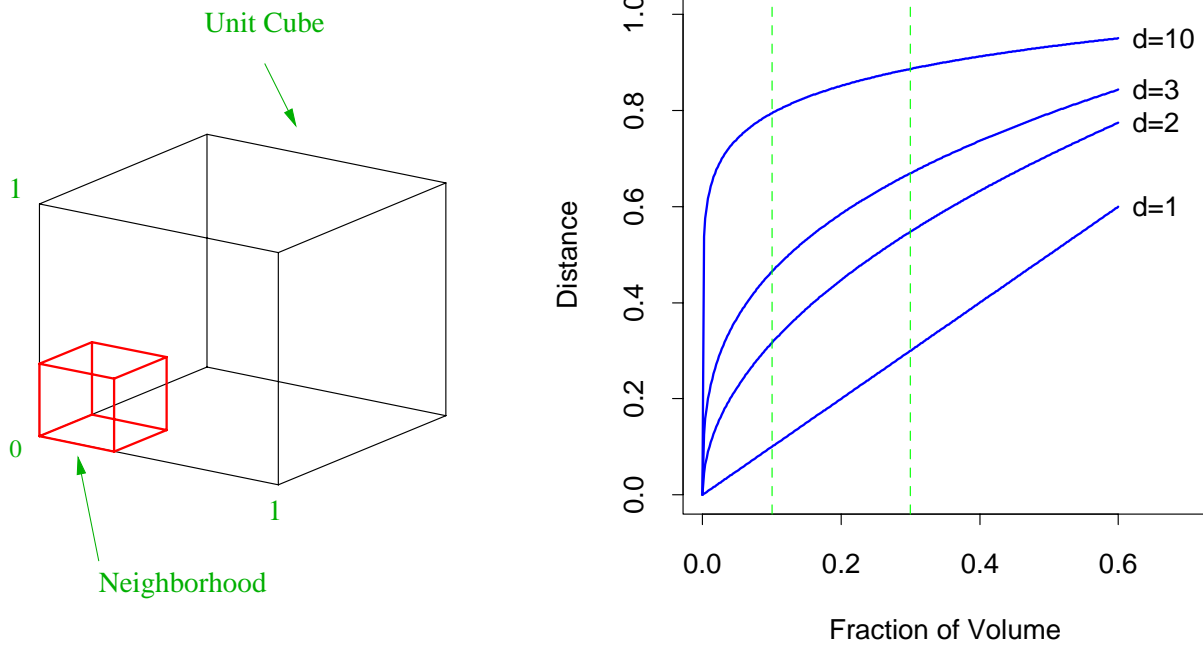


Figure 2.6: *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p . In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

Linear Regression of 0/1 Response

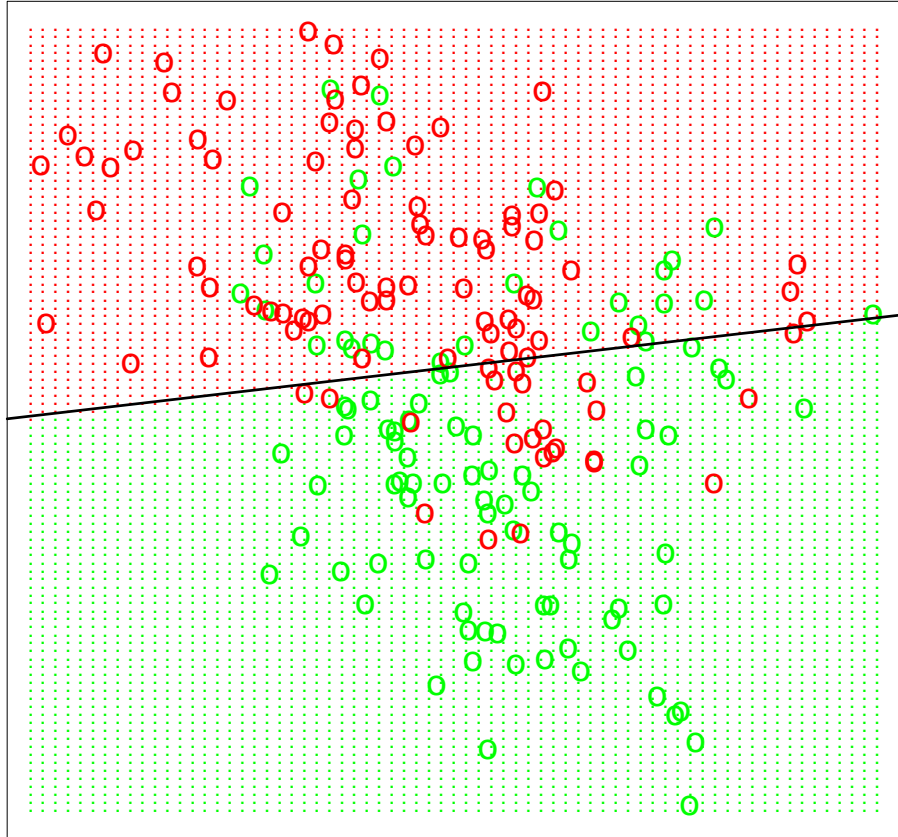
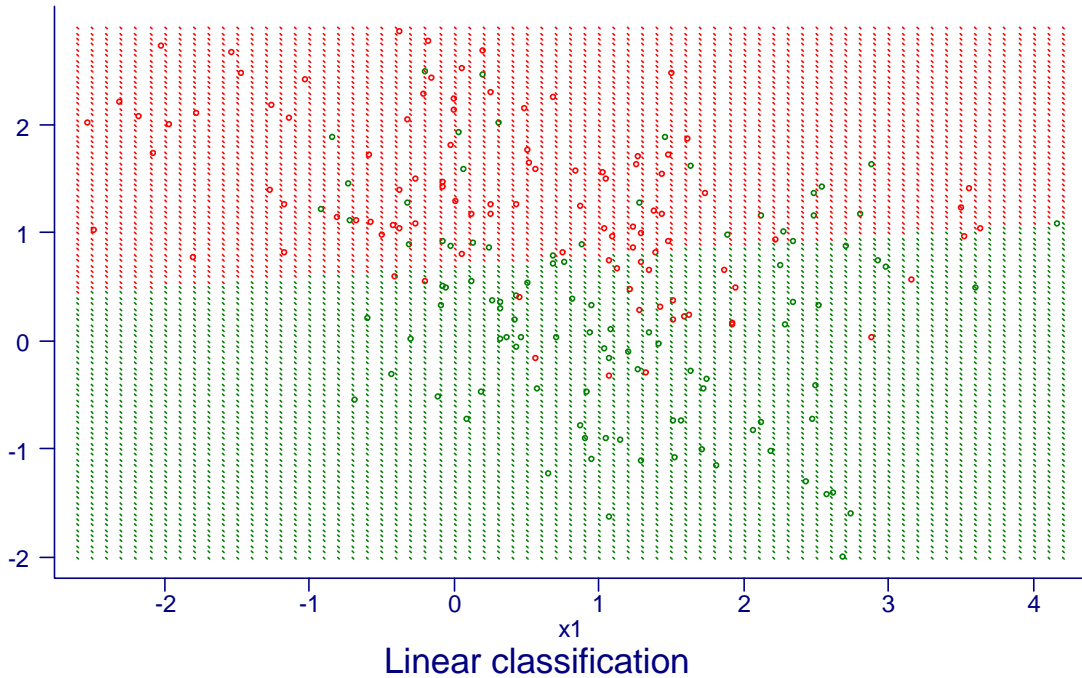


Figure 2.1: *A classification example in two dimensions. The classes are coded as a binary variable—**GREEN** = 0, **RED** = 1—and then fit by linear regression. The line is the decision boundary defined by $x^T \hat{\beta} = 0.5$. The red shaded region denotes that part of input space classified as **RED**, while the green region is classified as **GREEN**.*

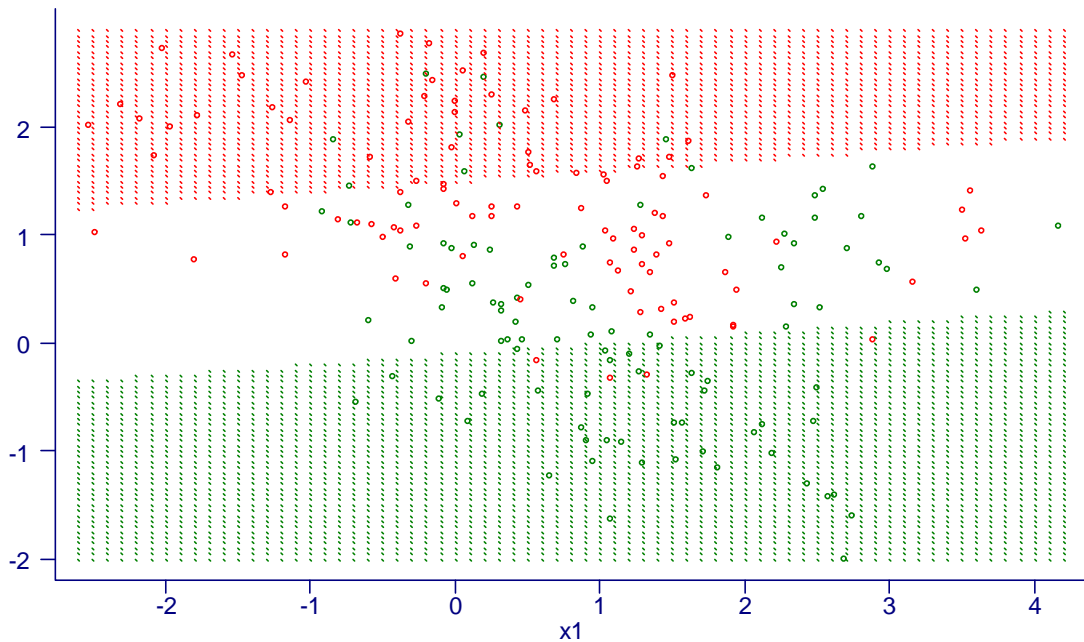
○ Original data: class 0
● Linear prediction: class 0

○ Original data: class 1
● Linear prediction: class 1



○ Original data: class 0
· Linear prediction: class 0

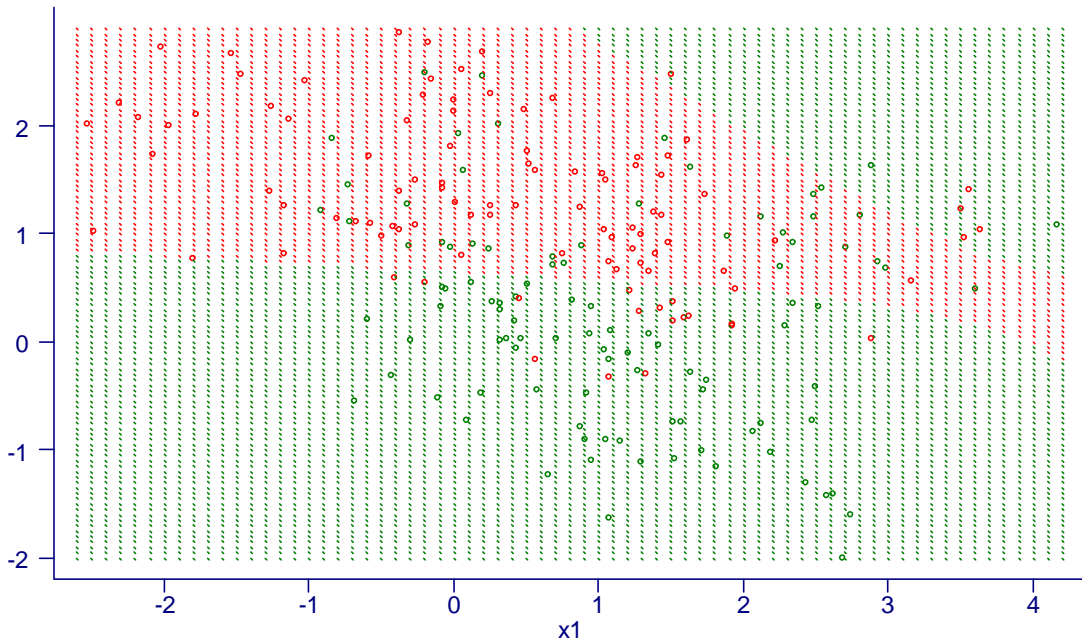
○ Original data: class 1
· Linear prediction: class 1



Linear classification: softer boundary

○ Original data: class 0
○ Quadratic prediction: class 0

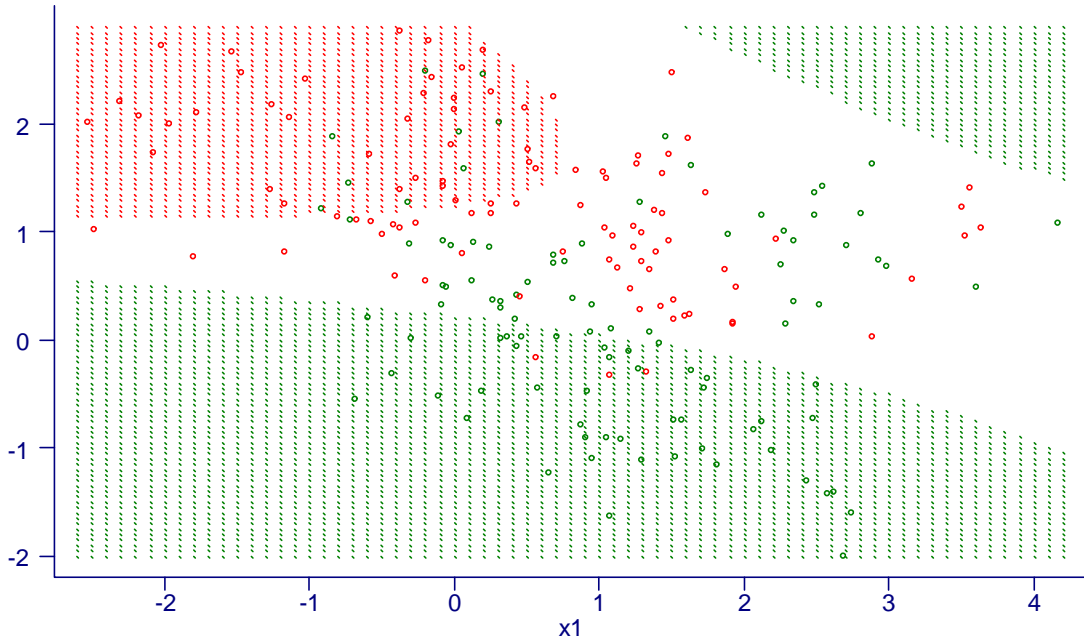
○ Original data: class 1
○ Quadratic prediction: class 1



Classification with quadratic terms

○ Original data: class 0
○ Quadratic prediction: class 0

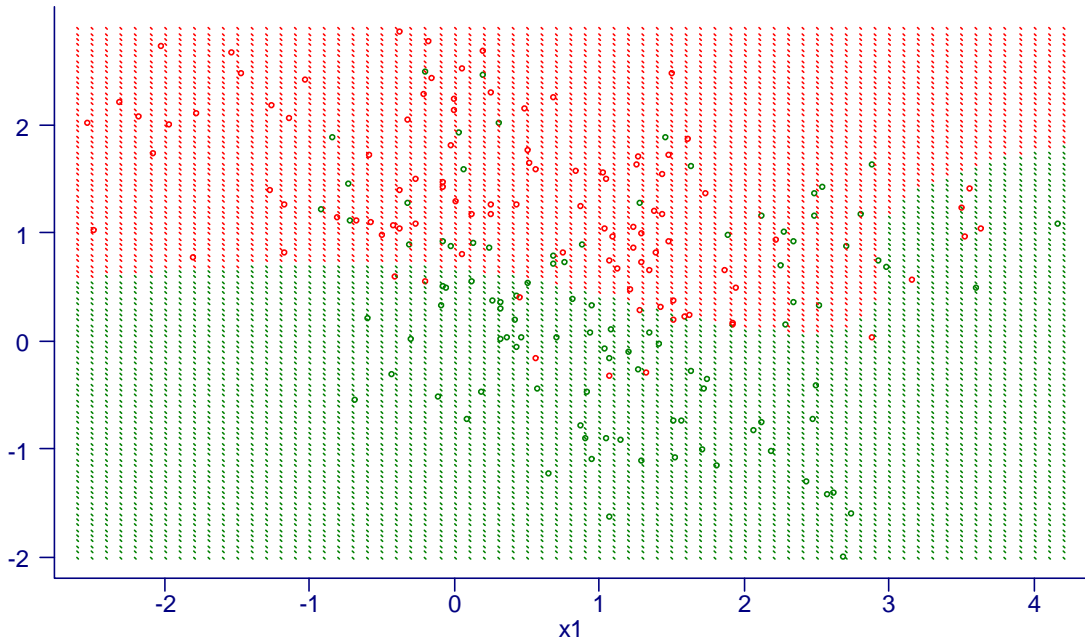
○ Original data: class 1
○ Quadratic prediction: class 1



Softer classification with quadratic terms

○ Original data: class 0
○ Cubic prediction: class 0

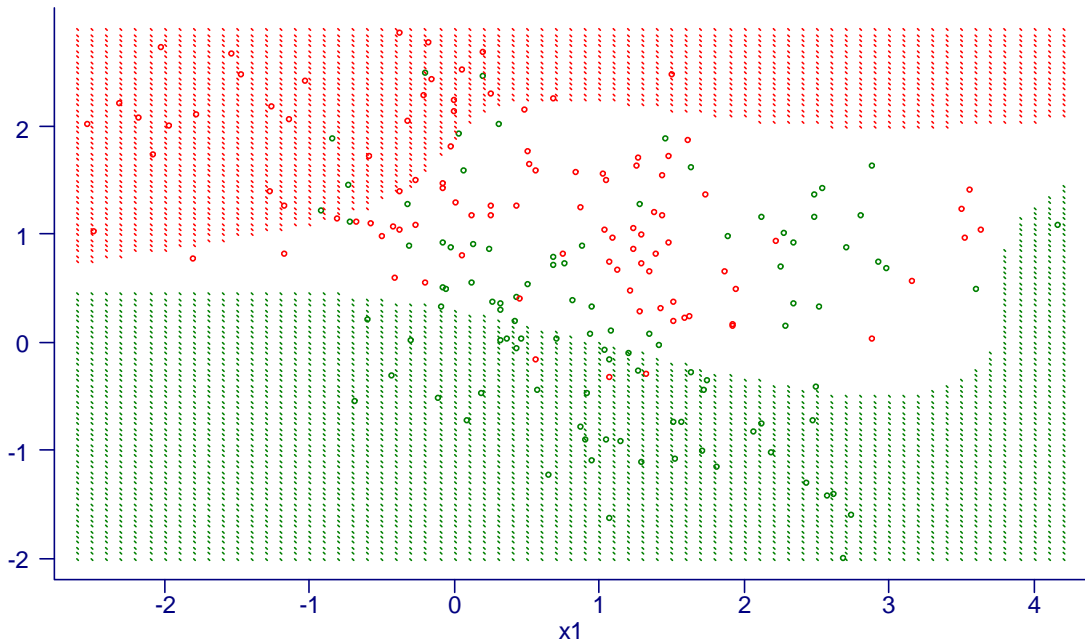
○ Original data: class 1
○ Cubic prediction: class 1



Classification with cubic terms

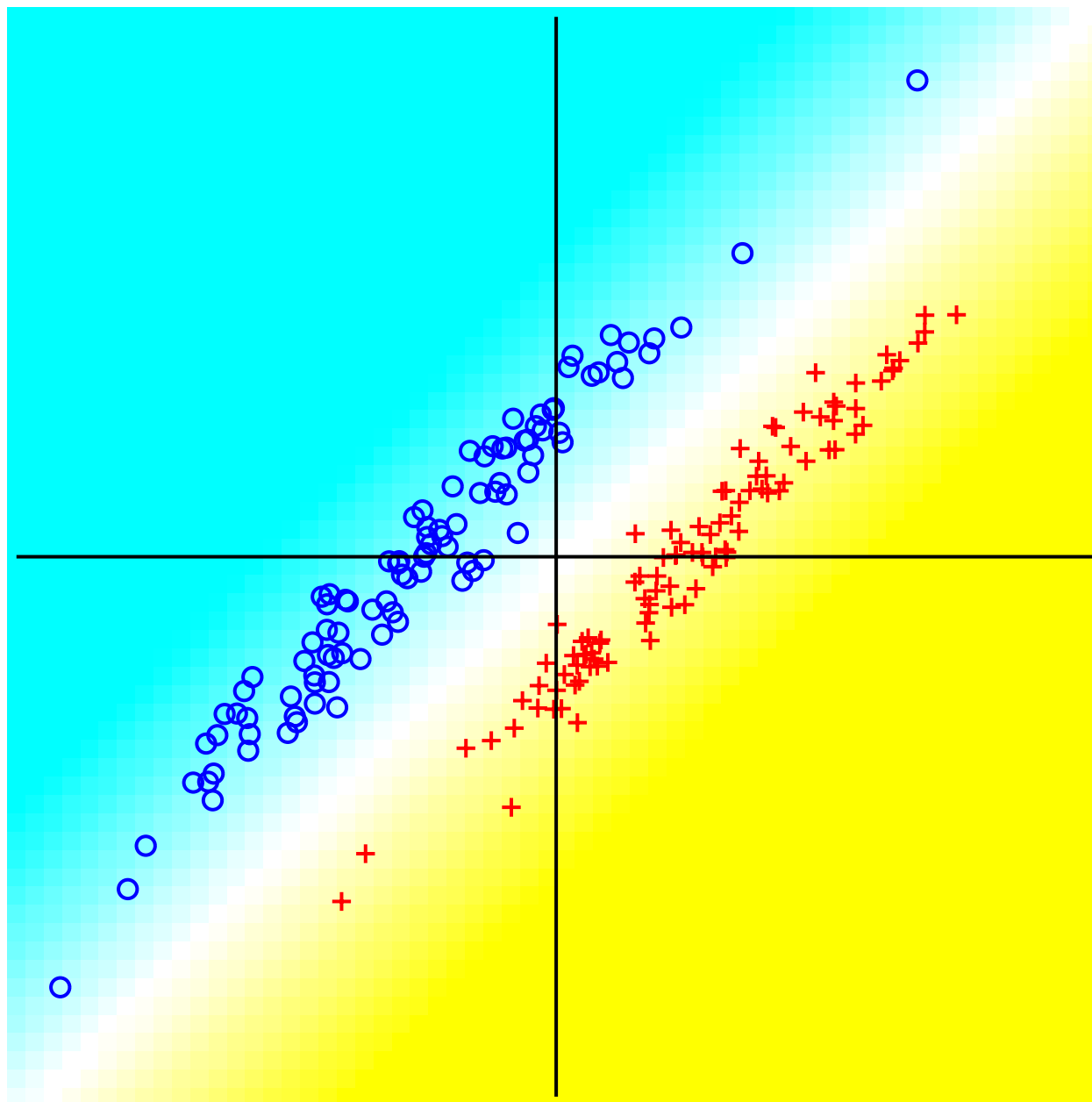
○ Original data: class 0
○ Cubic prediction: class 0

○ Original data: class 1
○ Cubic prediction: class 1

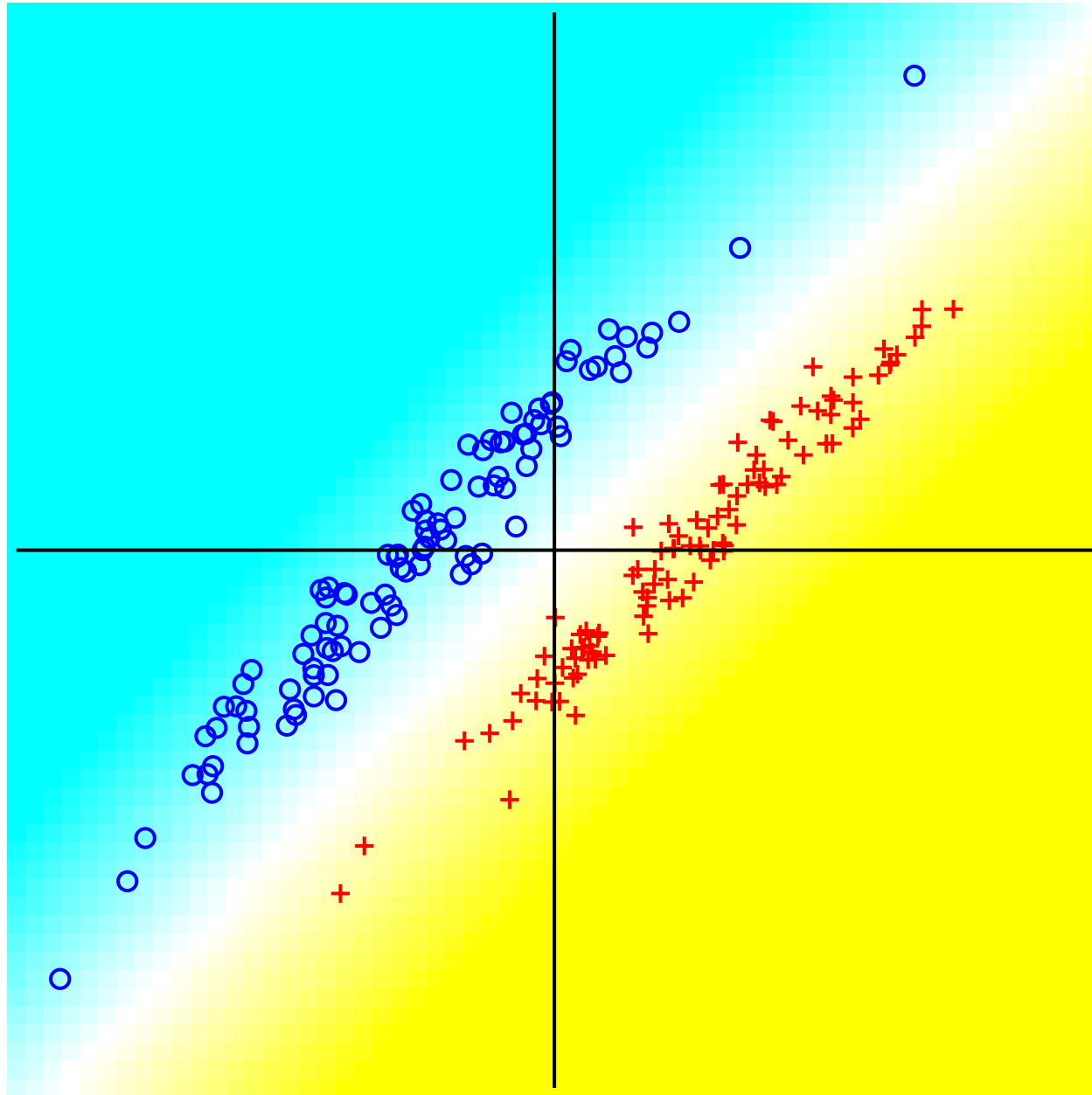


Classification with cubic terms

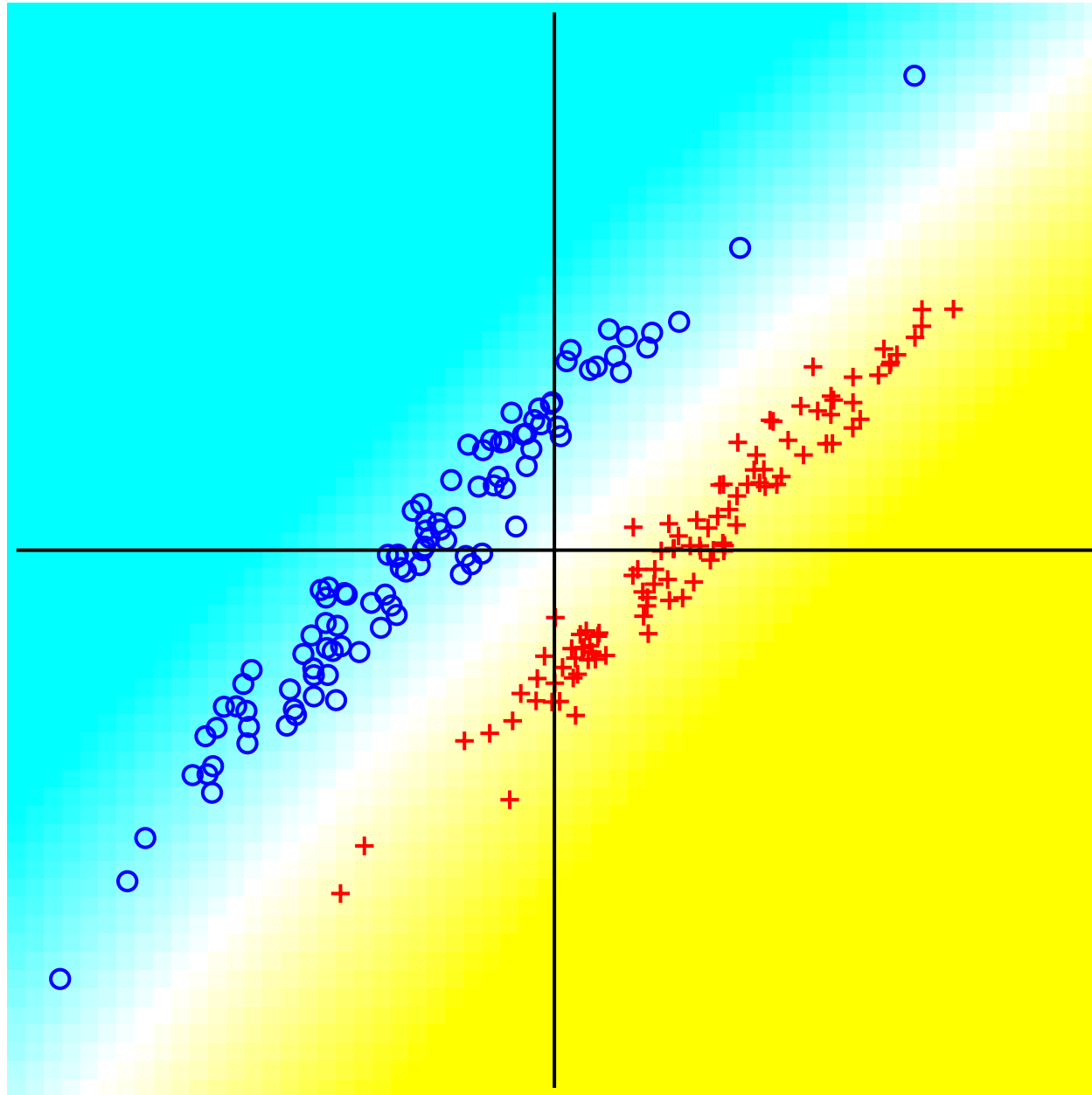
Toy Data 1, Disc: FLD, Embed: x_1, x_2 only



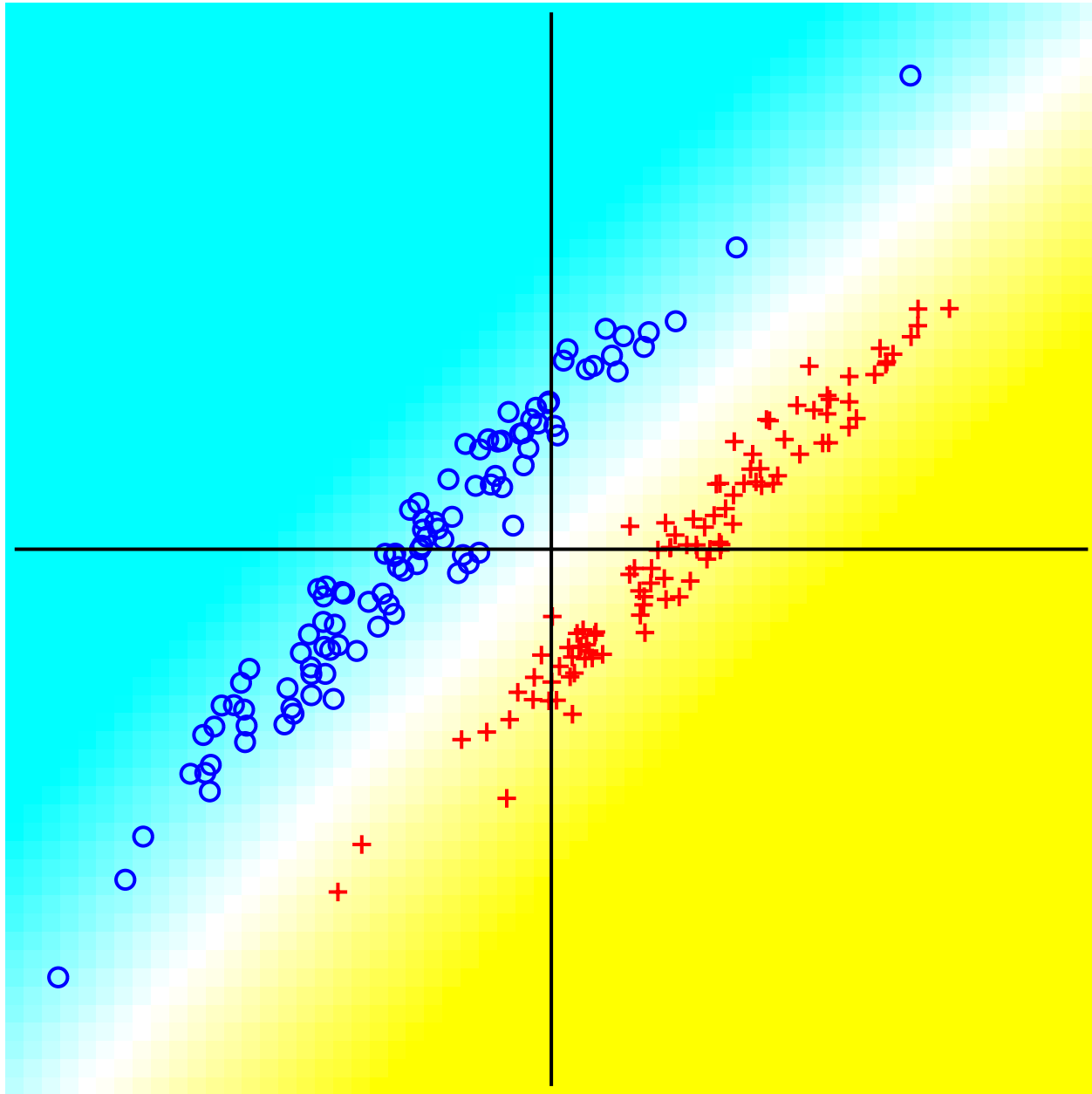
Toy Data 1, Disc: FLD, Embed: x_1, x_2, x_1^2



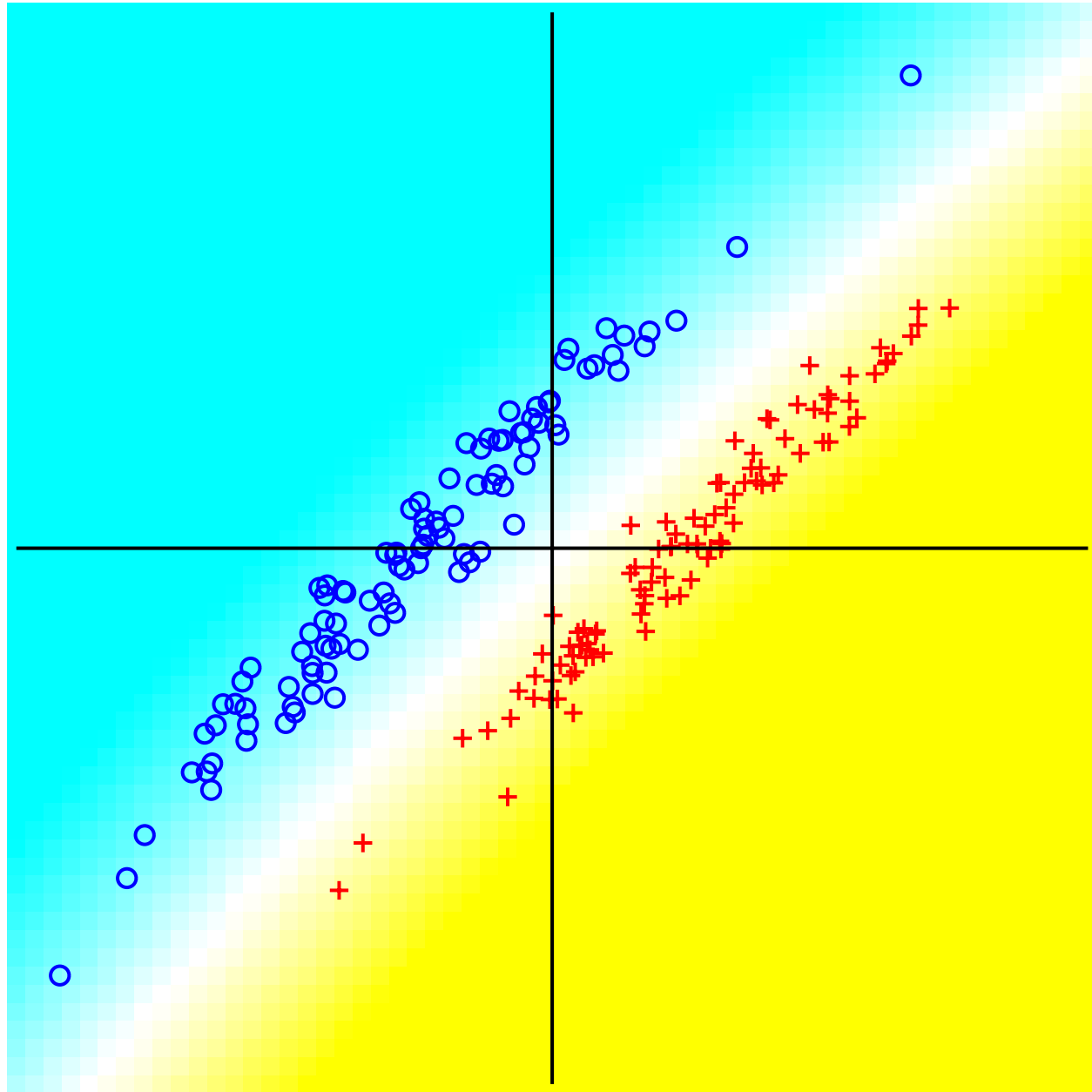
Toy Data 1, Disc: FLD, Embed: x_1, x_2, x_2^2



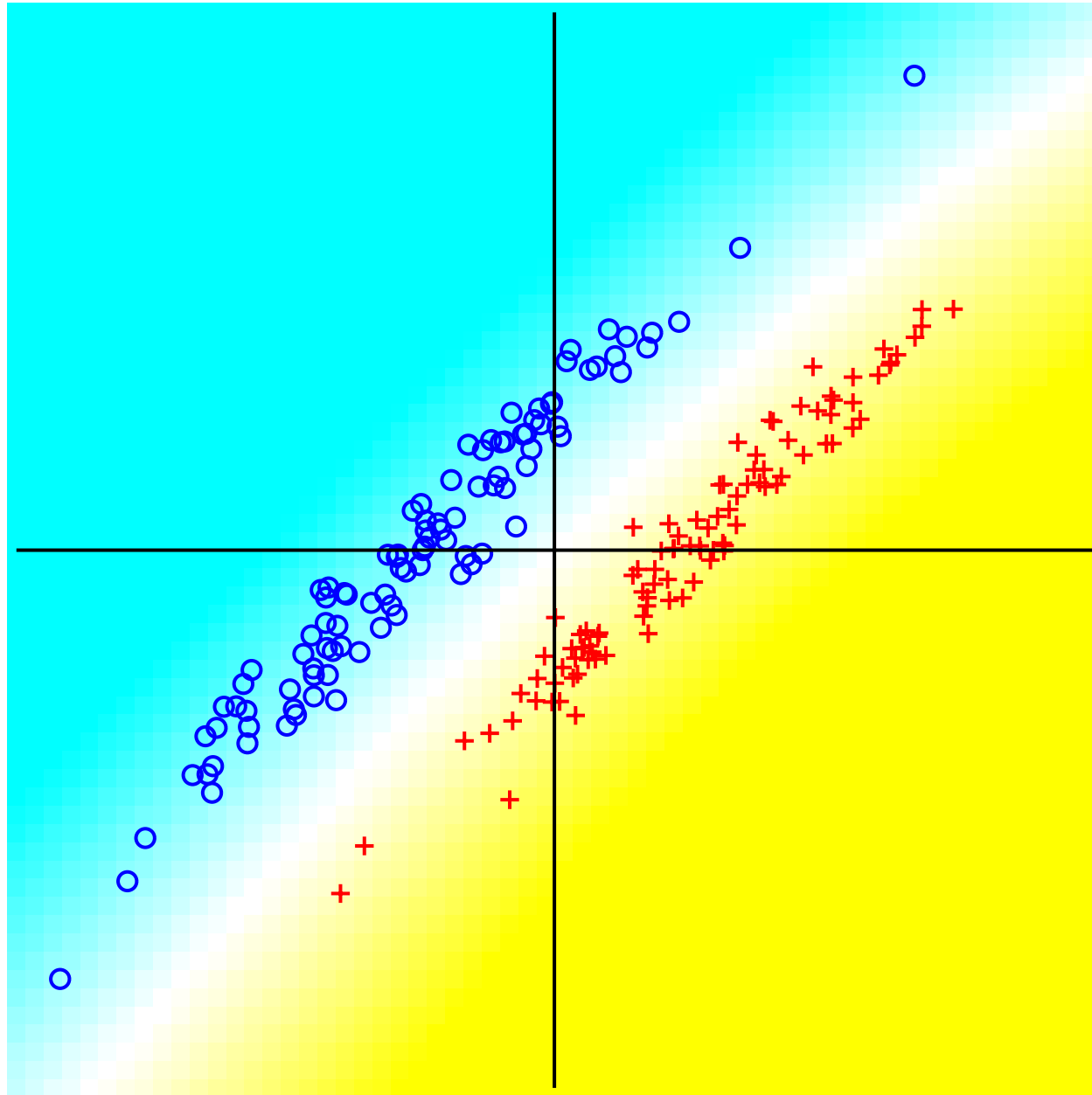
Toy Data 1, Disc: FLD, Embed: x_1, x_2, x_1^2, x_2^2



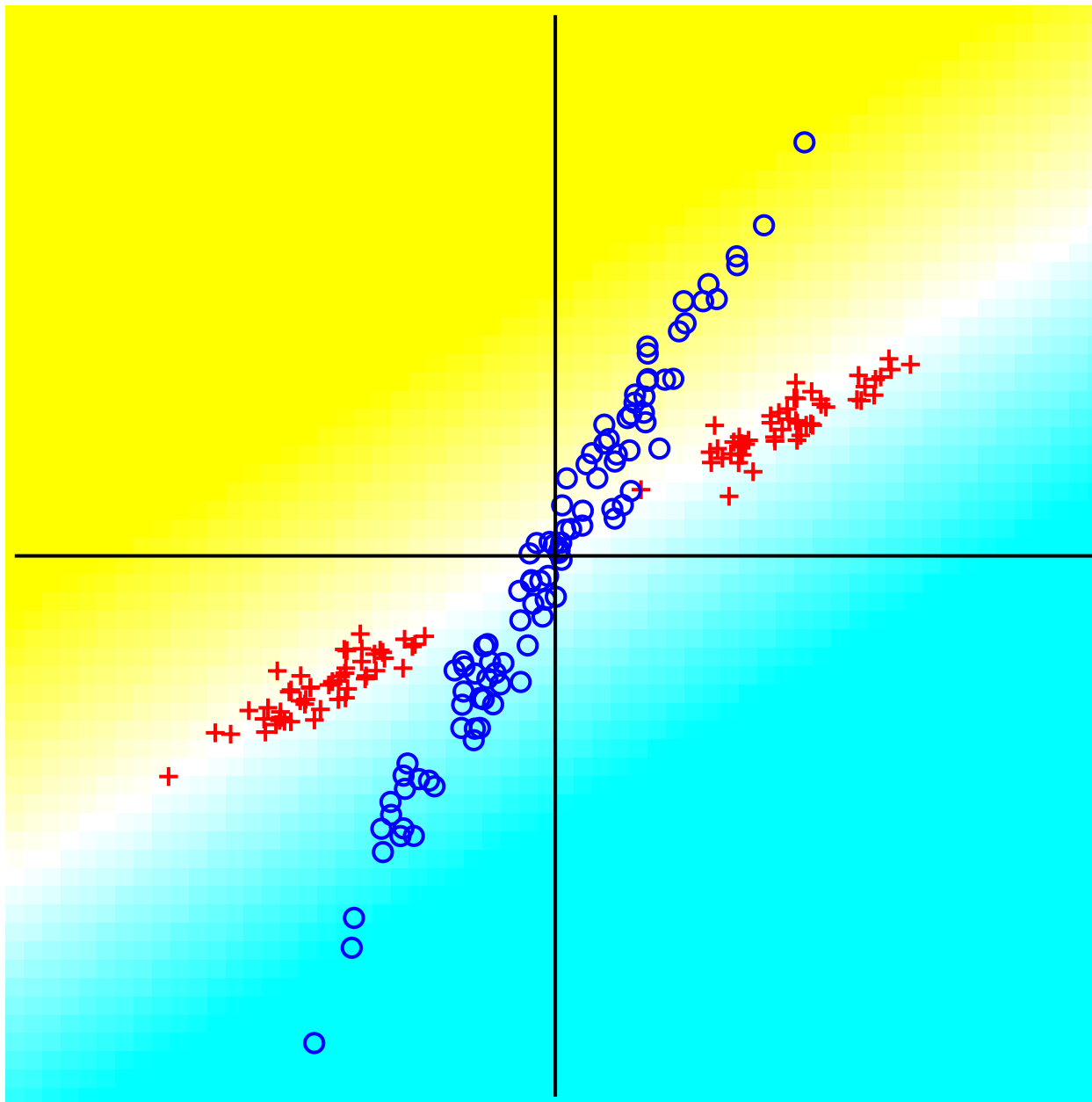
Toy Data 1, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1x_2$



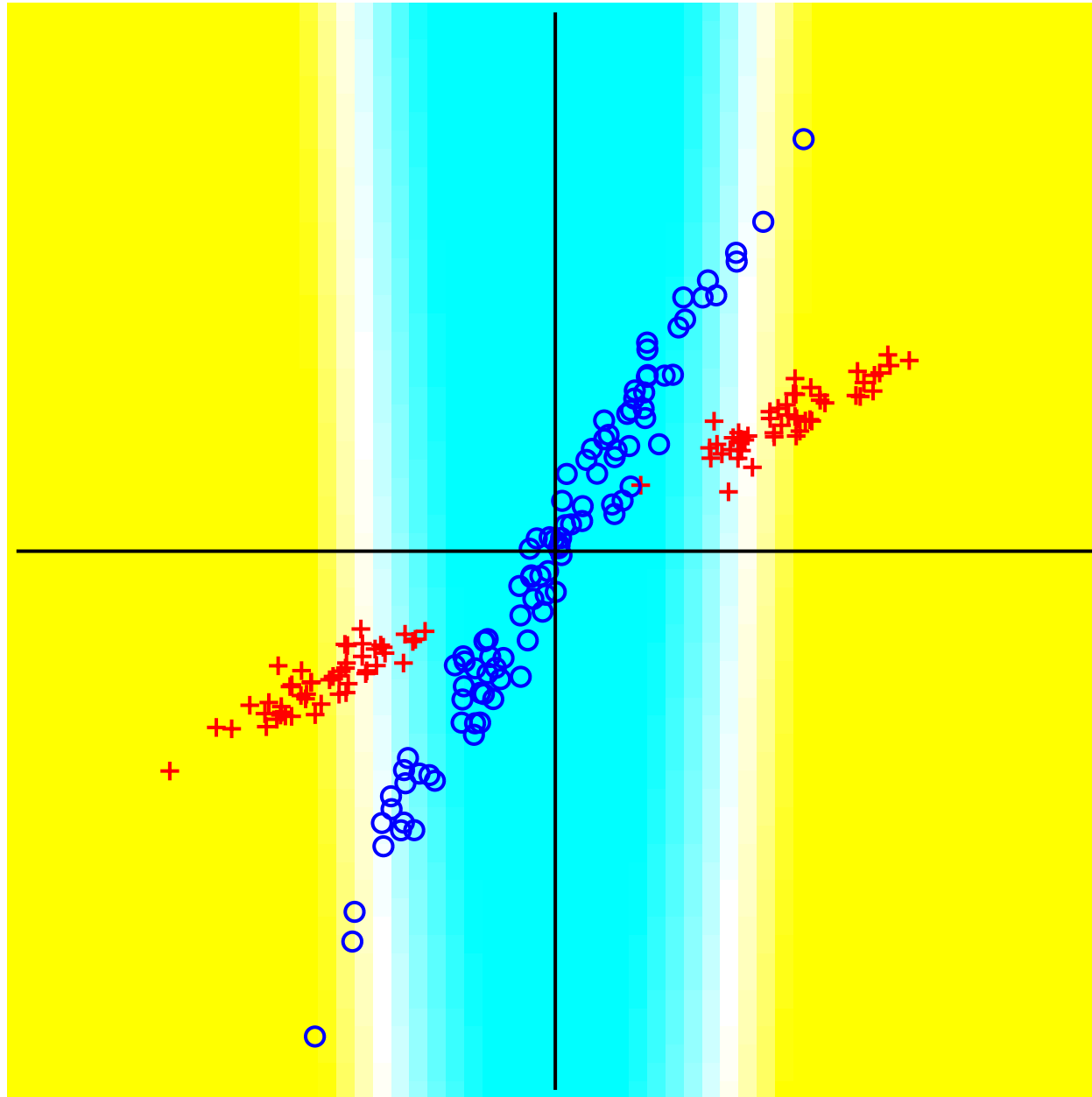
Toy Data 1, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3$



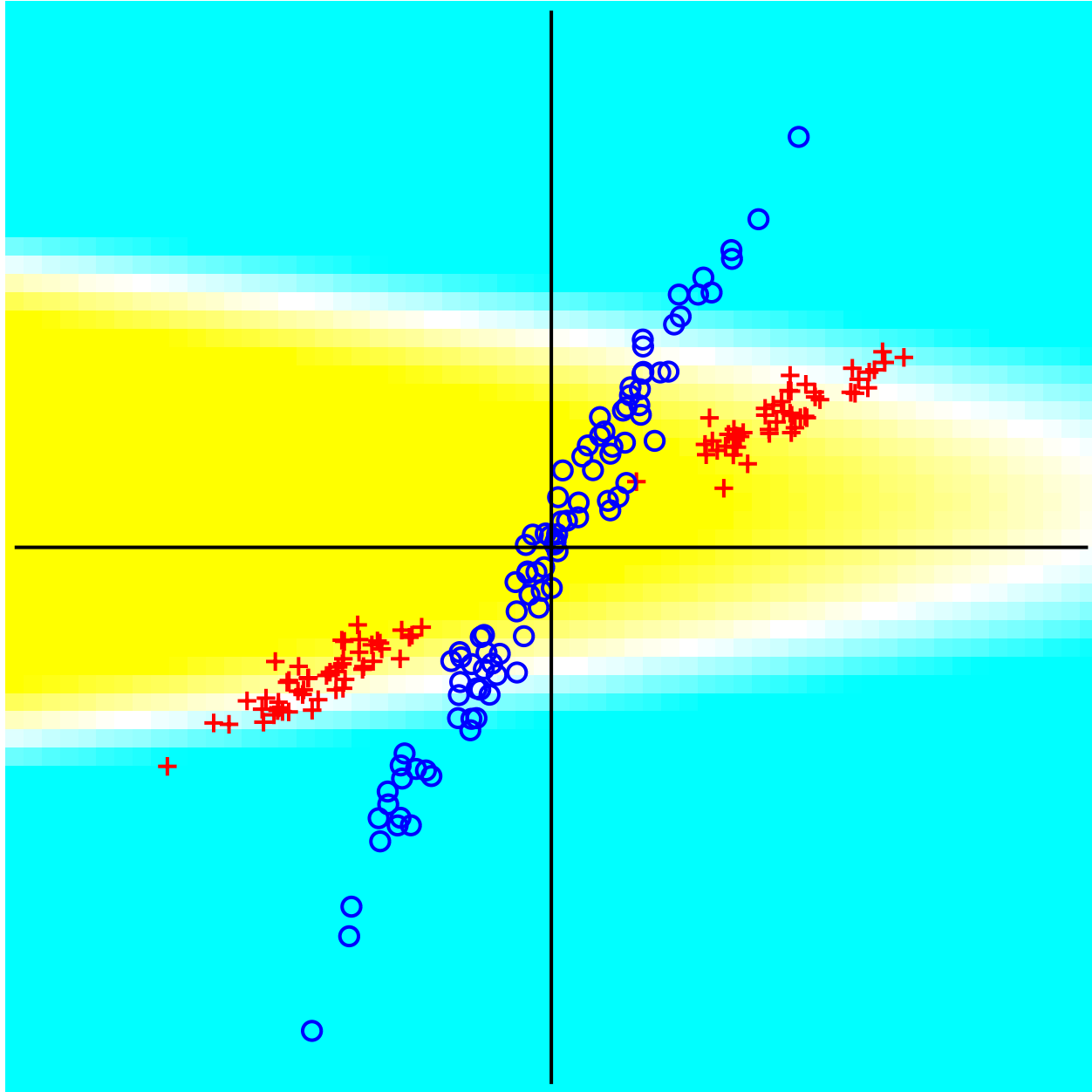
Split X, Disc: FLD, Embed: x_1, x_2 only



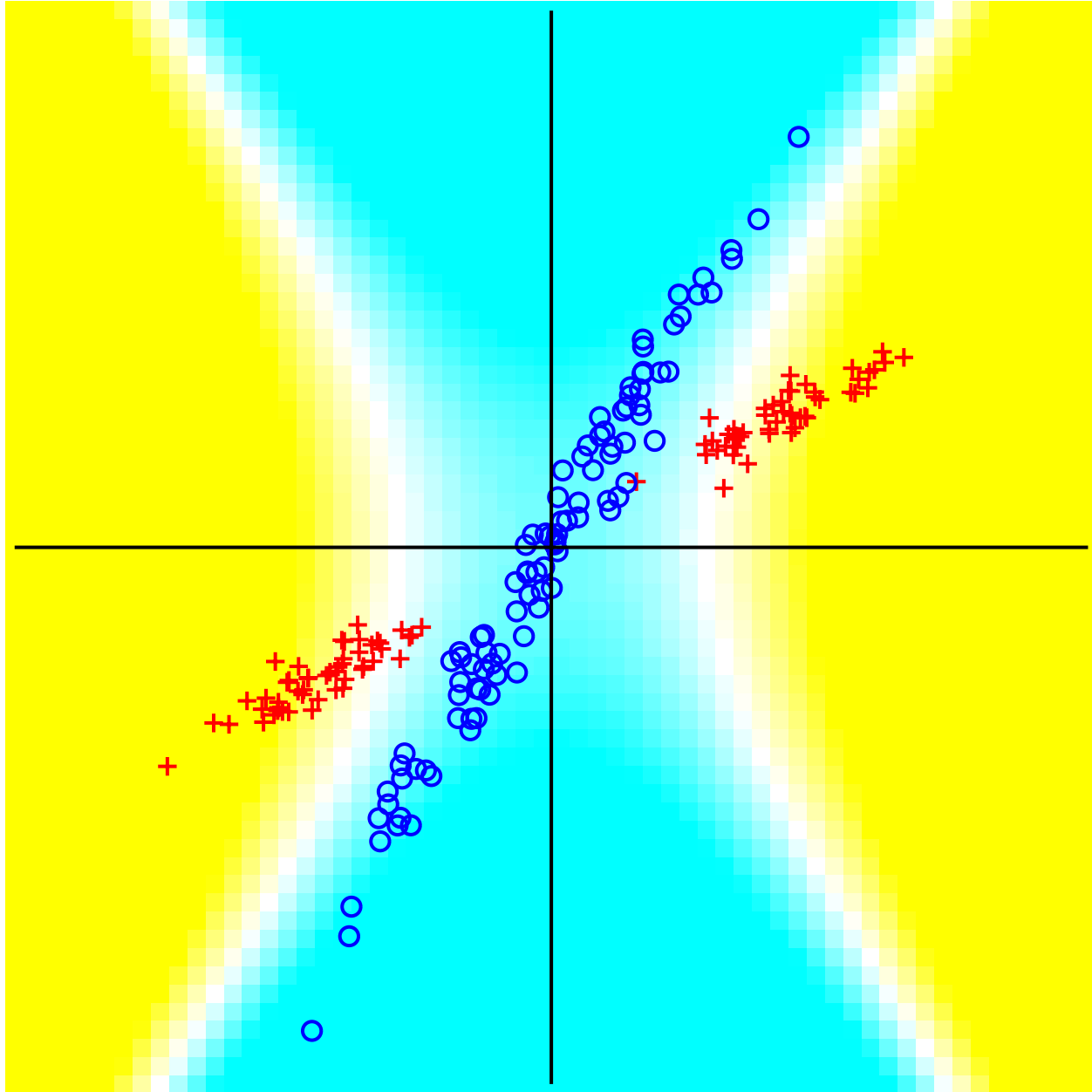
Split X, Disc: FLD, Embed: x_1, x_2, x_1^2



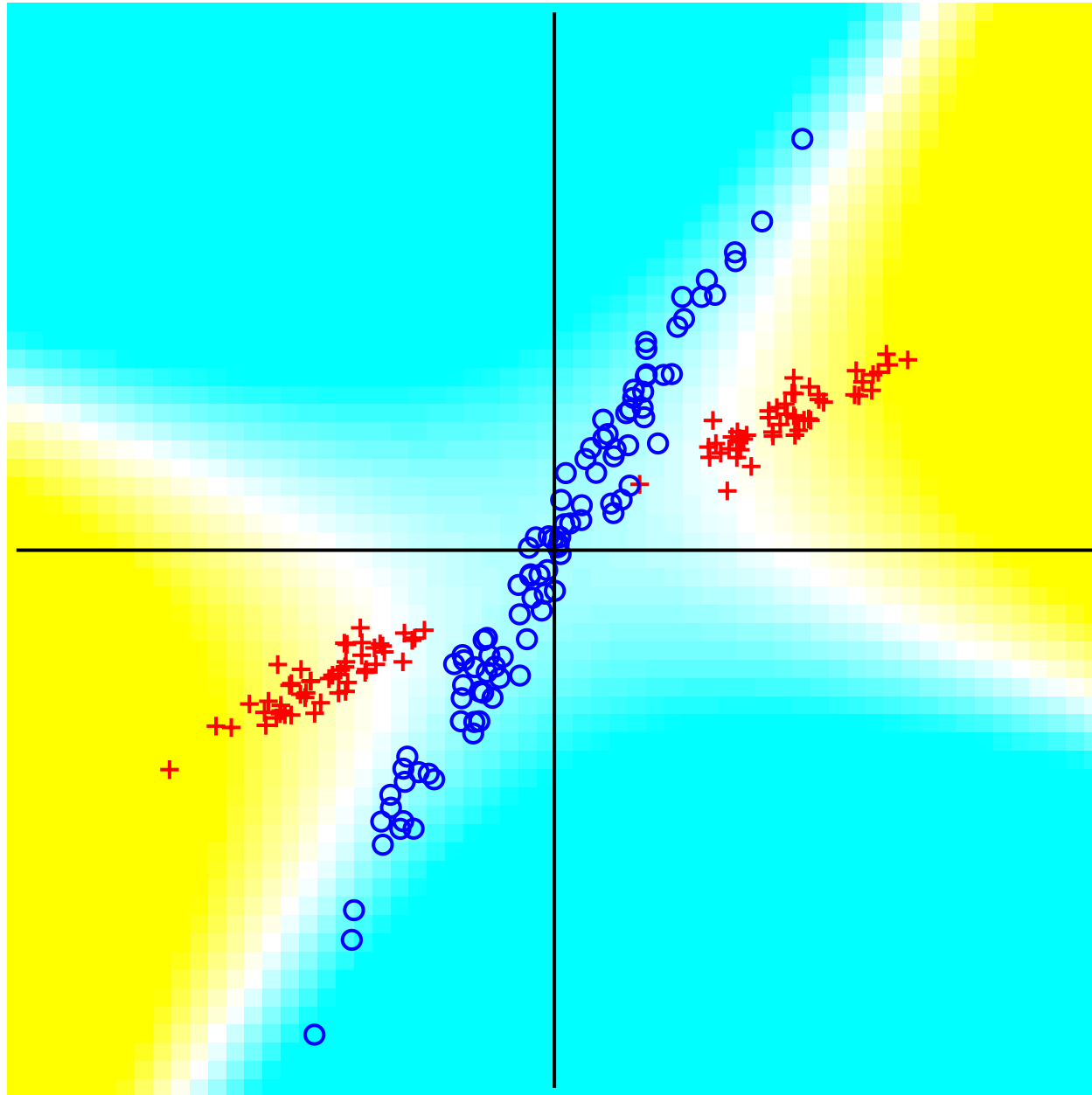
Split X, Disc: FLD, Embed: x_1, x_2, x_2^2



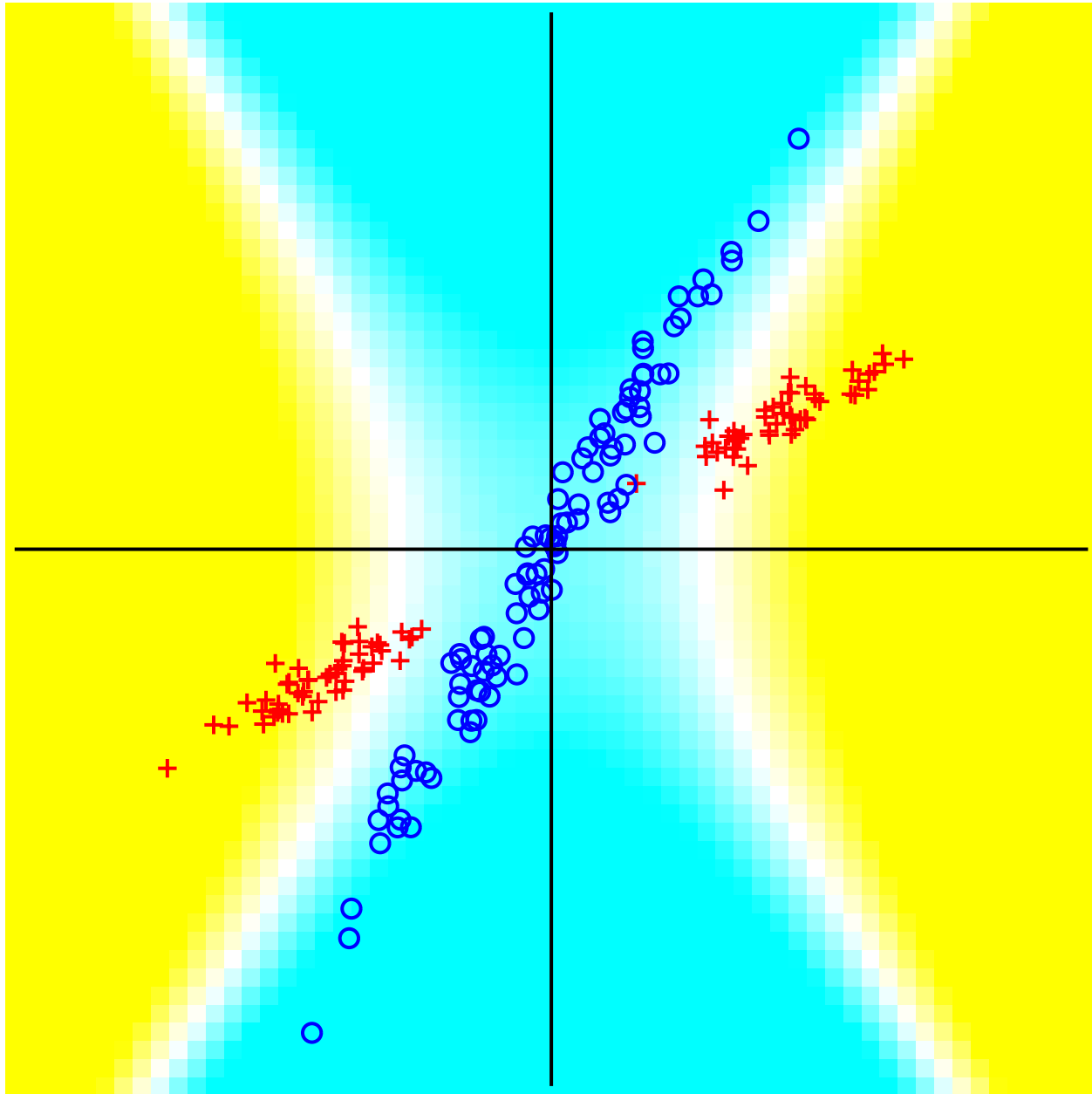
Split X, Disc: FLD, Embed: x_1, x_2, x_1^2, x_2^2



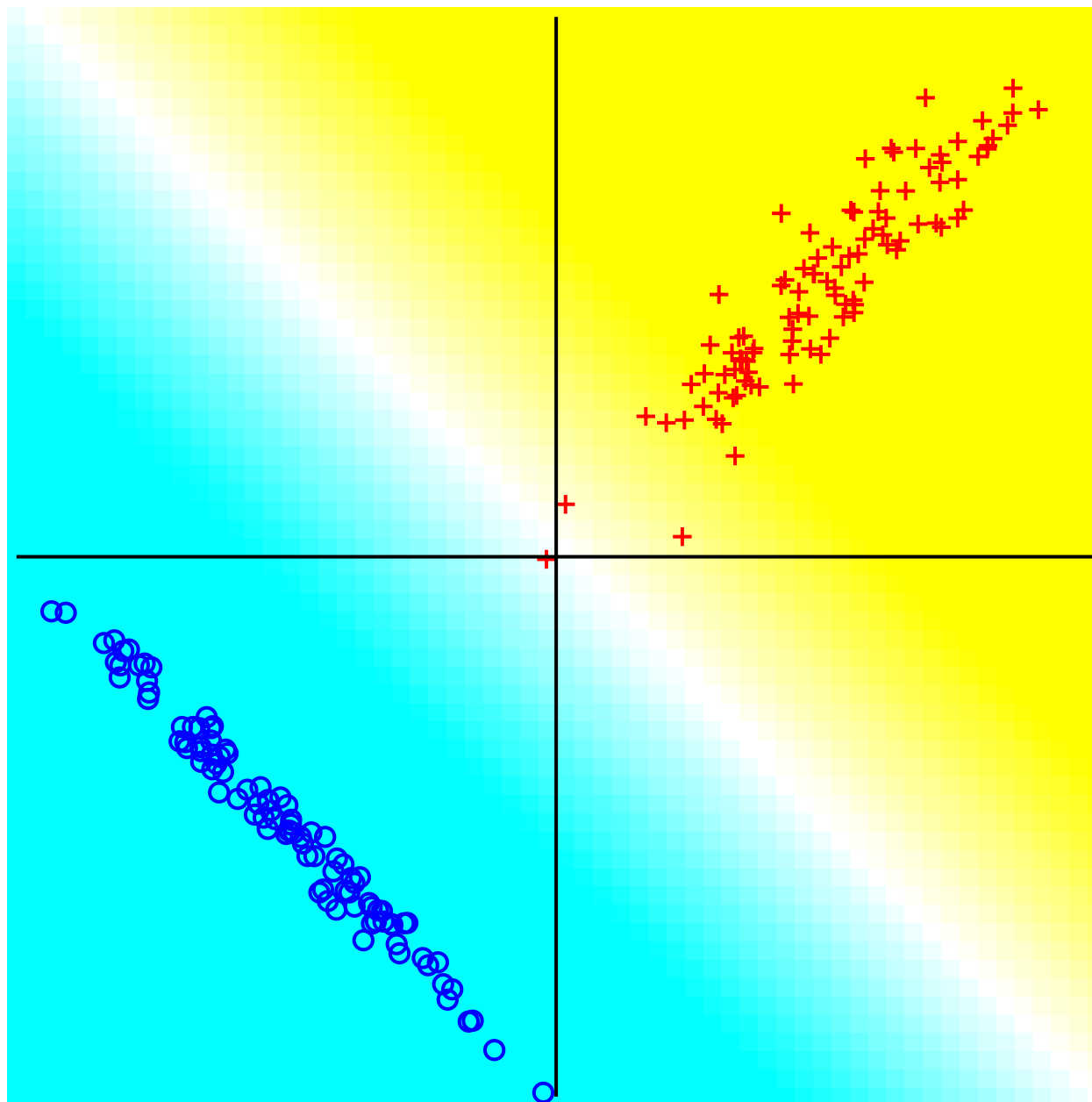
Split X, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1x_2$



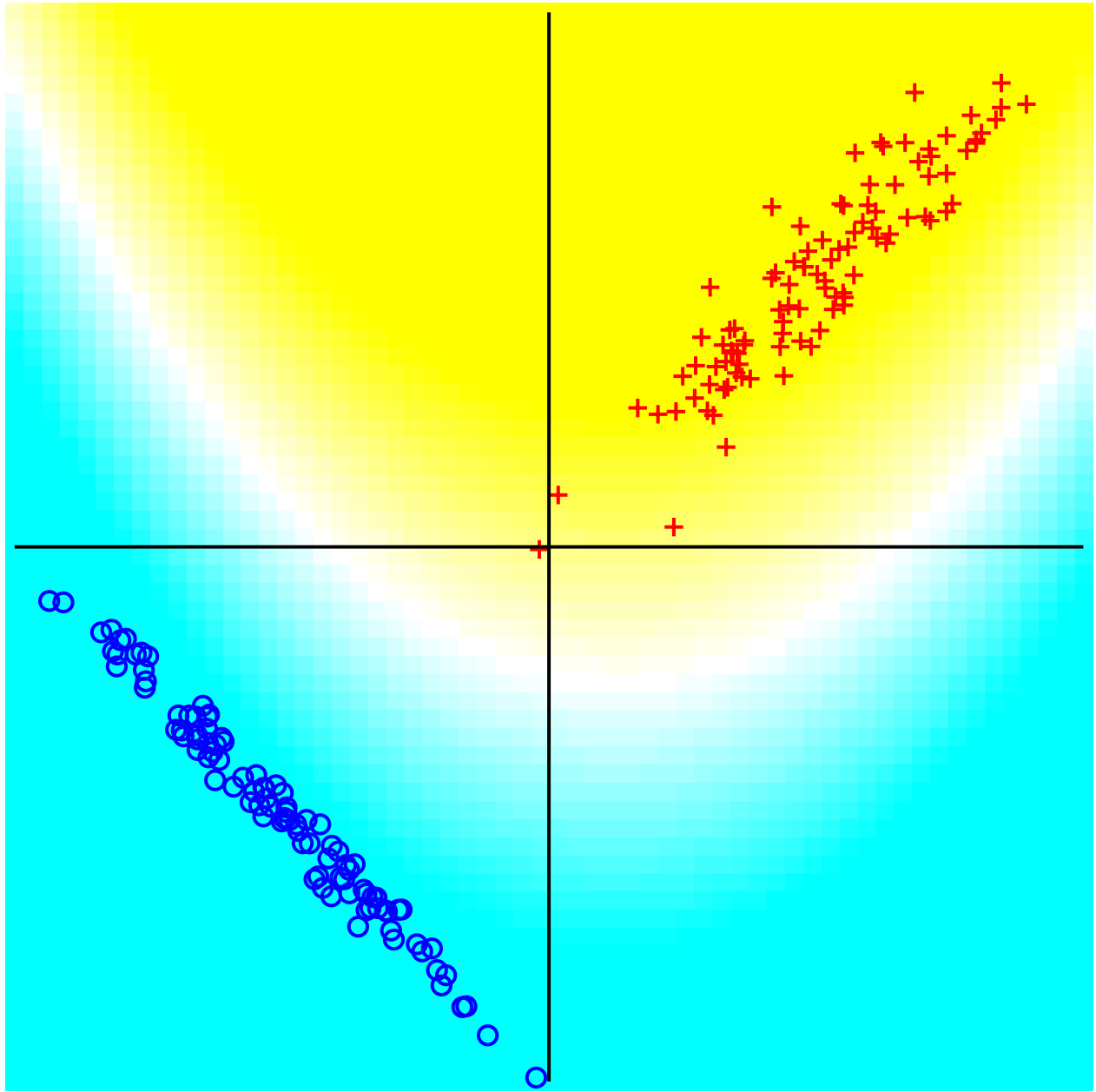
Split X, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3$



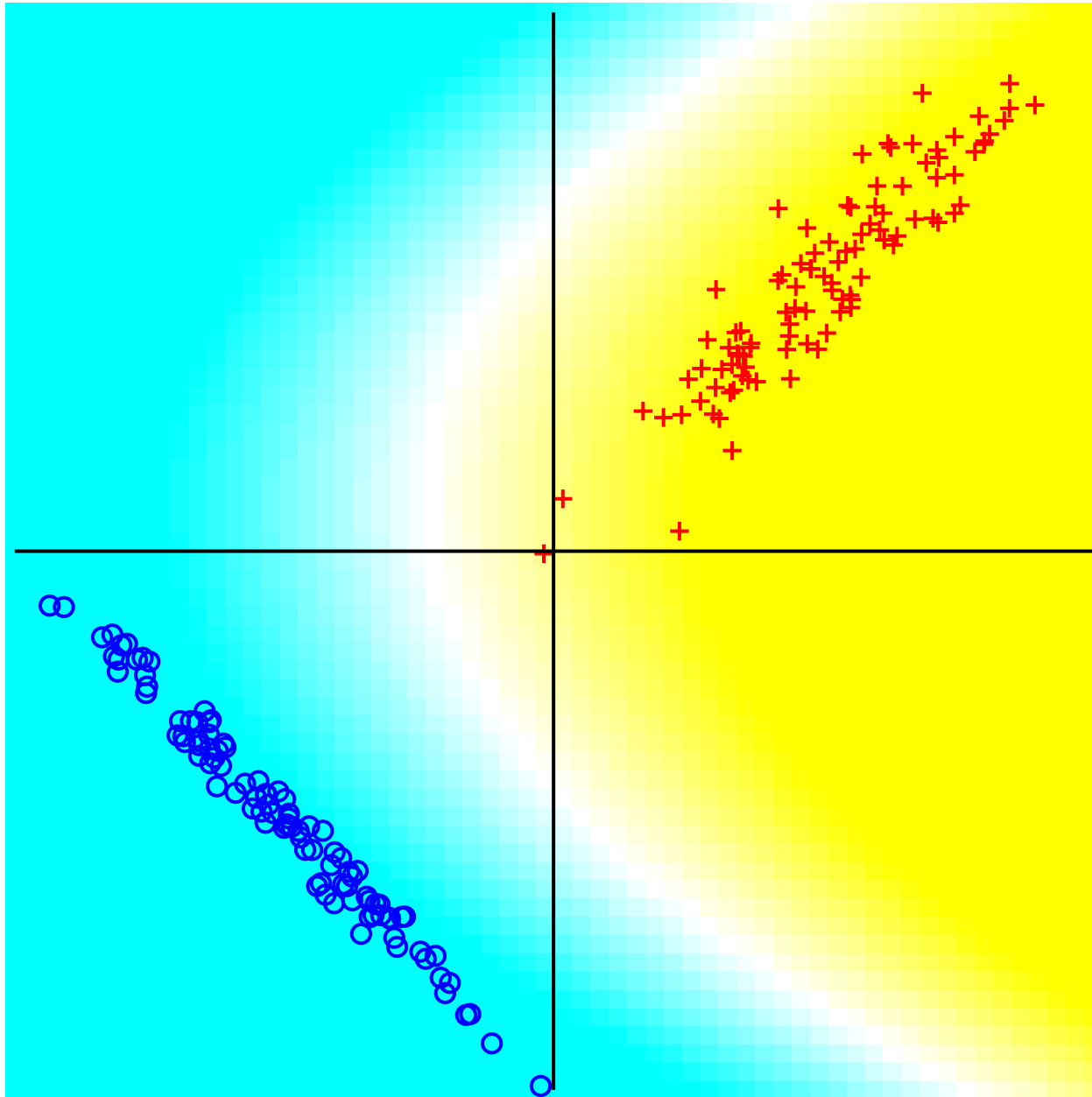
Two Cloud, Disc: FLD, Embed: x_1, x_2 only



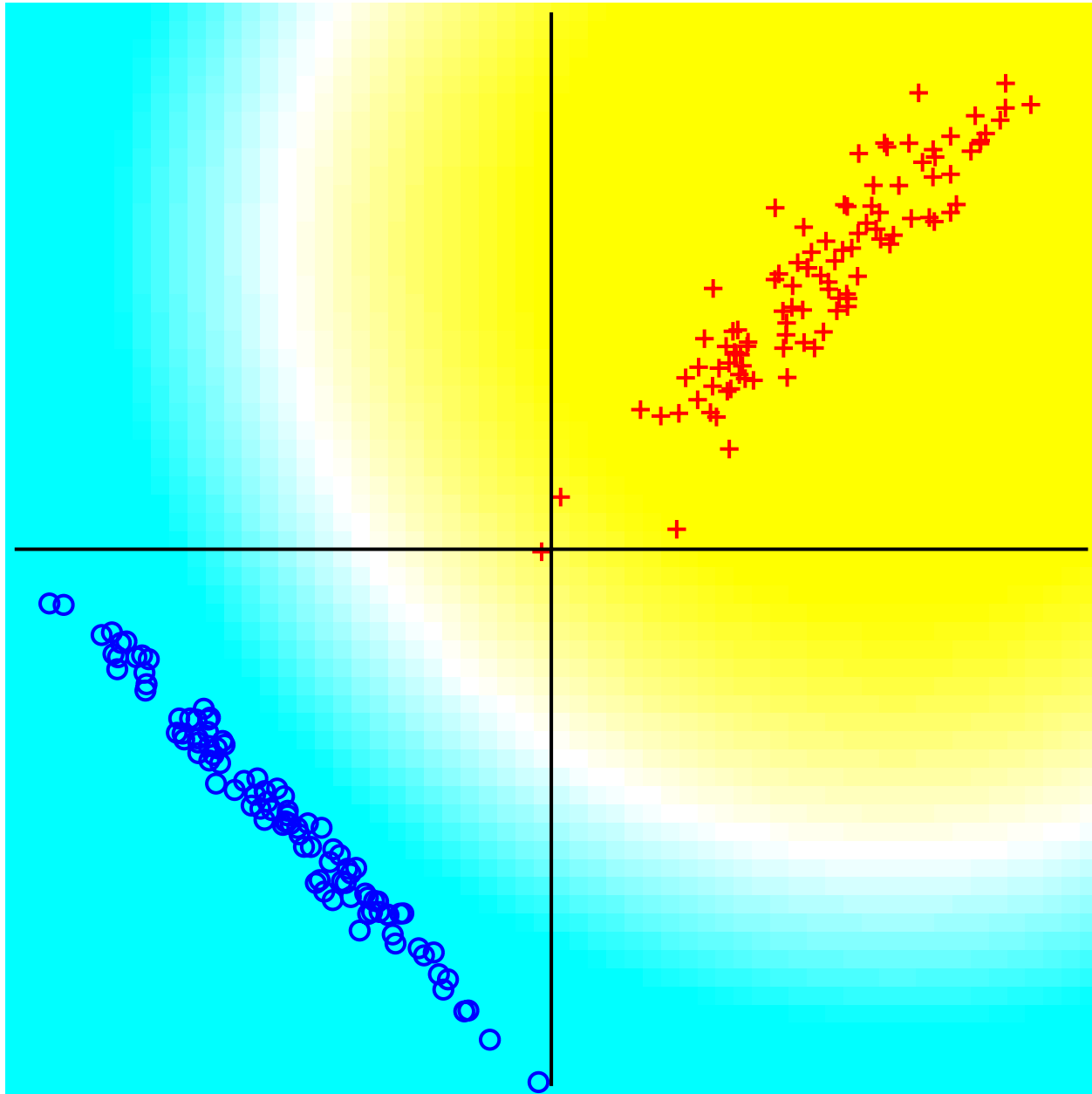
Two Cloud, Disc: FLD, Embed: x_1, x_2, x_1^2



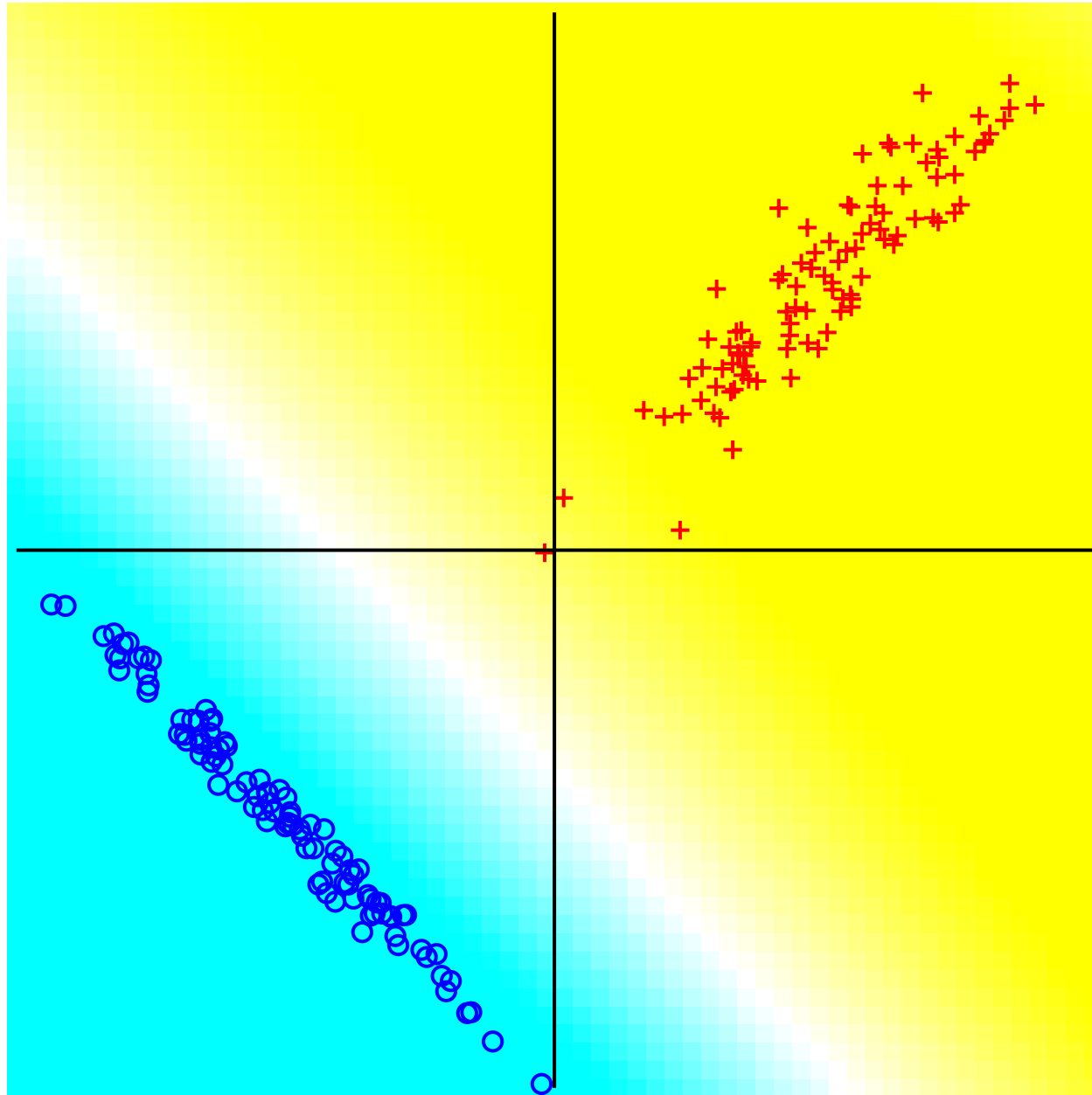
Two Cloud, Disc: FLD, Embed: x_1, x_2, x_2^2



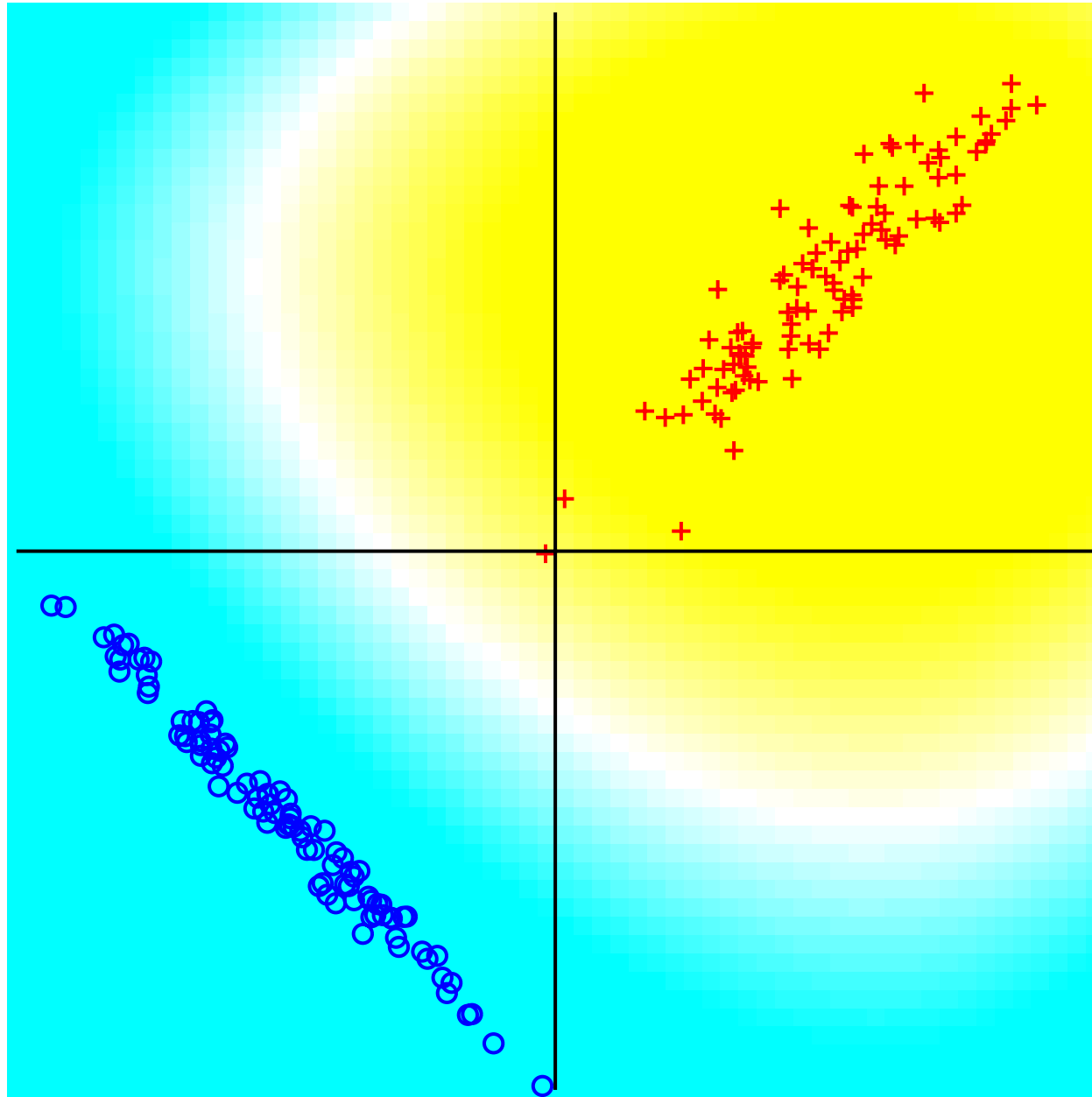
Two Cloud, Disc: FLD, Embed: x_1, x_2, x_1^2, x_2^2



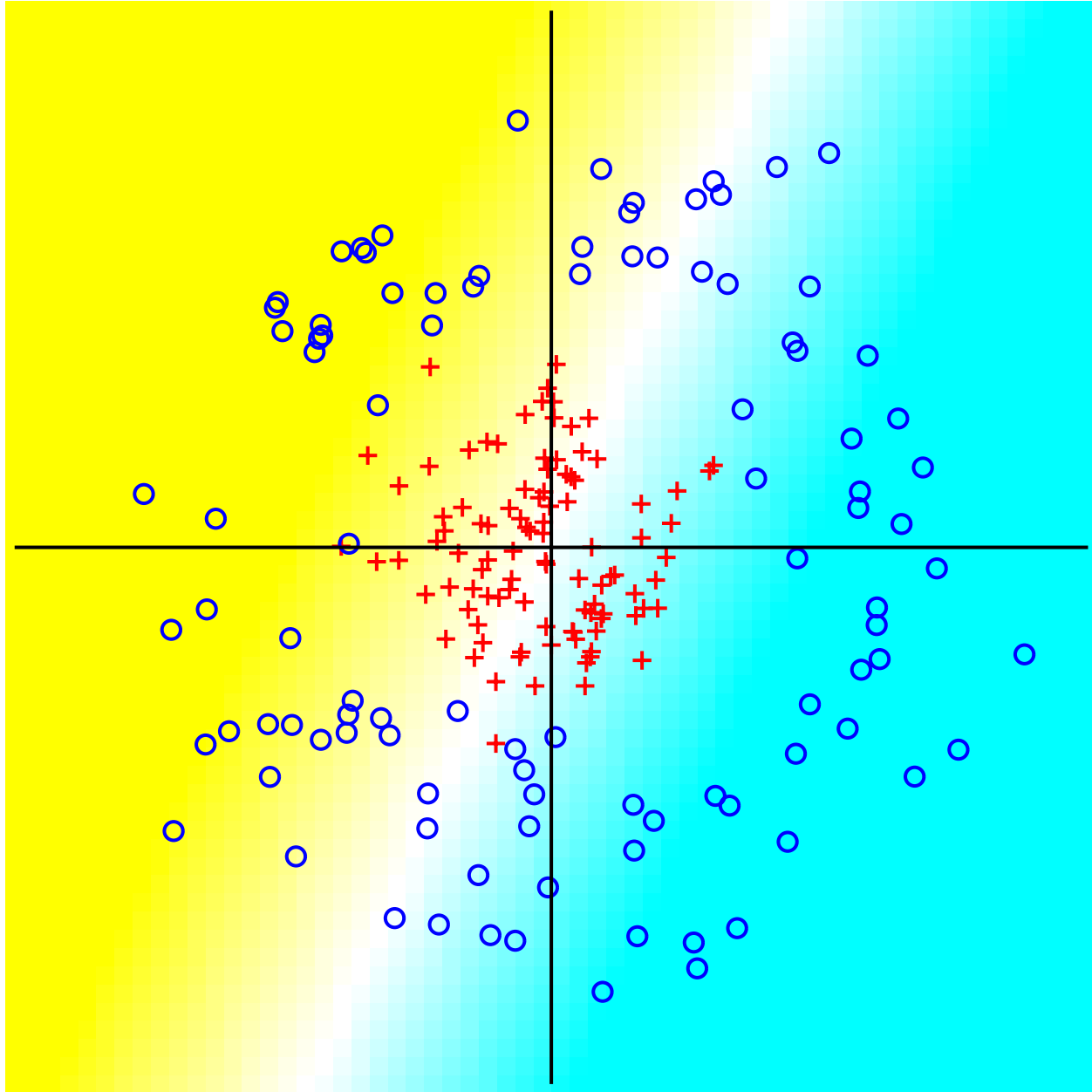
Two Cloud, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1x_2$



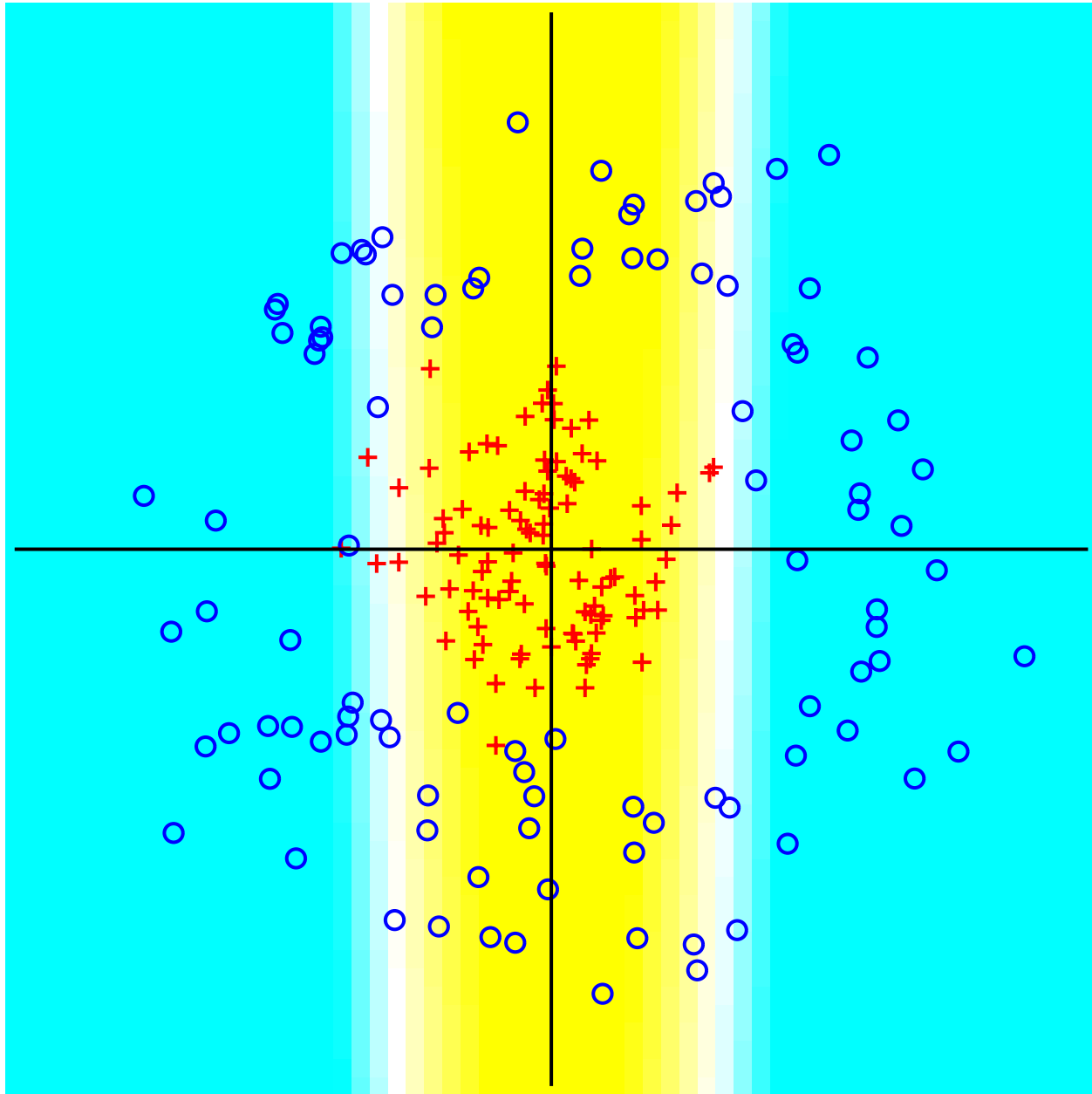
Two Cloud, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3$



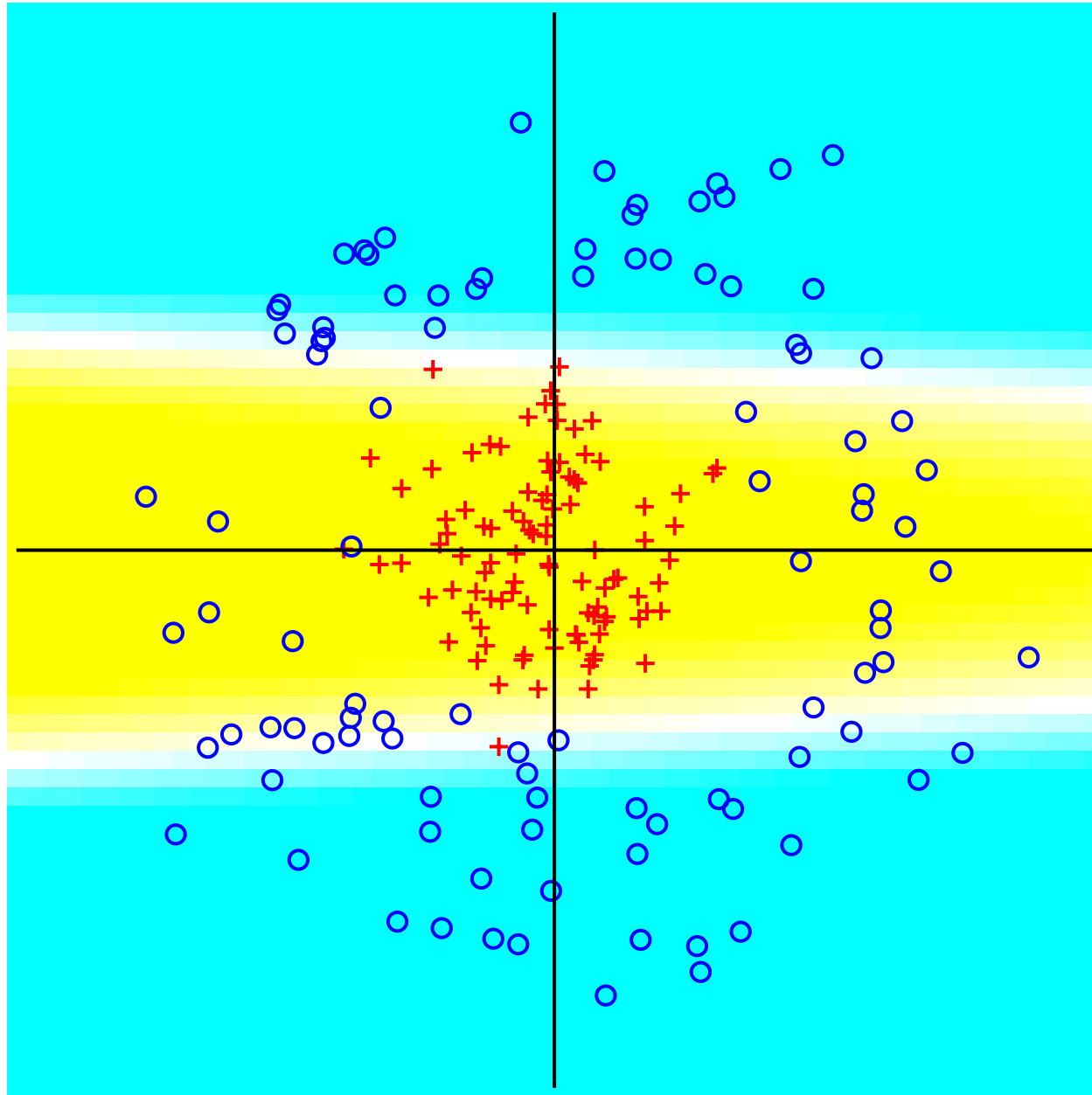
Donut, Disc: FLD, Embed: x_1, x_2 only



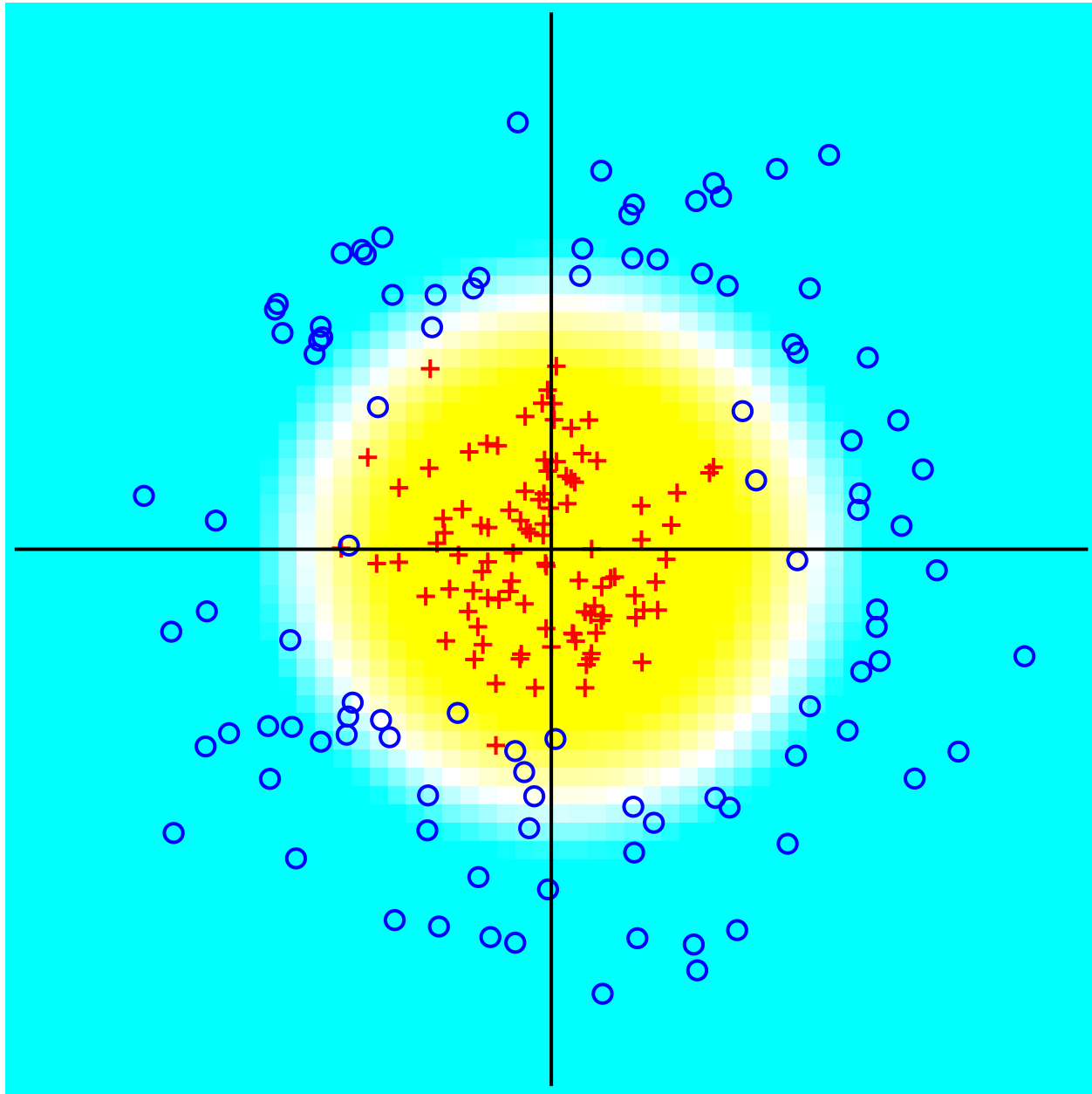
Donut, Disc: FLD, Embed: x_1, x_2, x_1^2



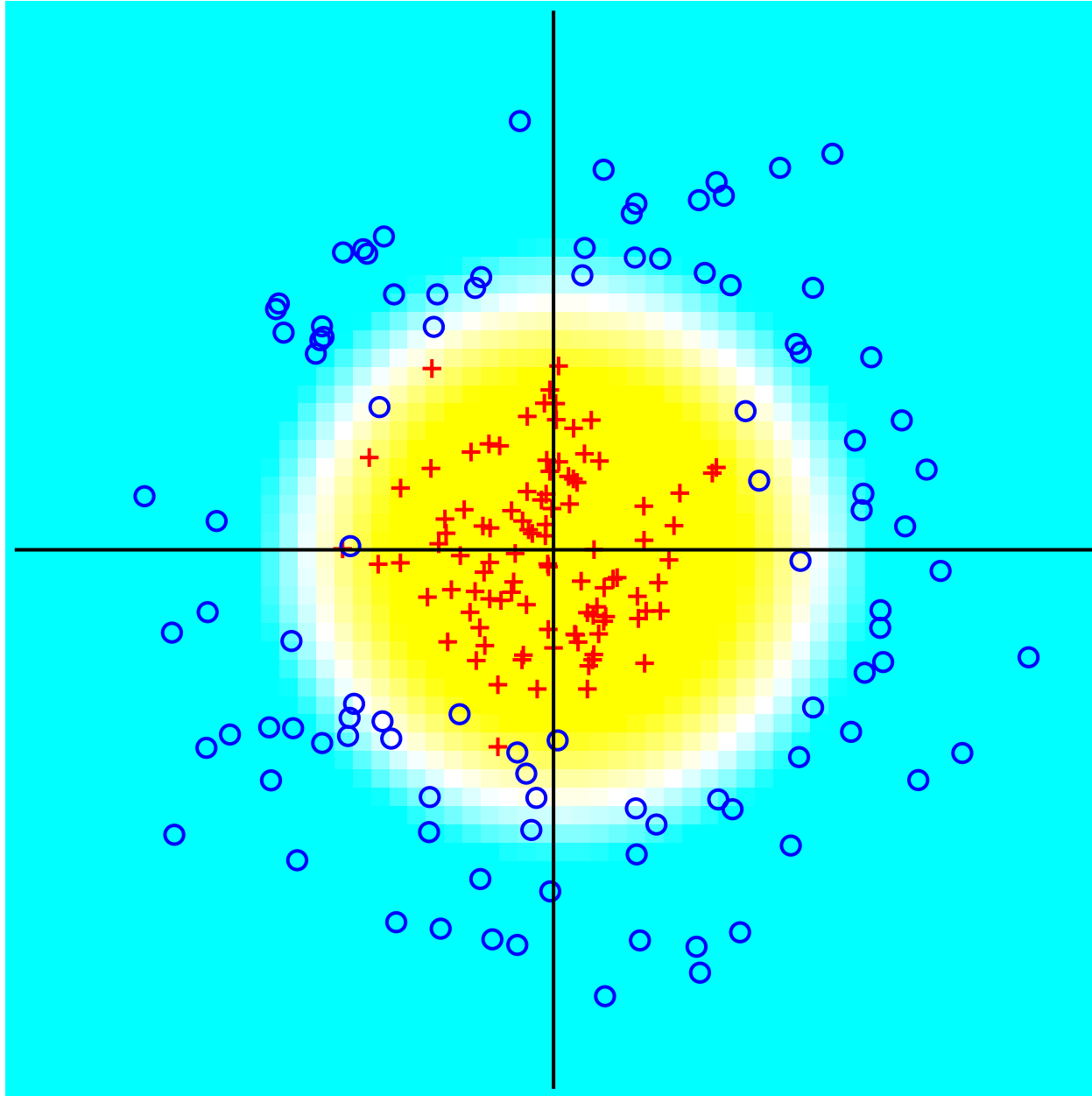
Donut, Disc: FLD, Embed: x_1, x_2, x_2^2



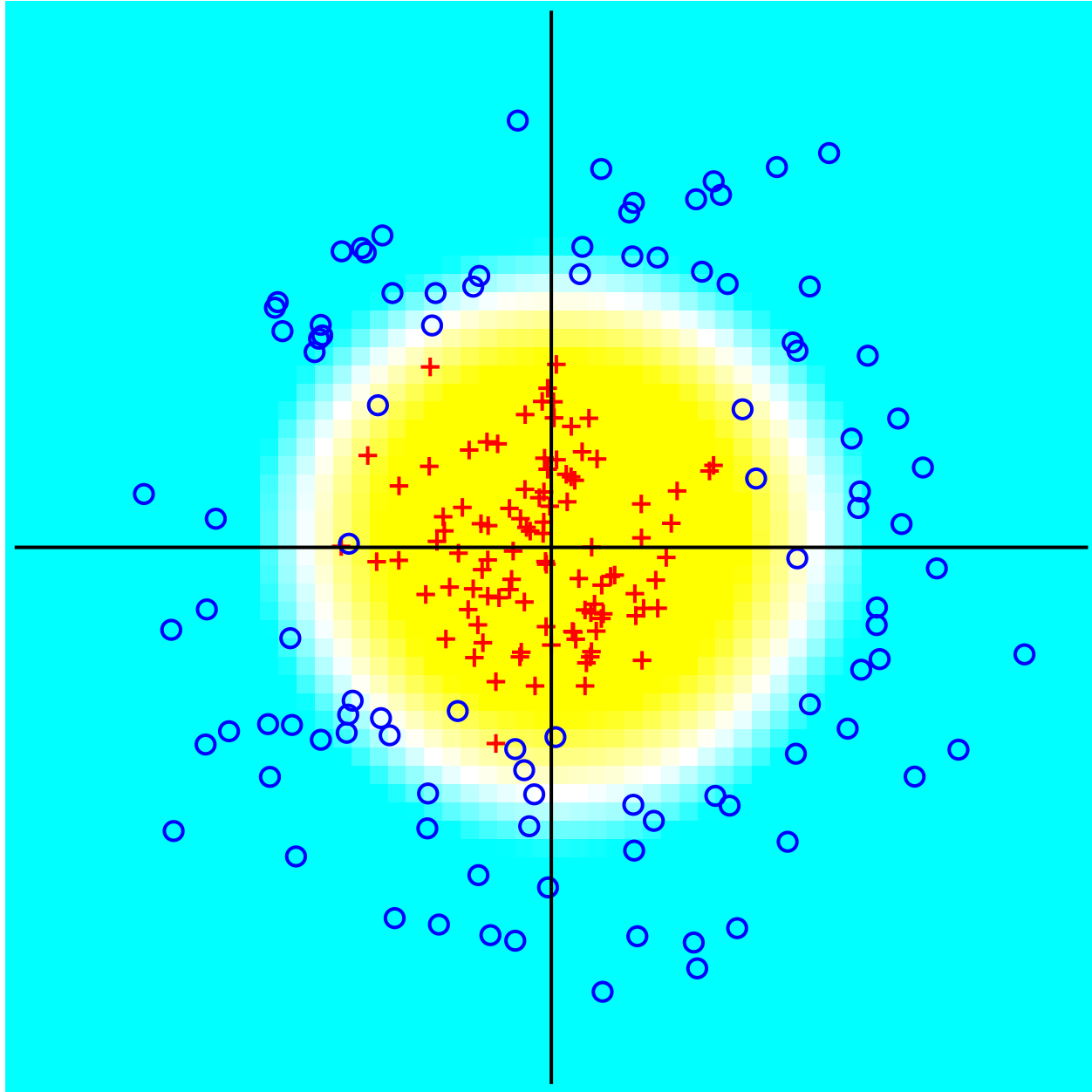
Donut, Disc: FLD, Embed: x_1, x_2, x_1^2, x_2^2



Donut, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1x_2$

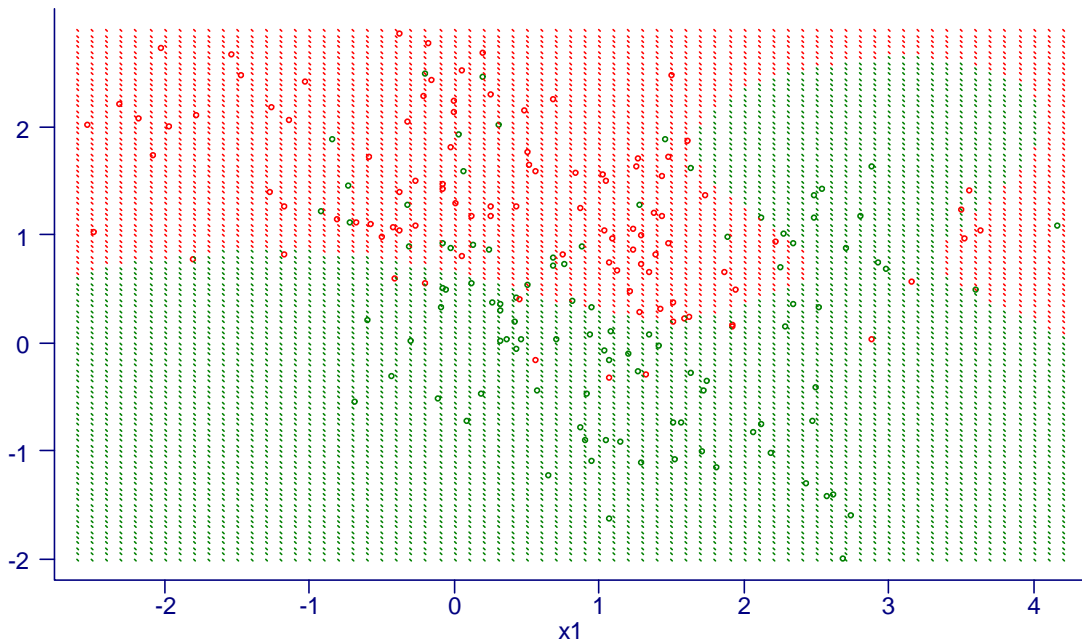


Donut, Disc: FLD, Embed: $x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3$



○ Original data: class 0
○ B-spline prediction: class 0

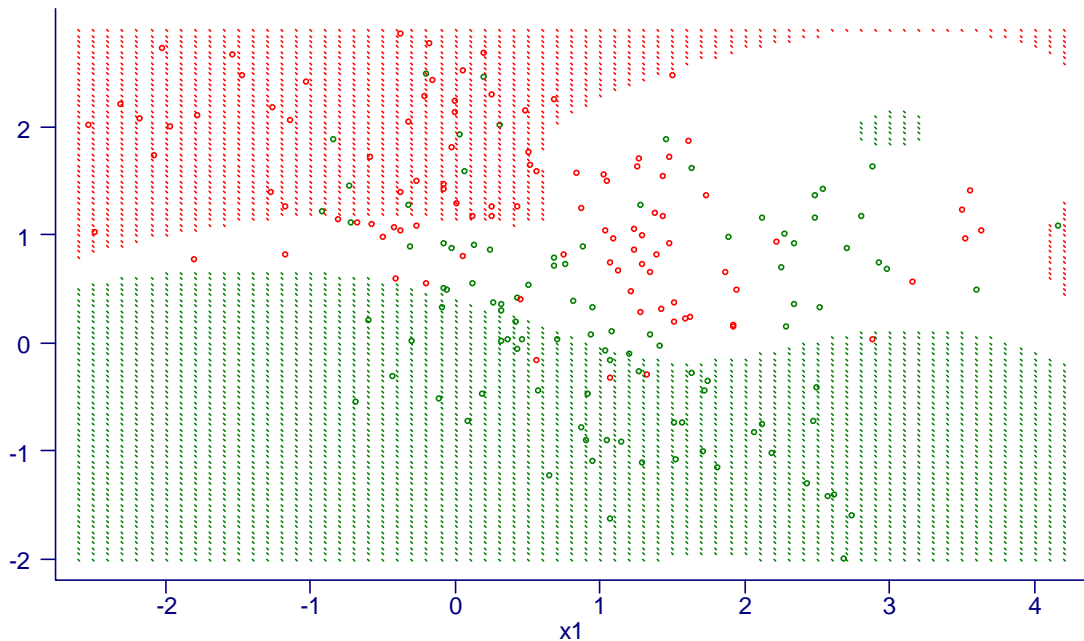
○ Original data: class 1
○ B-spline prediction: class 1



Classification with B-splines

○ Original data: class 0
● B-spline prediction: class 0

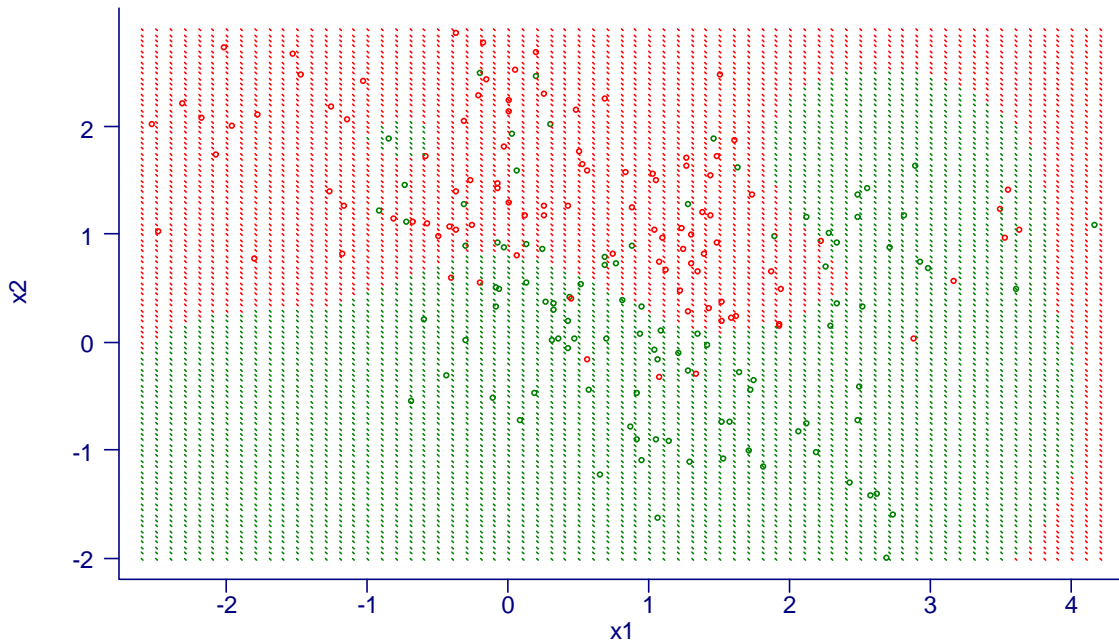
○ Original data: class 1
● B-spline prediction: class 1



Classification with B-splines

○ Original data: class 0
○ TP spline prediction: class 0

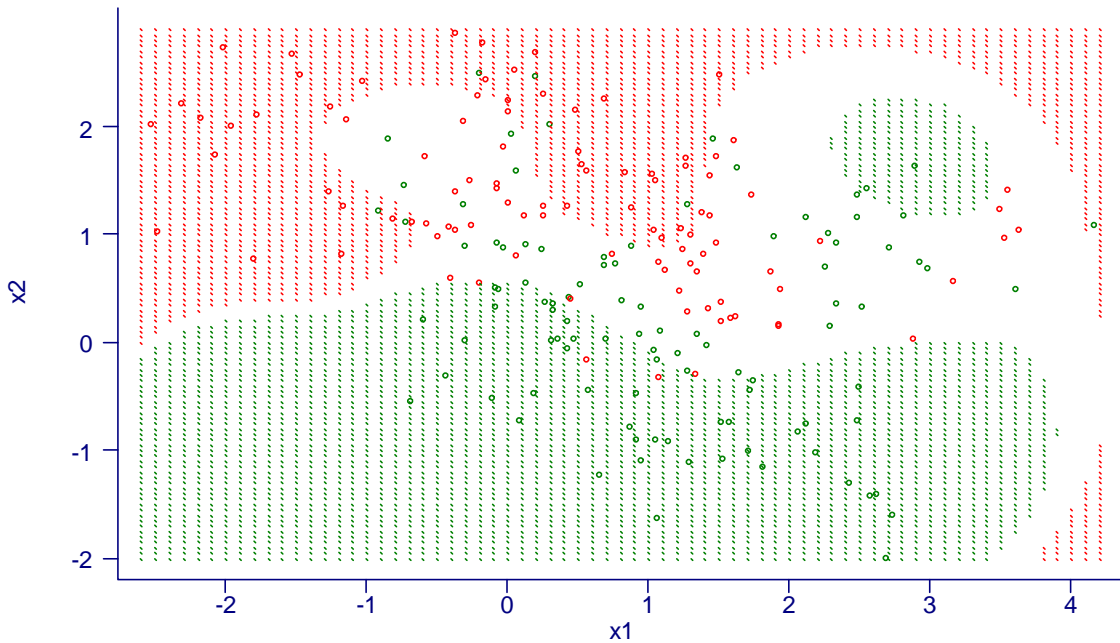
○ Original data: class 1
○ TP spline prediction: class 1



Classification with thin-plate splines

○ Original data: class 0
○ Weaker TP spline prediction: cl

○ Original data: class 1
○ Weaker TP spline prediction: cl



Classification with thin-plate splines

Bayes Optimal Classifier

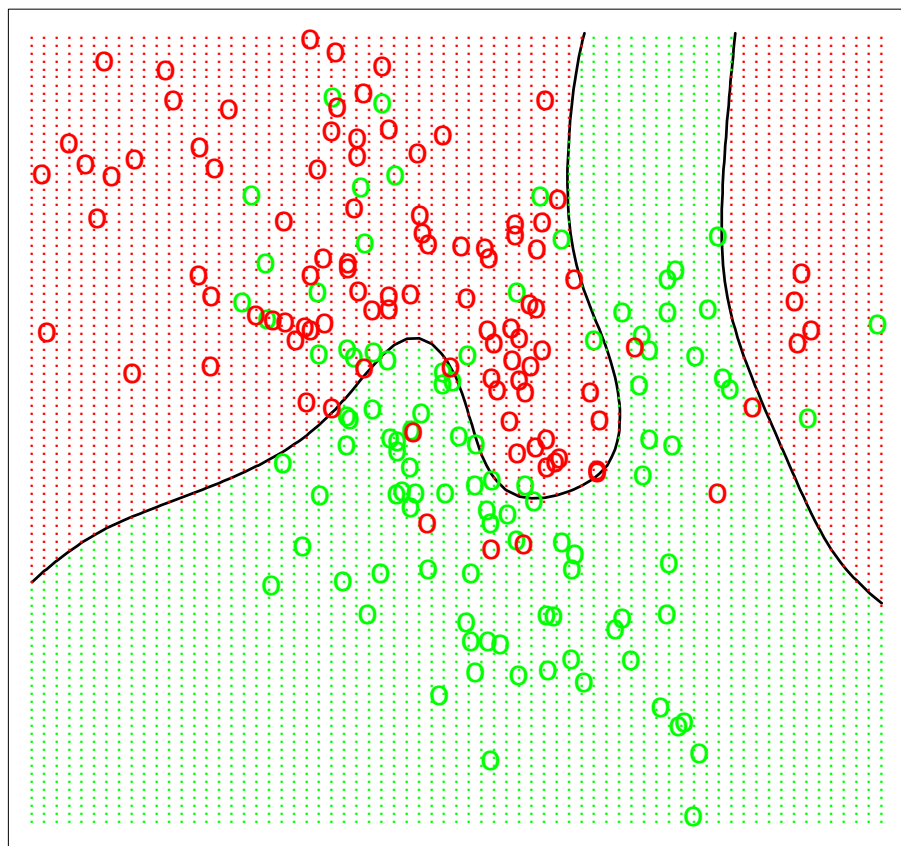


Figure 2.5: *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

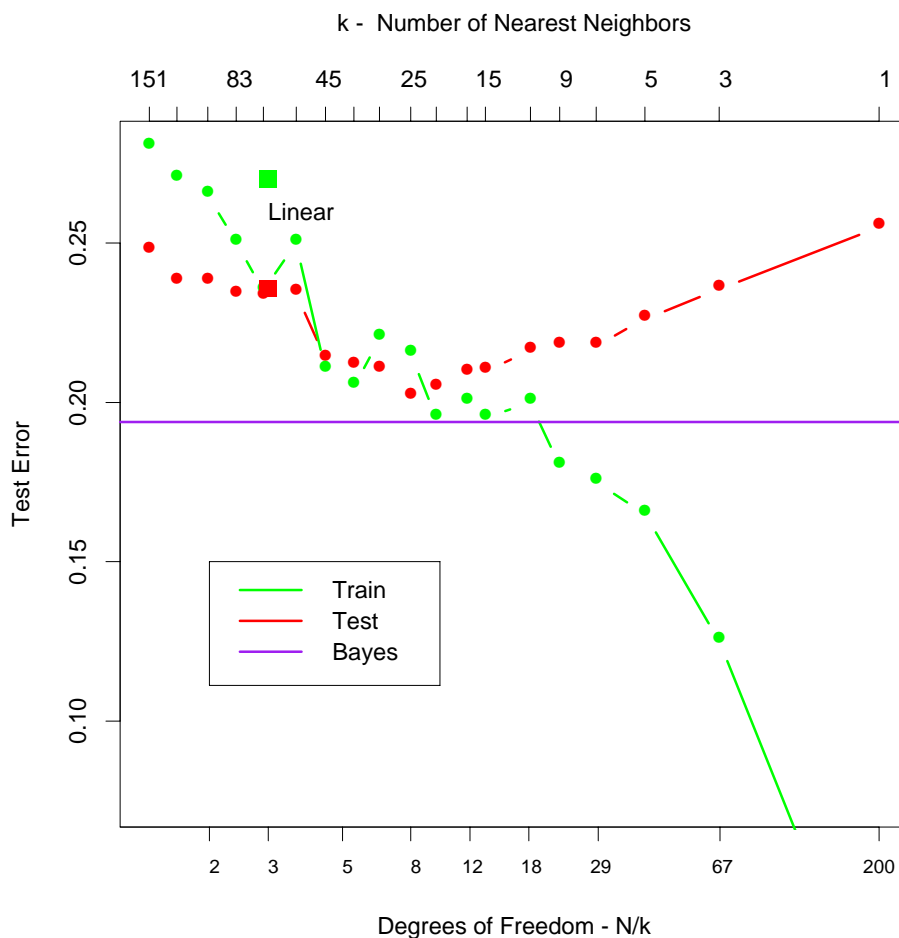


Figure 2.4: *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The red curves are test and the green are training error for k -nearest-neighbor classification. The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple line is the optimal Bayes Error Rate.*

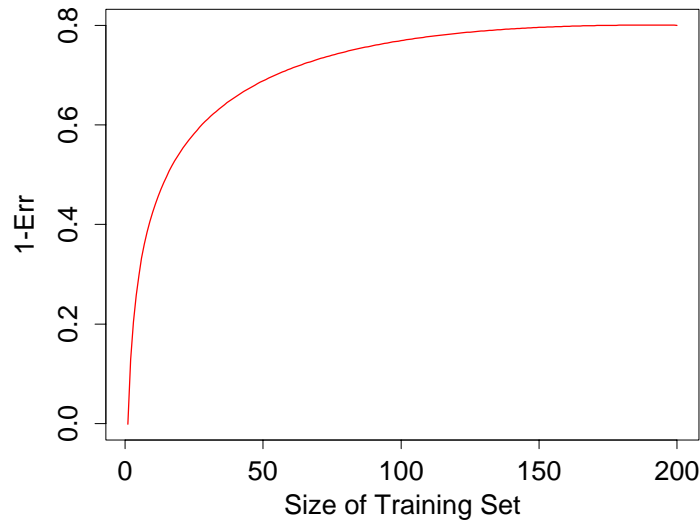
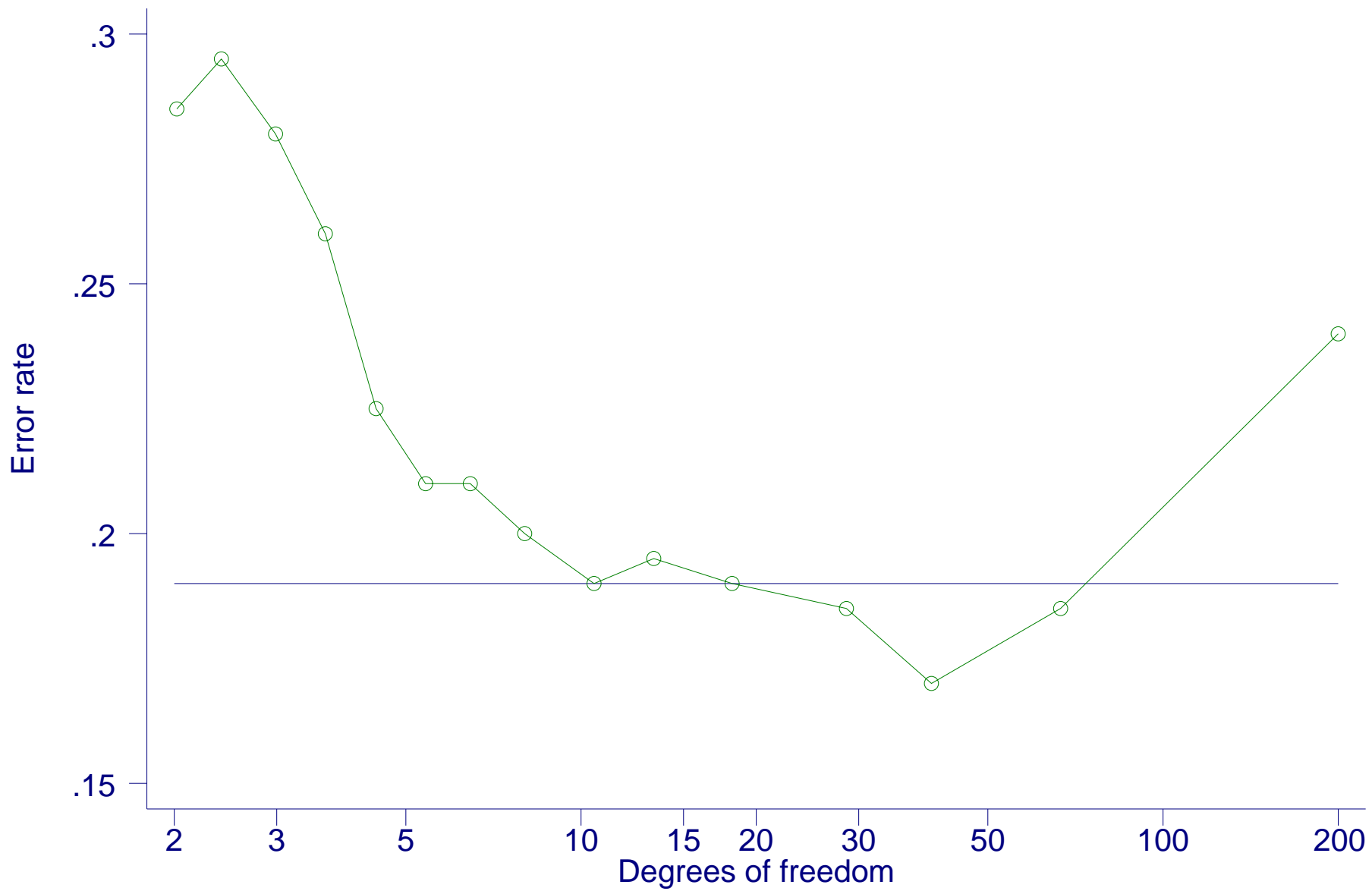
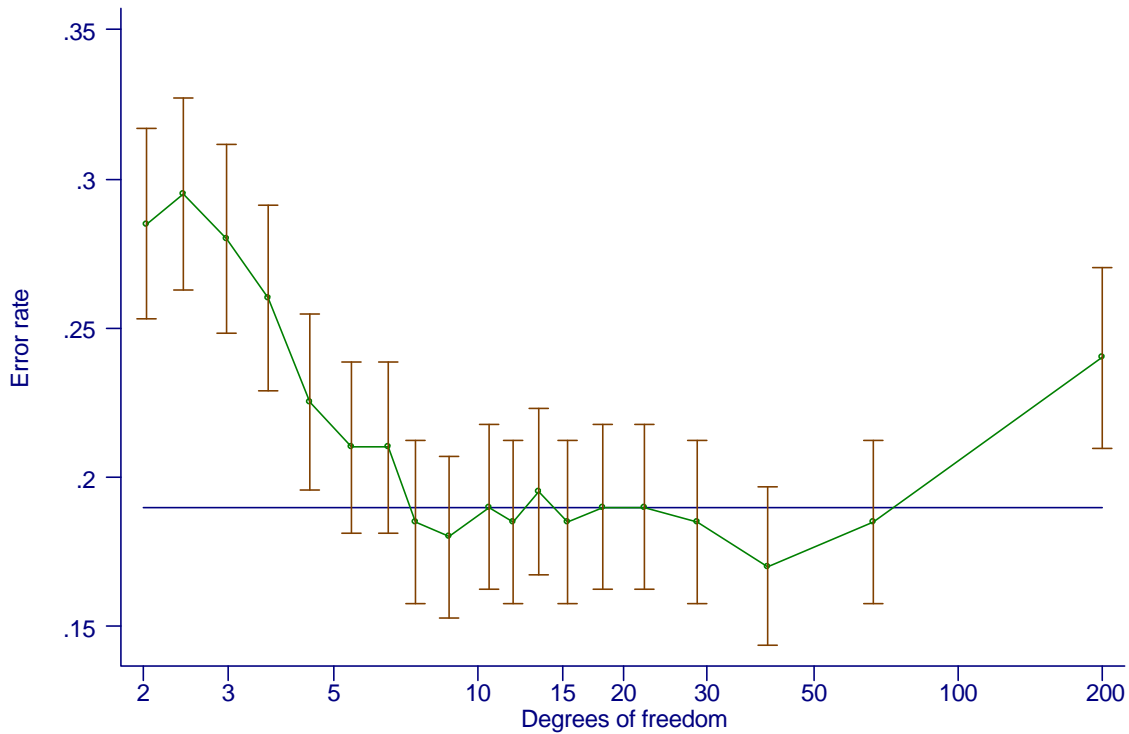


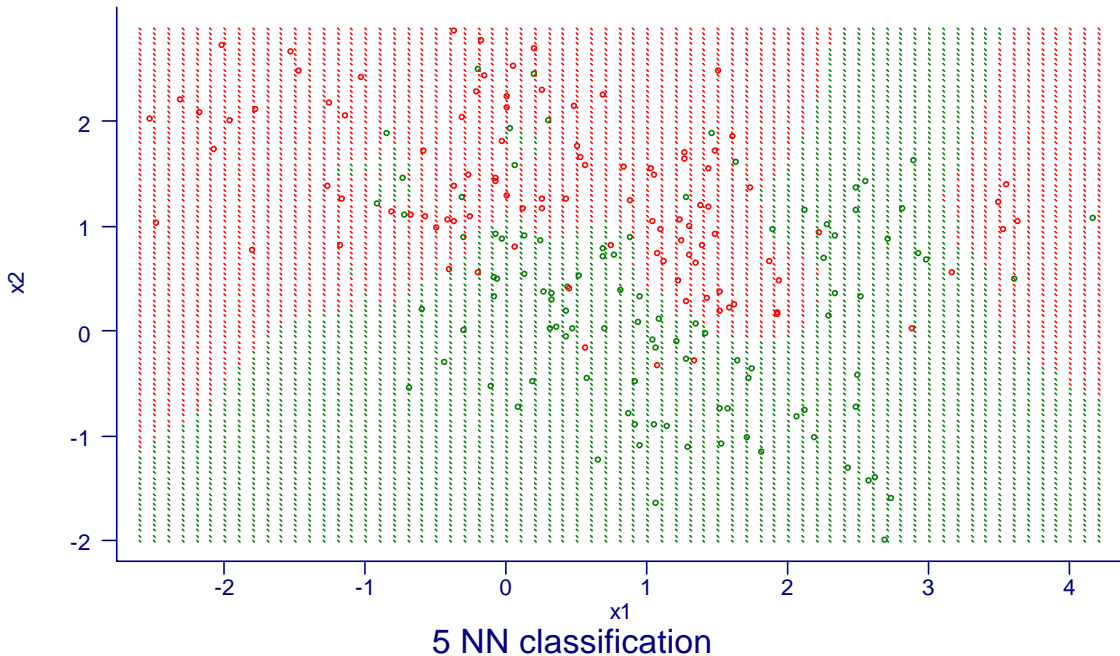
Figure 7.8: *Hypothetical learning curve for a classifier on a given task; a plot of $1 - \text{Err}$ versus the size of the training set N . With a dataset of 200 observations, fivefold cross-validation would use training sets of size 160, which would behave much like the full set. However, with a dataset of 50 observations fivefold cross-validation would use training sets of size 40, and this would result in a considerable overestimate of prediction error.*

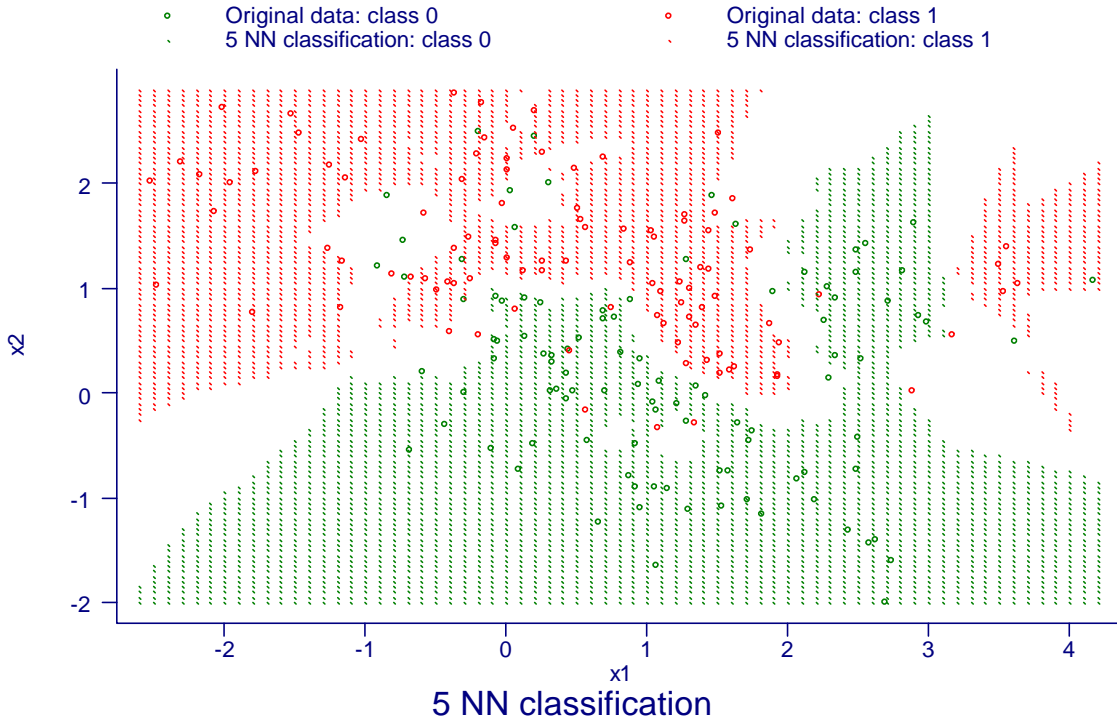




○ Original data: class 0
○ 5 NN classification: class 0

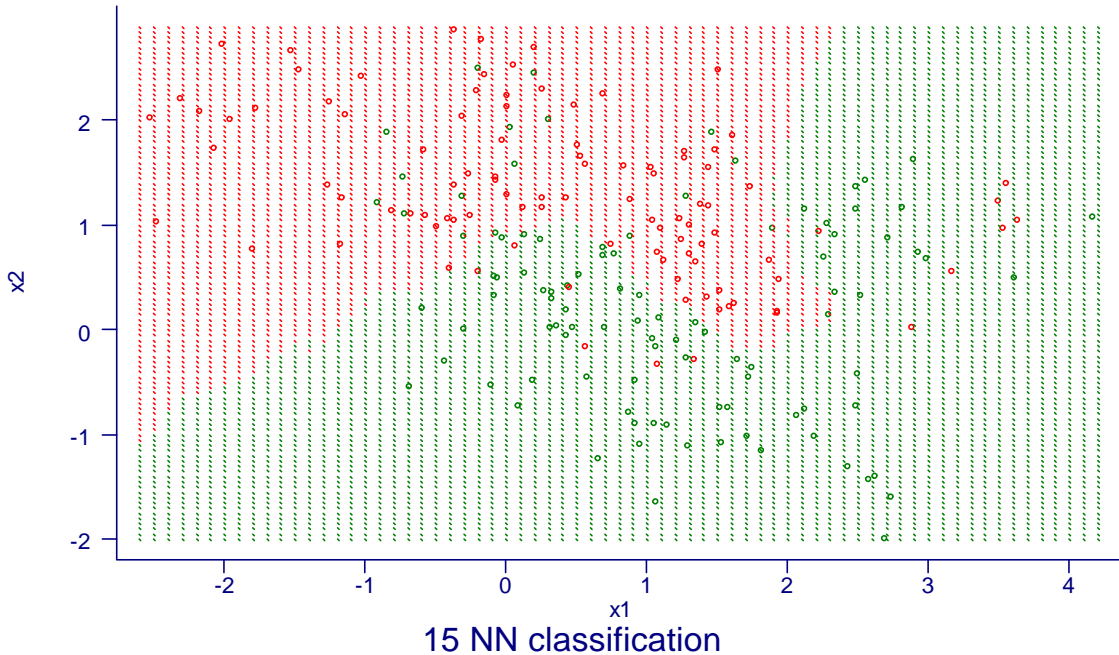
○ Original data: class 1
○ 5 NN classification: class 1





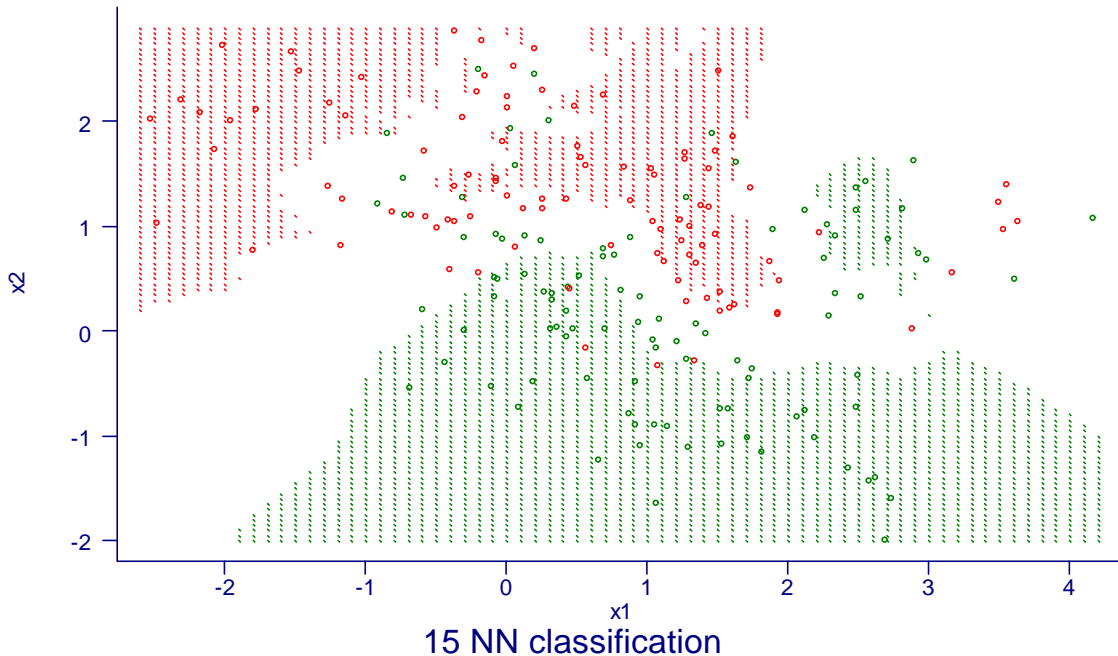
○ Original data: class 0
● 15 NN classification: class 0

○ Original data: class 1
● 15 NN classification: class 1



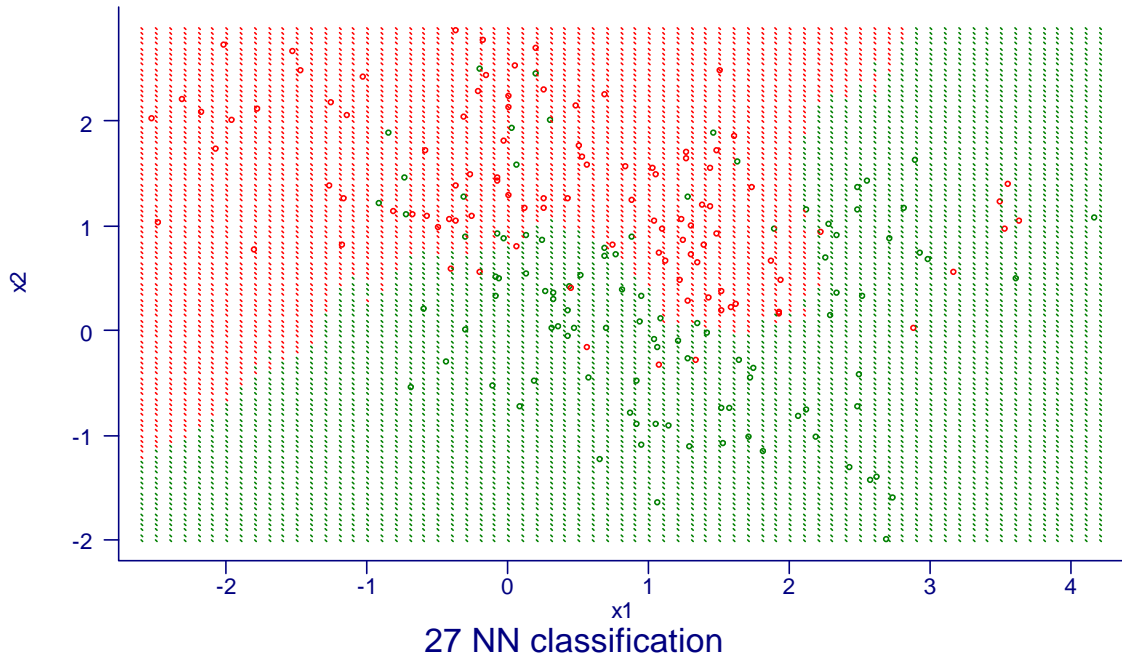
Original data: class 0
15 NN classification: class 0

Original data: class 1
15 NN classification: class 1



○ Original data: class 0
○ 27 NN classification: class 0

○ Original data: class 1
○ 27 NN classification: class 1



○ Original data: class 0
○ 27 NN classification: class 0

○ Original data: class 1
○ 27 NN classification: class 1

