Multivariate classification

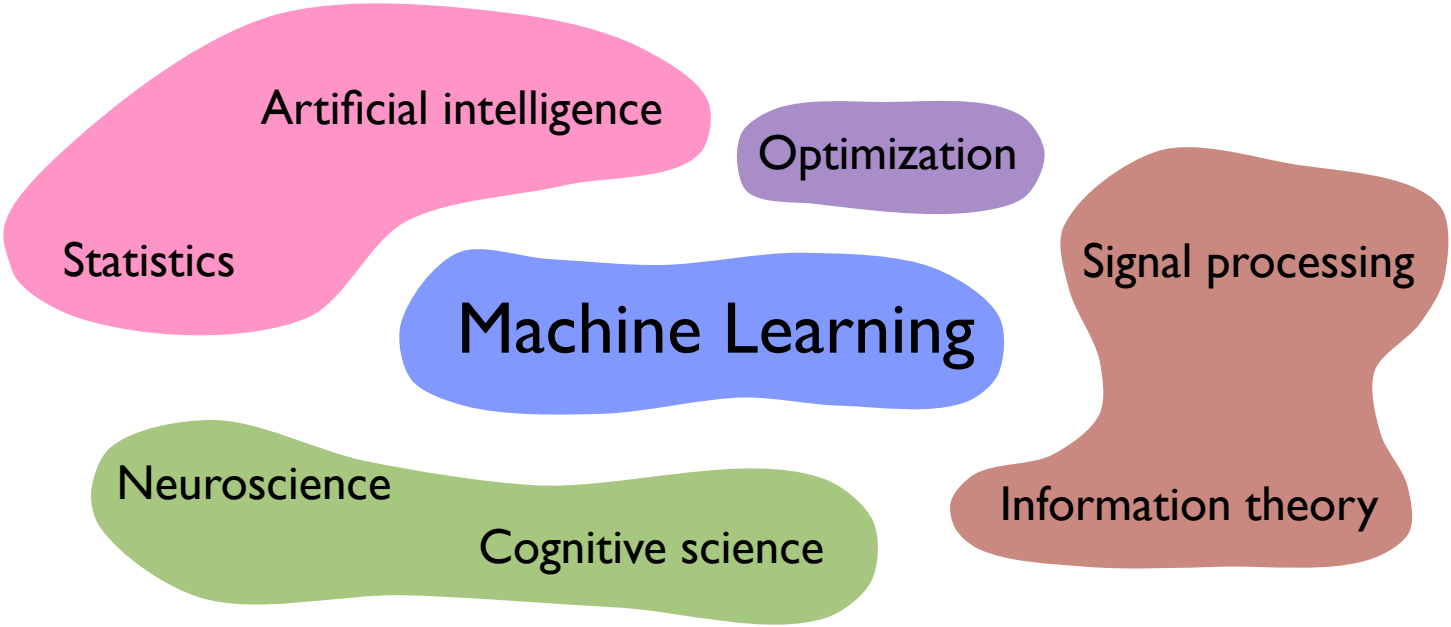
Balázs Kégl
(pronounced Bolage)
LAL / University of Paris-Sud

School of Statistics
27 May, 2014

Machine learning

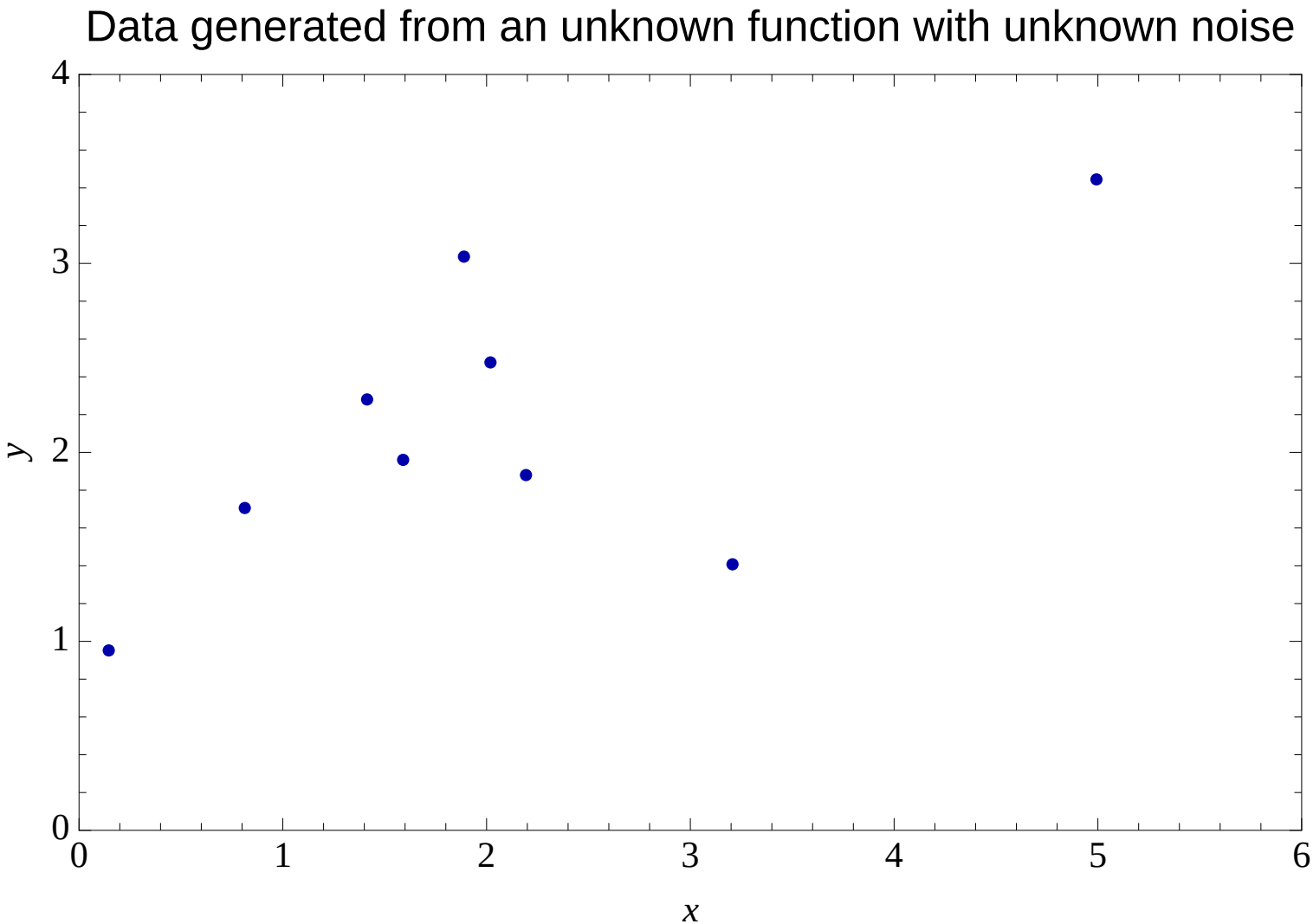
- Can be considered as a sub-domain of statistics
  - Often non-parametric, high-dimensional spaces, large data sets $\rightarrow$ computational issues (optimization) become as important as statistical issues (capacity control)
  - Most often associated to multivariate discrimination (classification)
- Best-known algorithms
  - Classification: neural network, boosting, support vector machine
  - Regression: neural network, Gaussian process
  - Density estimaton: mixture of Gaussians
  - Clustering, dimensionality reduction: PCA, k-means, spectral clustering, local linear embedding, ISOMAP, nonlinear PCA, kernel PCA

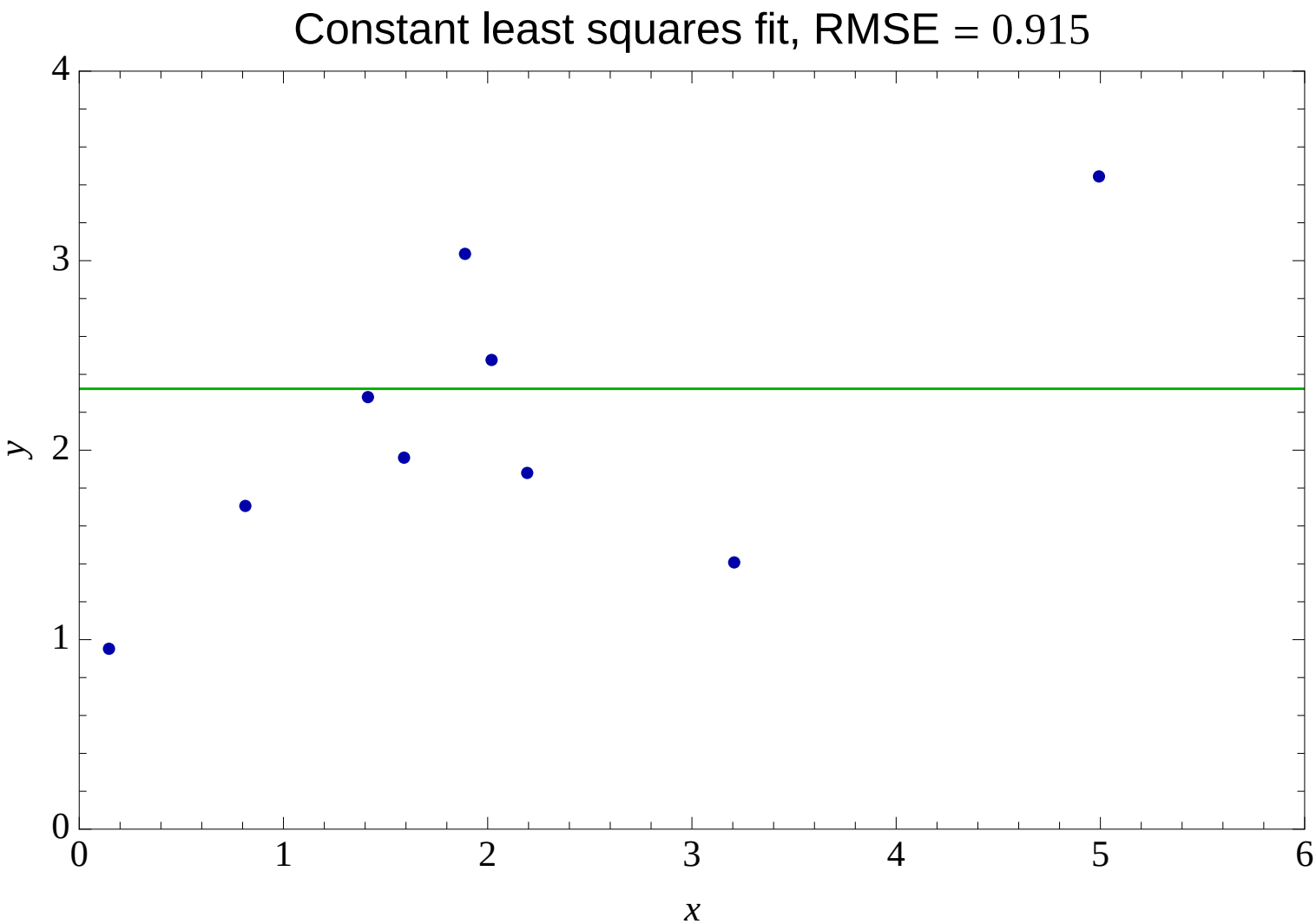
Machine learning at the crossroads

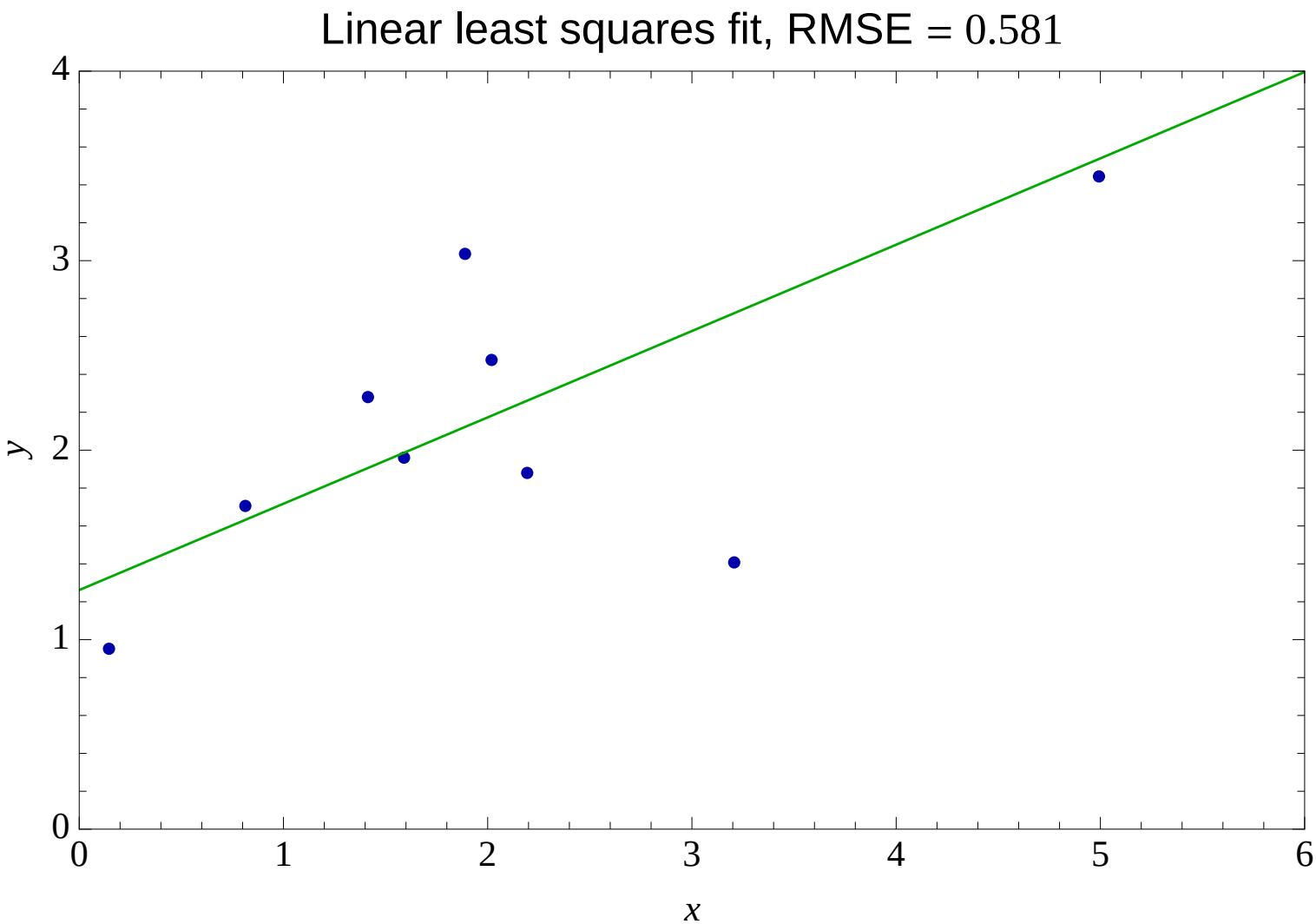


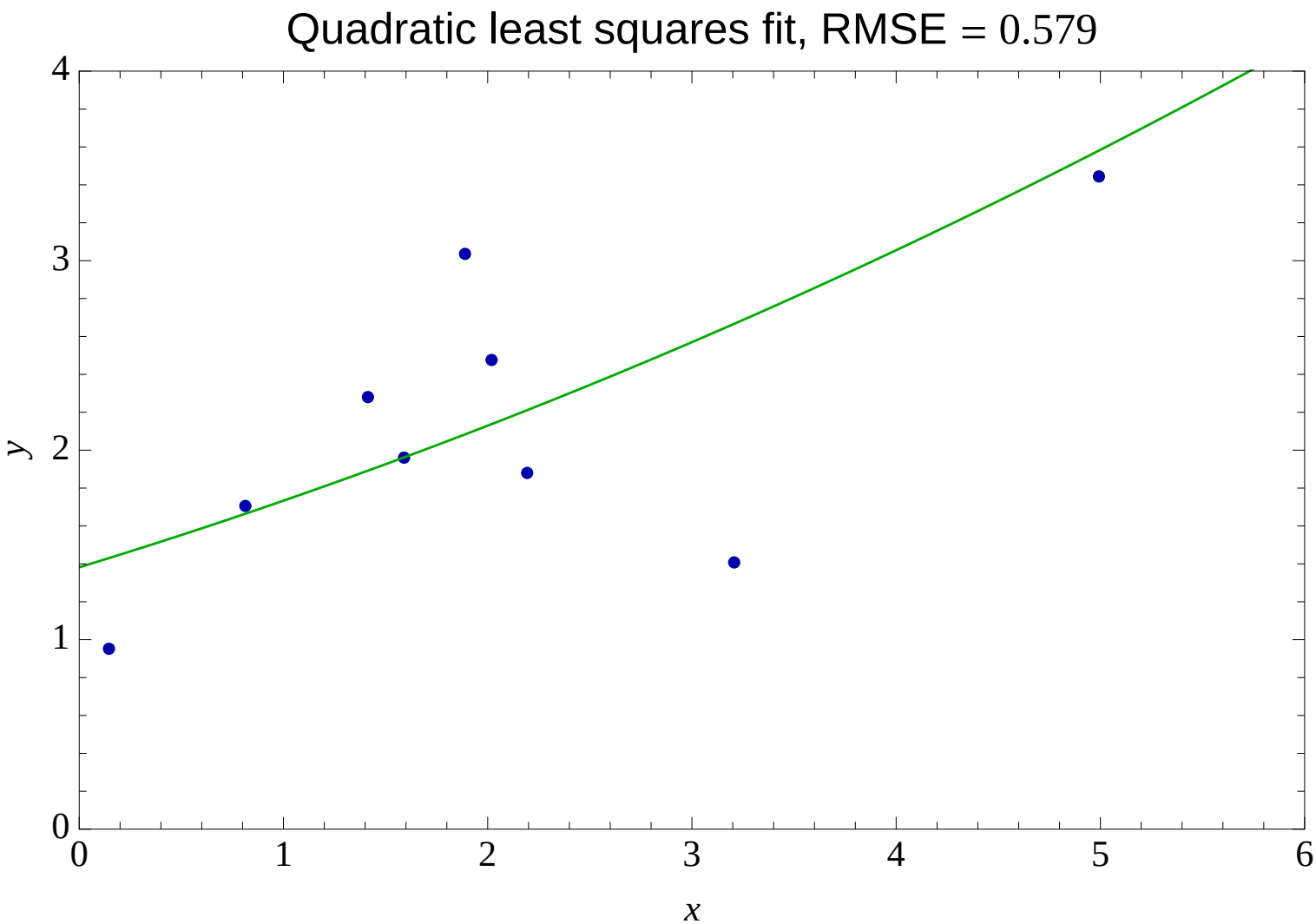
Outline

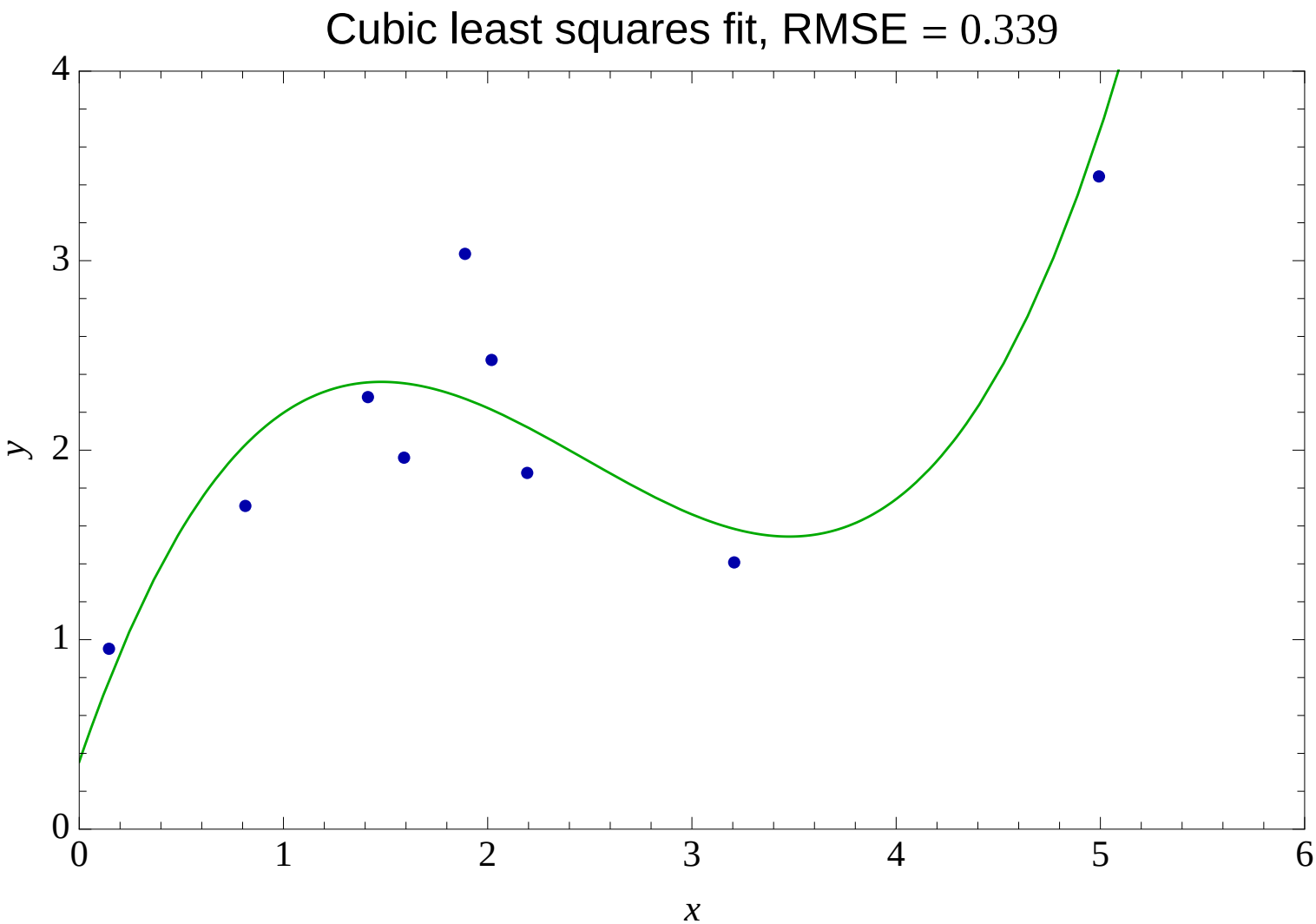
- Non-parametric fitting: two simple examples
- The formal model for classification, learning principles
- Classification algorithms (from a user's point of view)
  - perceptron, neural networks (NN)
  - AdaBoost
  - the Support Vector Machine (SVM)
- Machine learning research motivated by HEP applications

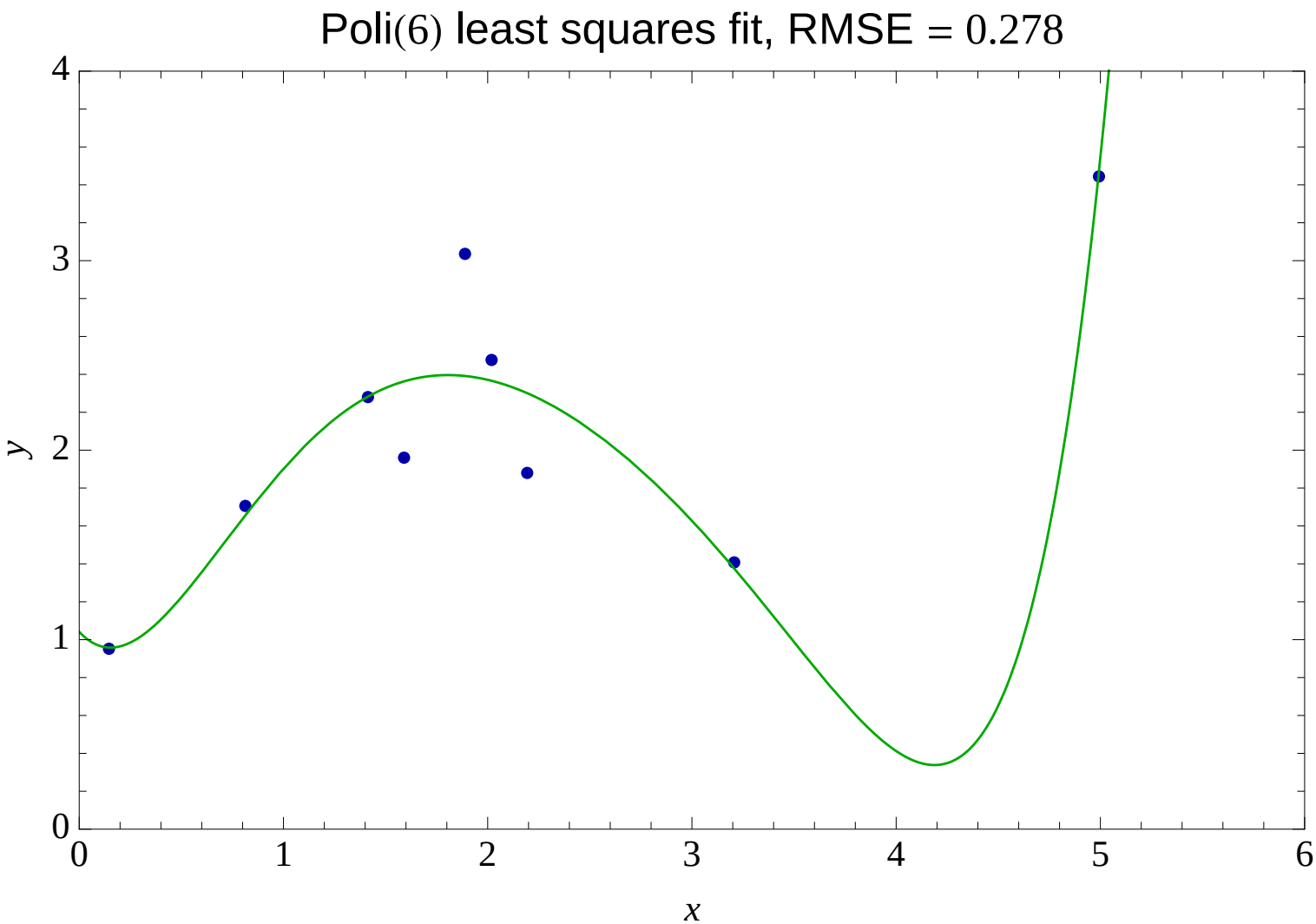

Data generated from an unknown function with unknown noise
y
x
0
1
2
3
4
0
1
2
3
4
5
6

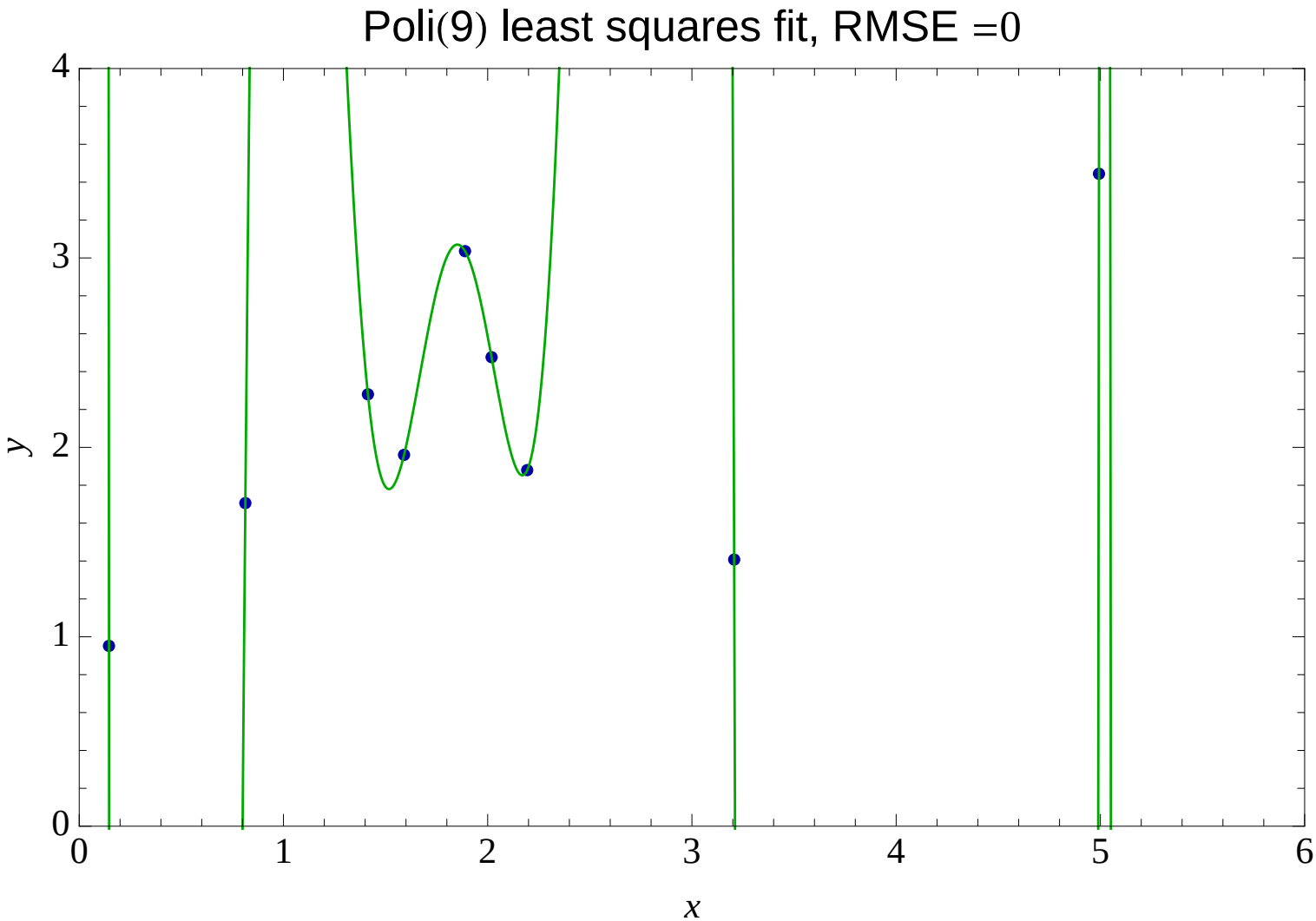

Constant least squares fit, RMSE = 0.915
y
x
0
1
2
3
4
5
6


Linear least squares fit, RMSE = 0.581
y
x
0
1
2
3
4
0
1
2
3
4
5
6


Quadratic least squares fit, RMSE = 0.579
y
x
0
1
2
3
4
5
6


Cubic least squares fit, RMSE = 0.339
y
x
0
1
2
3
4
5
6


Poli(6) least squares fit, RMSE = 0.278
y
x
0
1
2
3
4
5
6


Poli(9) least squares fit, RMSE =0
y
x
0
1
2
3
4
0
1
2
3
4
5
6

Non-parametric fitting

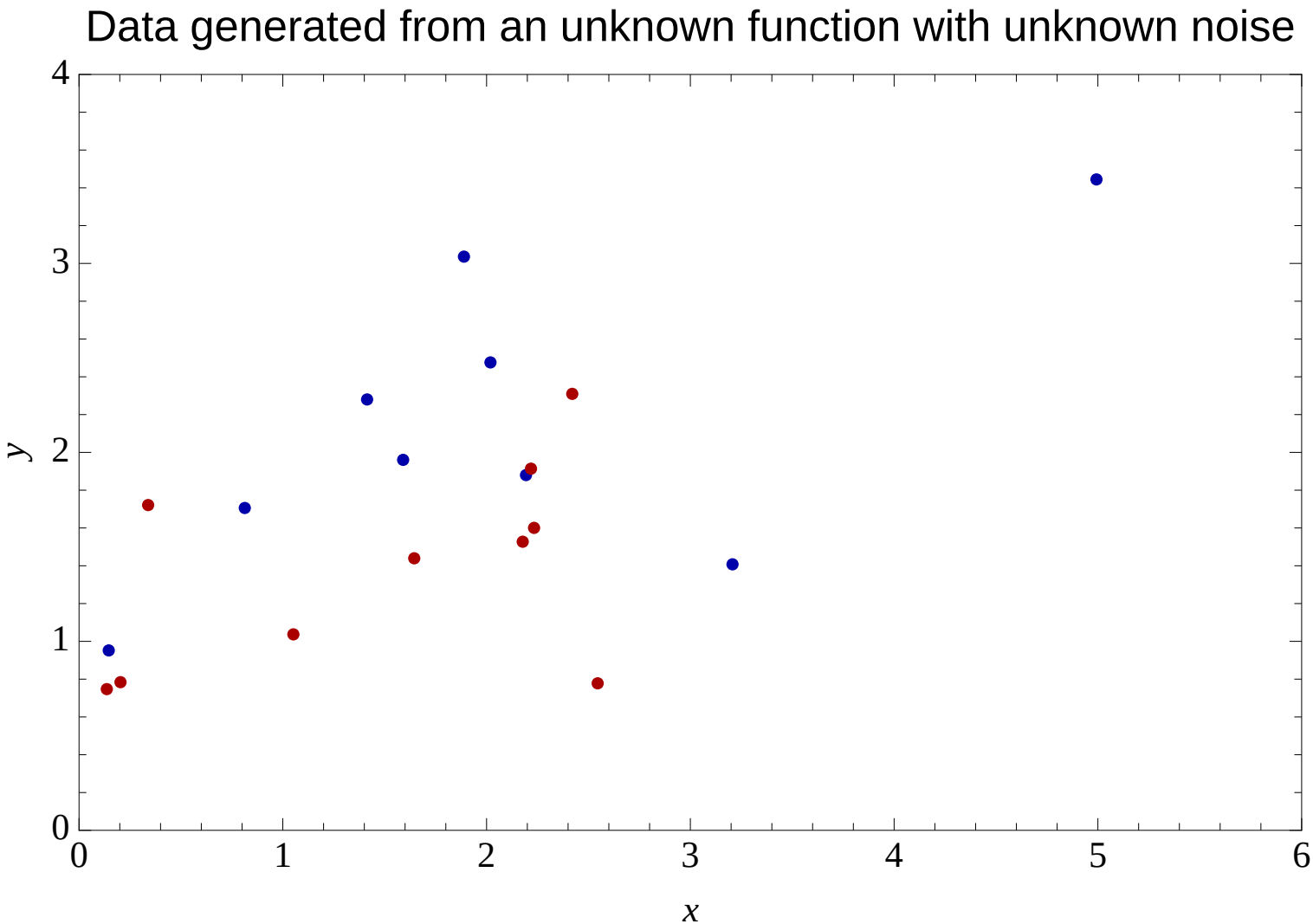
- Minimizing the training cost (RMSE) does not work if the function class is not fixed beforehand

- If you do not know the correct function class, you should not fix it

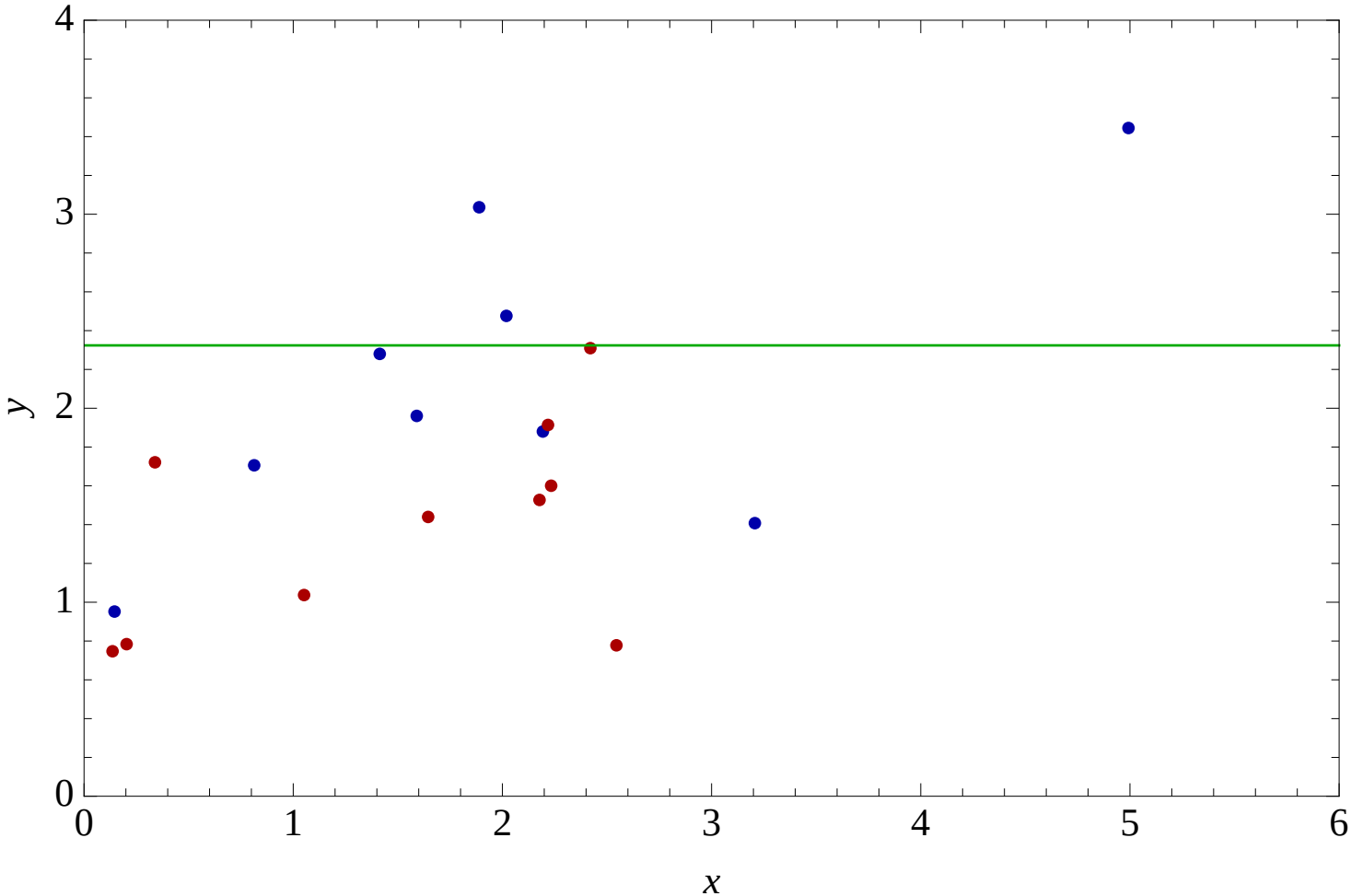
- Capacity control, regularization

  - theoretical results, data-independent, based on sample size and measures of complexity

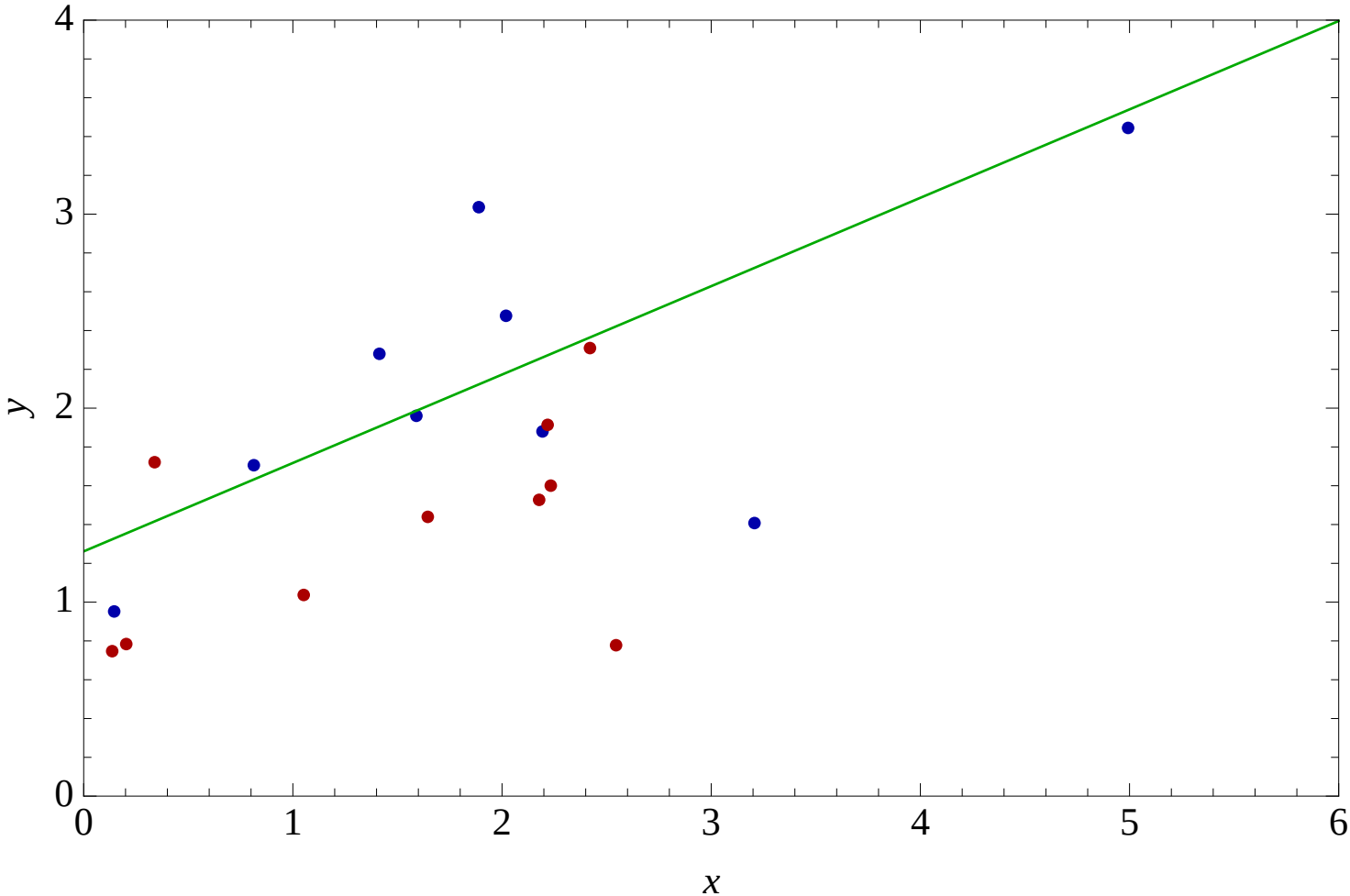
  - use independent (test) set


Data generated from an unknown function with unknown noise
y
x
0
1
2
3
4
0
1
2
3
4
5
6

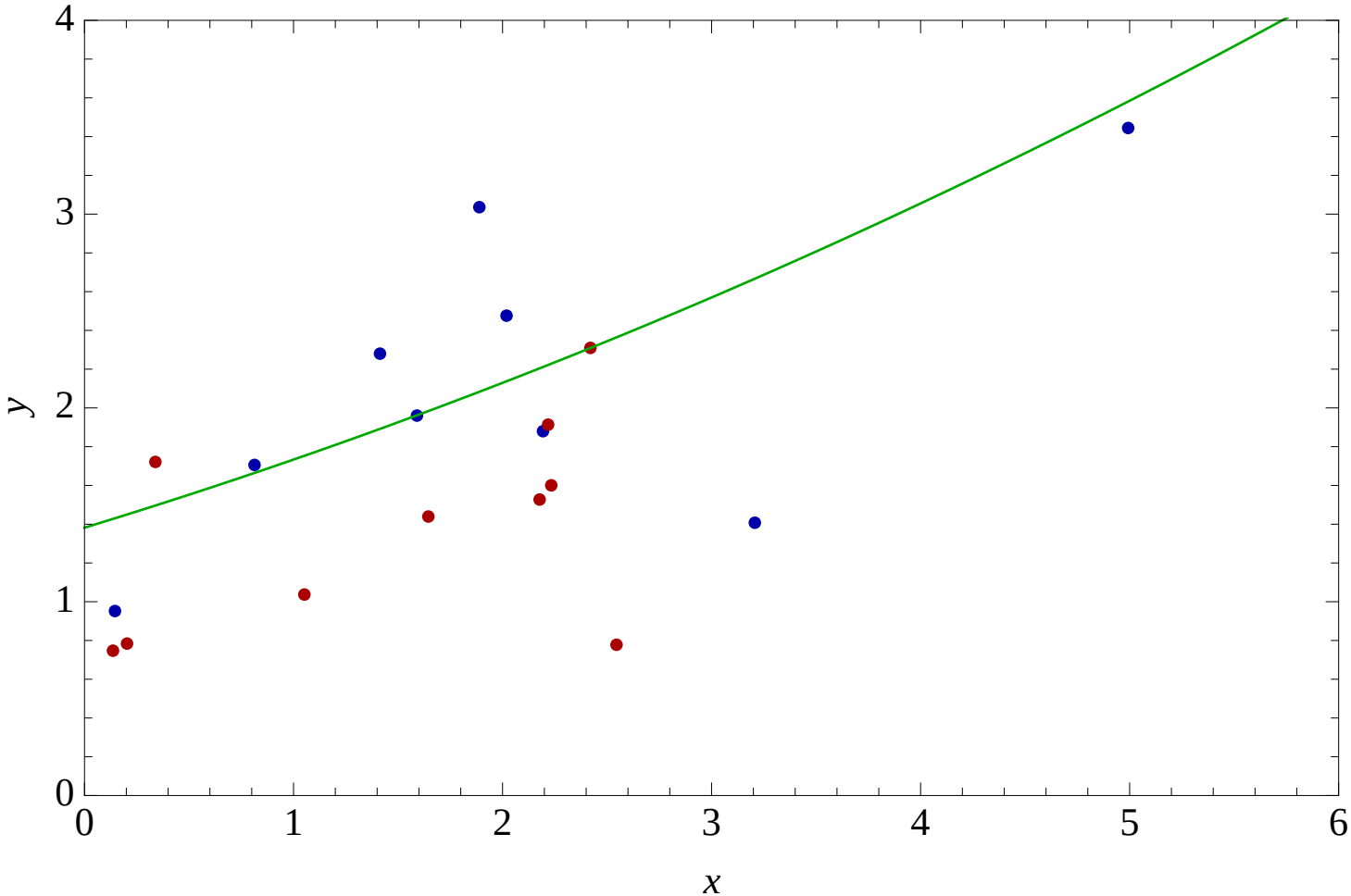
Const. least squares fit, training RMSE = 0.915, test RMSE = 1.067



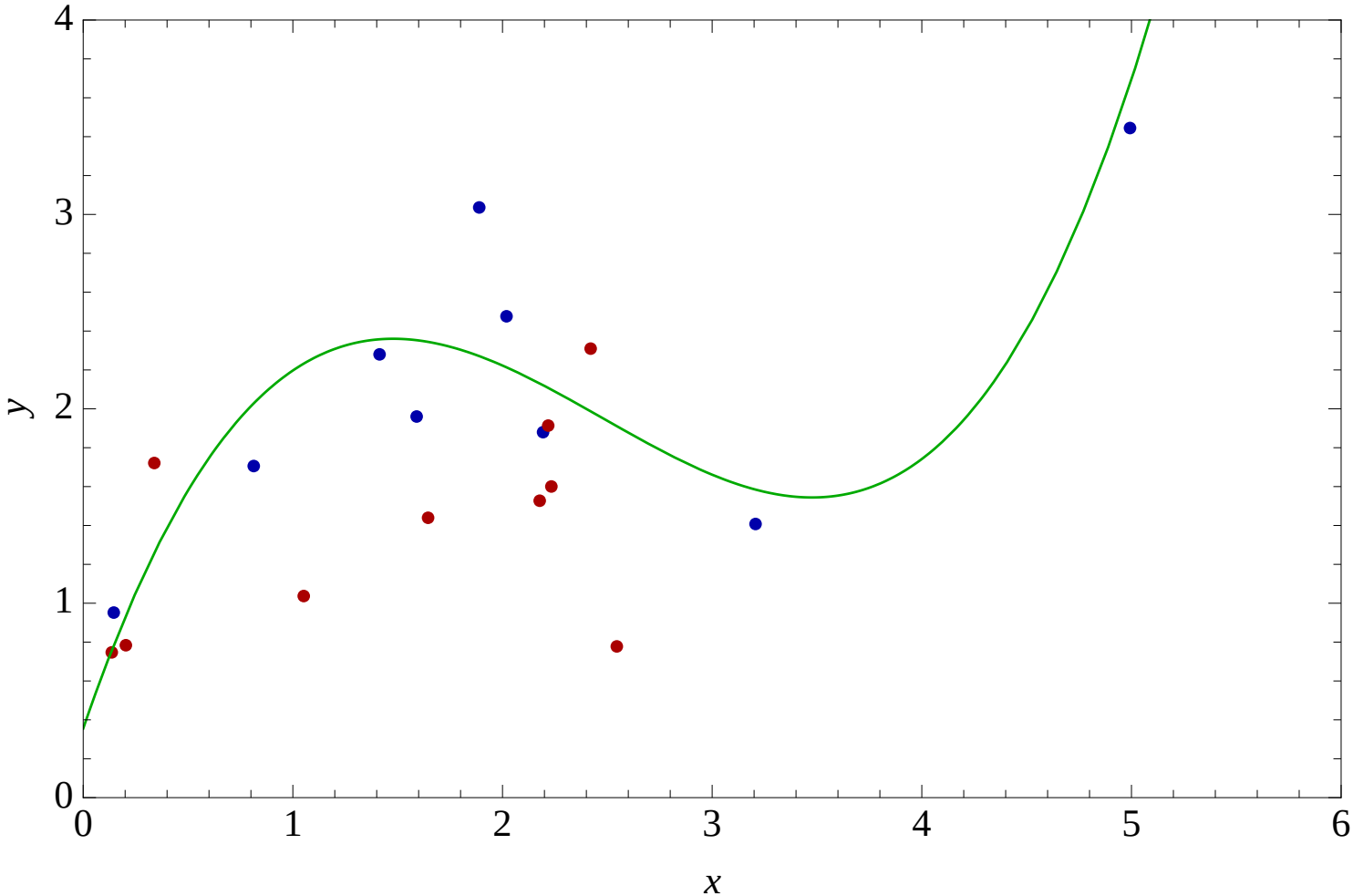
Linear least squares fit, training RMSE = 0.581, test RMSE = 0.734



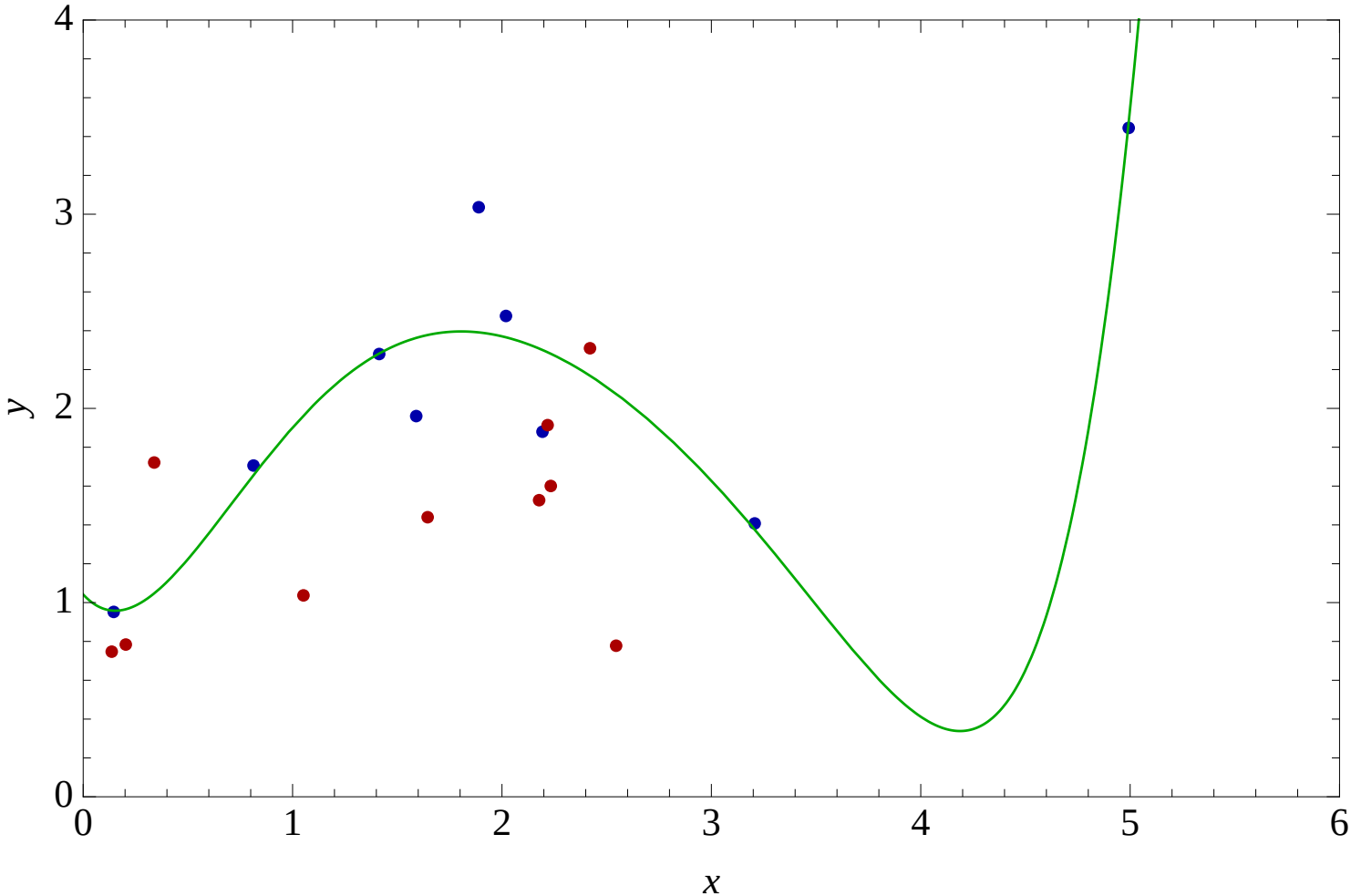
Quadr. least squares fit, training RMSE = 0.579, test RMSE = 0.723

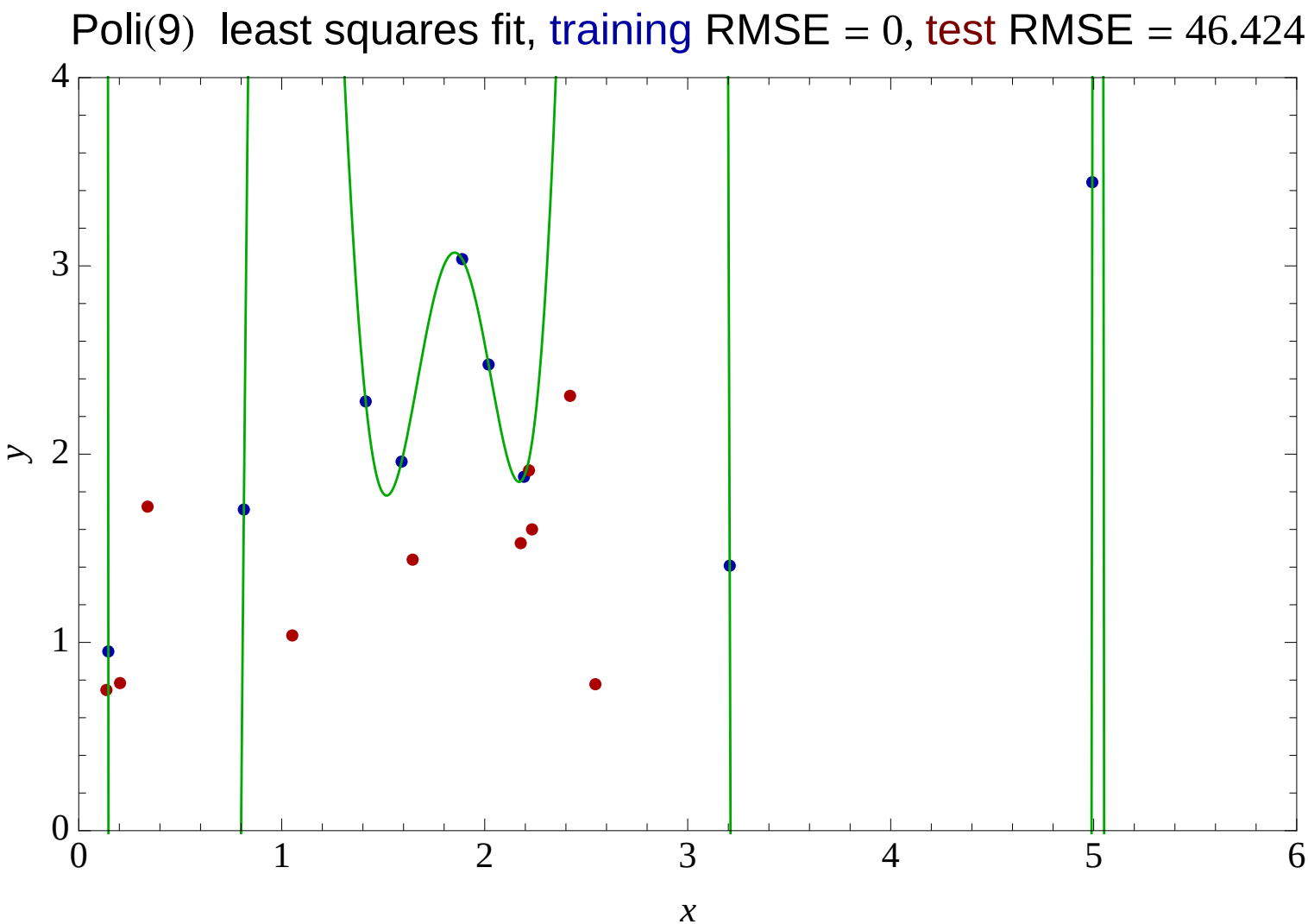


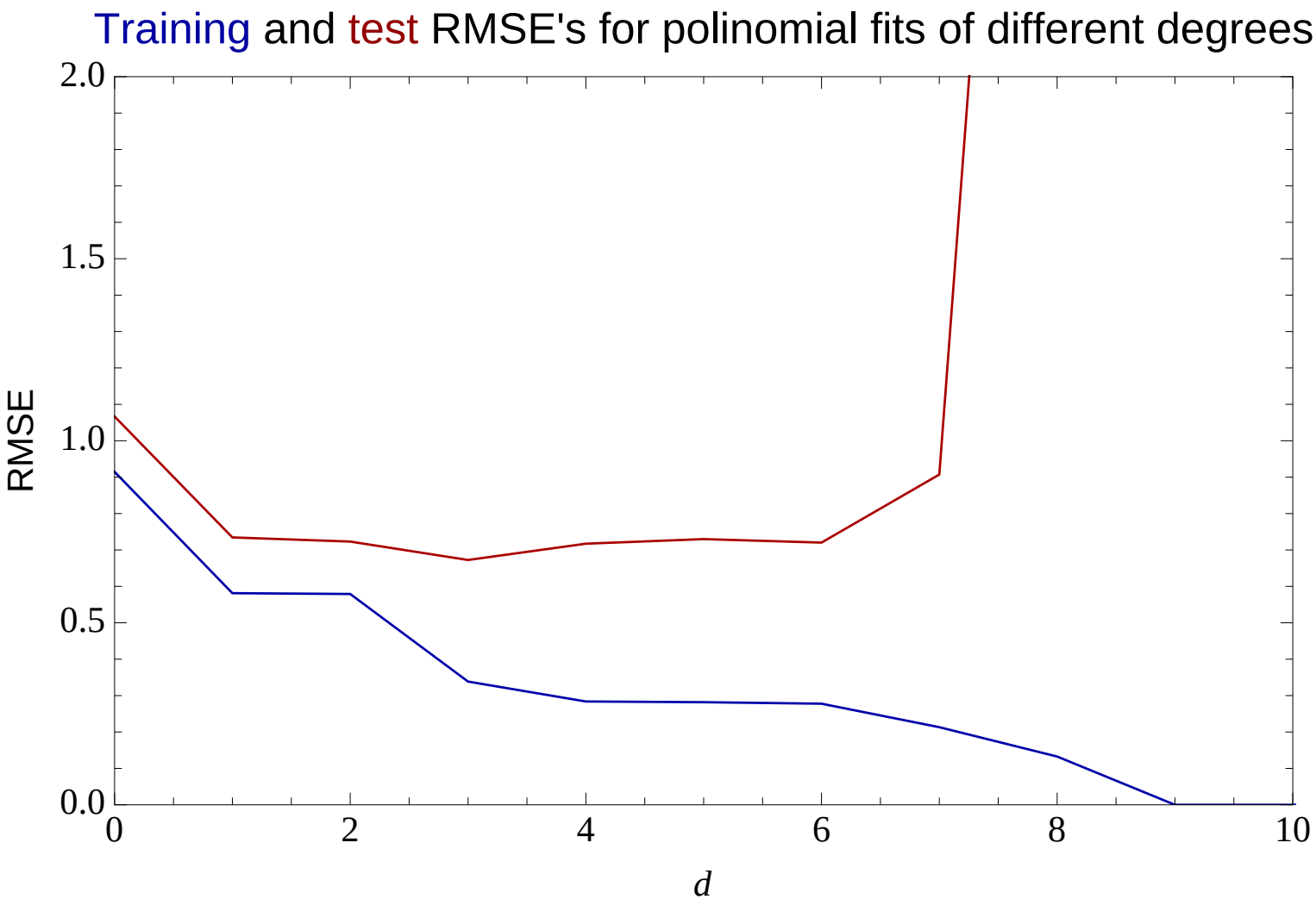
Cubic least squares fit, training RMSE = 0.339, test RMSE = 0.672

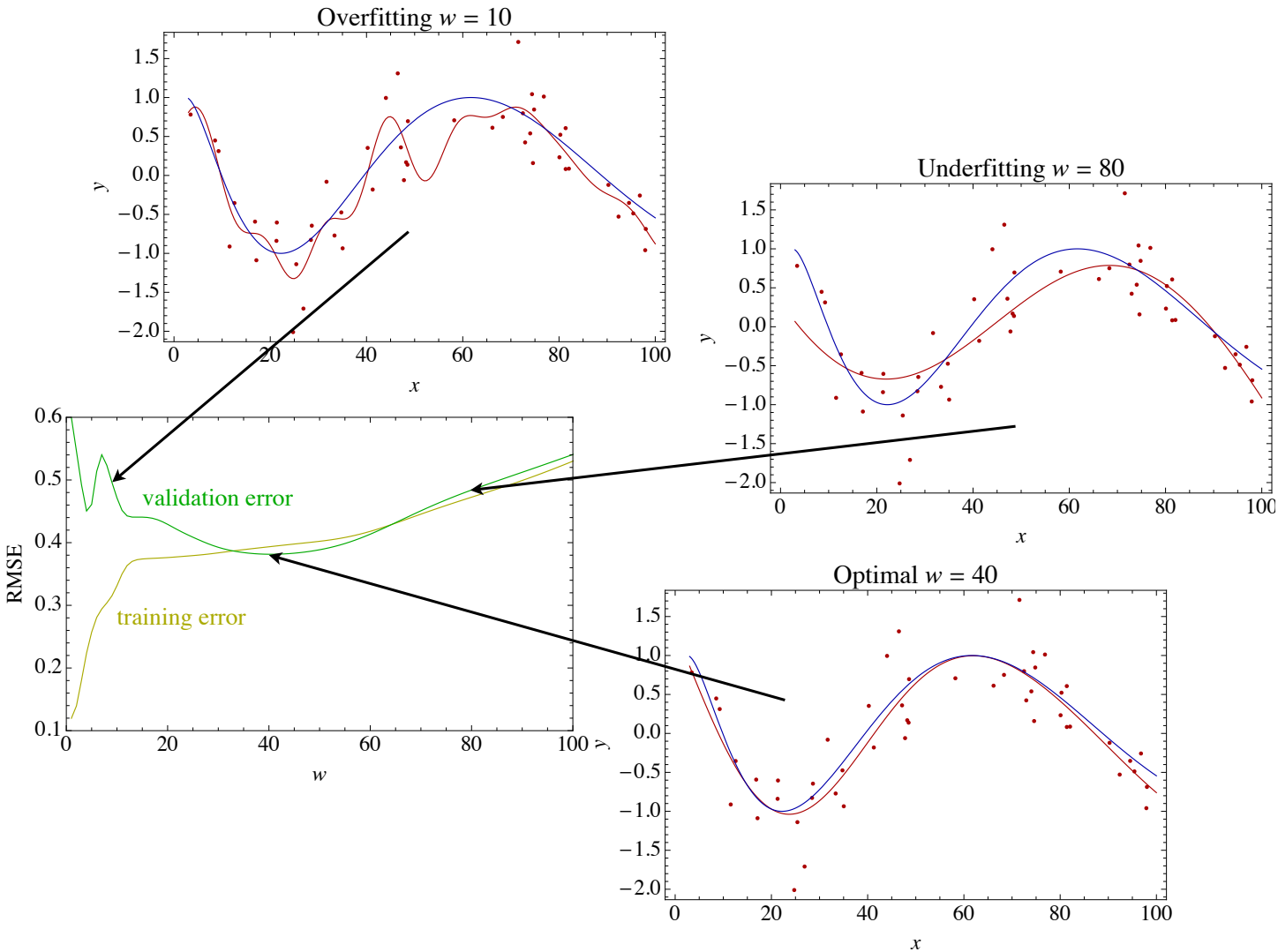


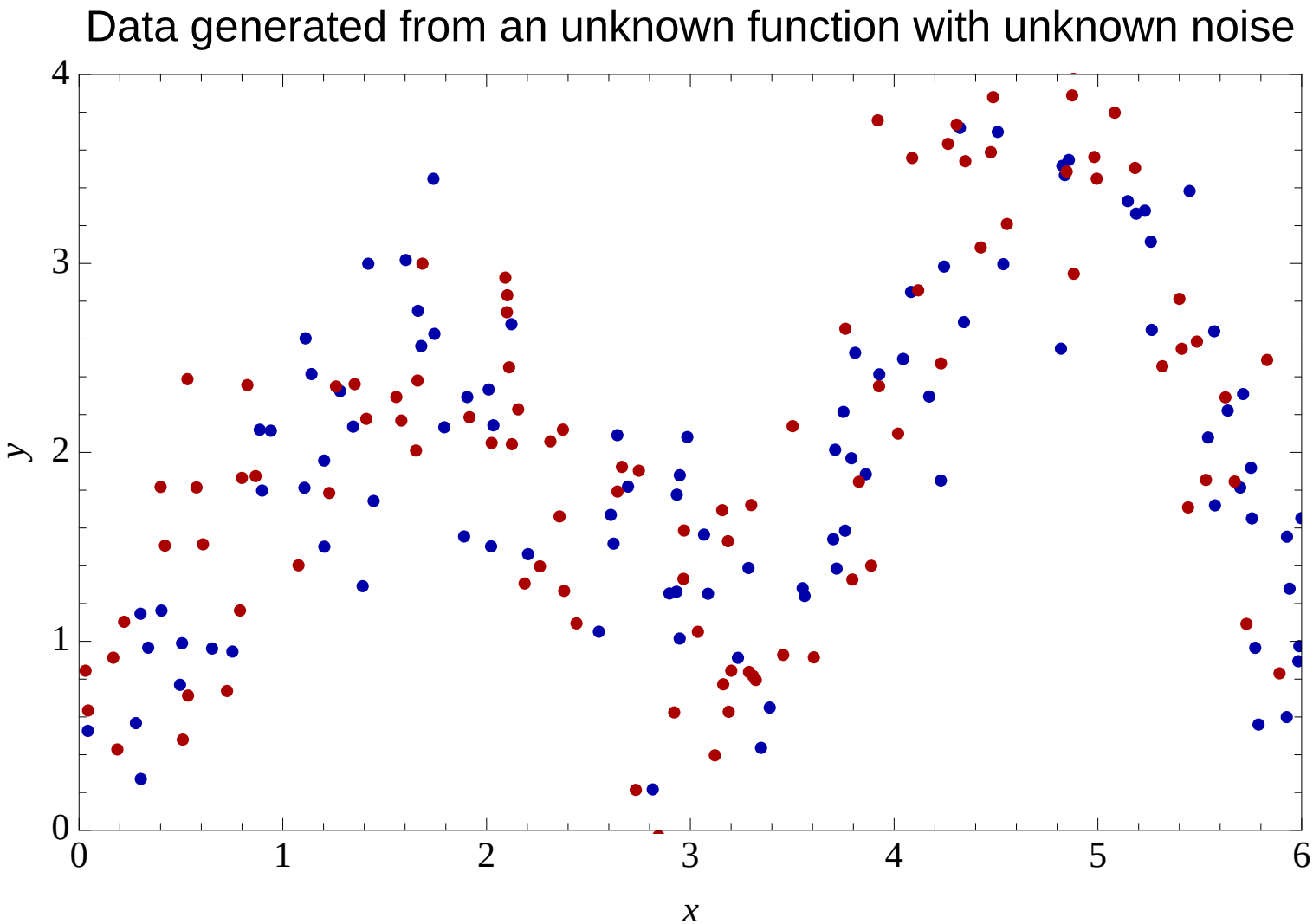
Poli(6) least squares fit, training RMSE = 0.278, test RMSE = 0.72

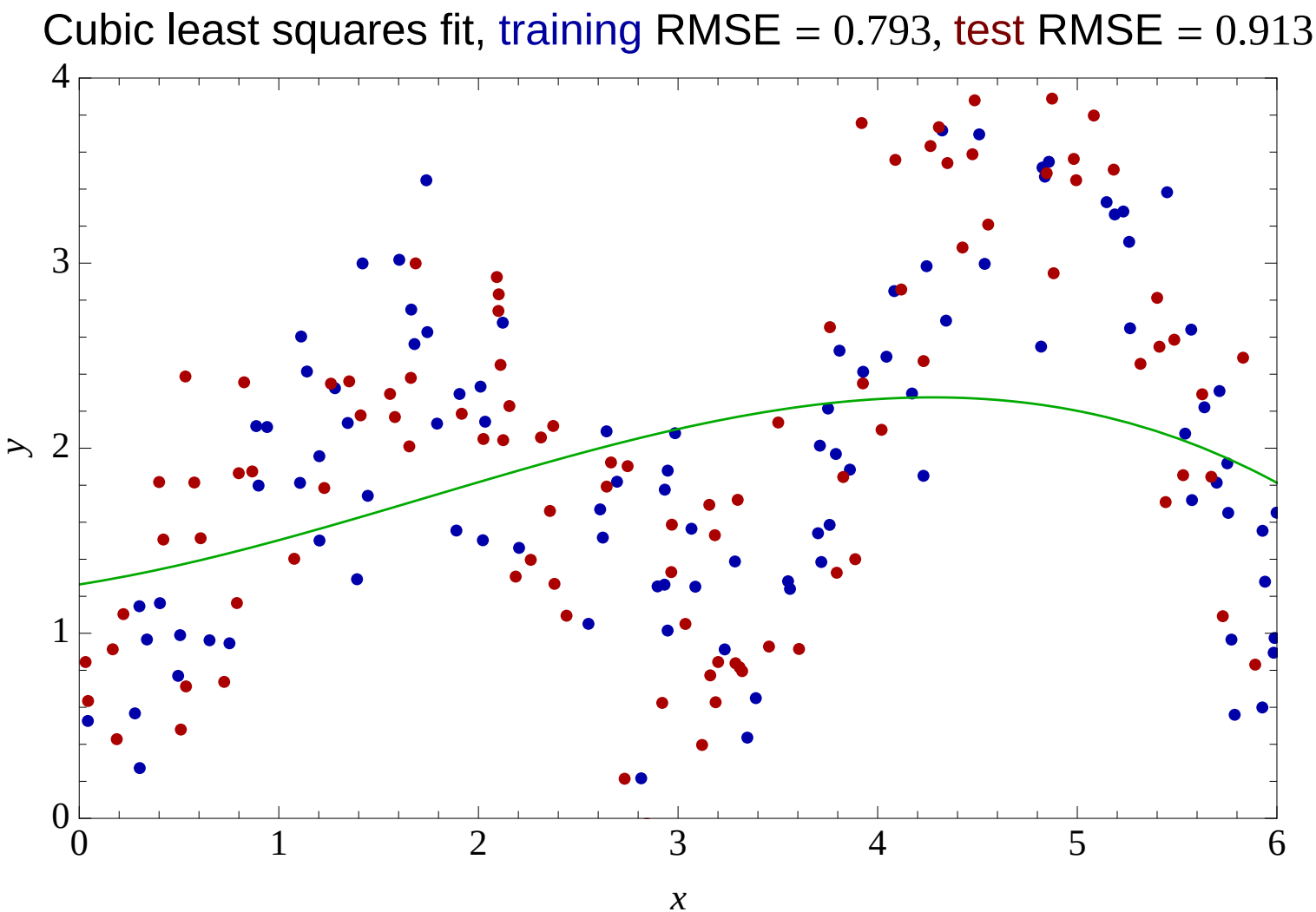



Poli(9) least squares fit, training RMSE = 0, test RMSE = 46.424
y
x
0
1
2
3
4
5
6

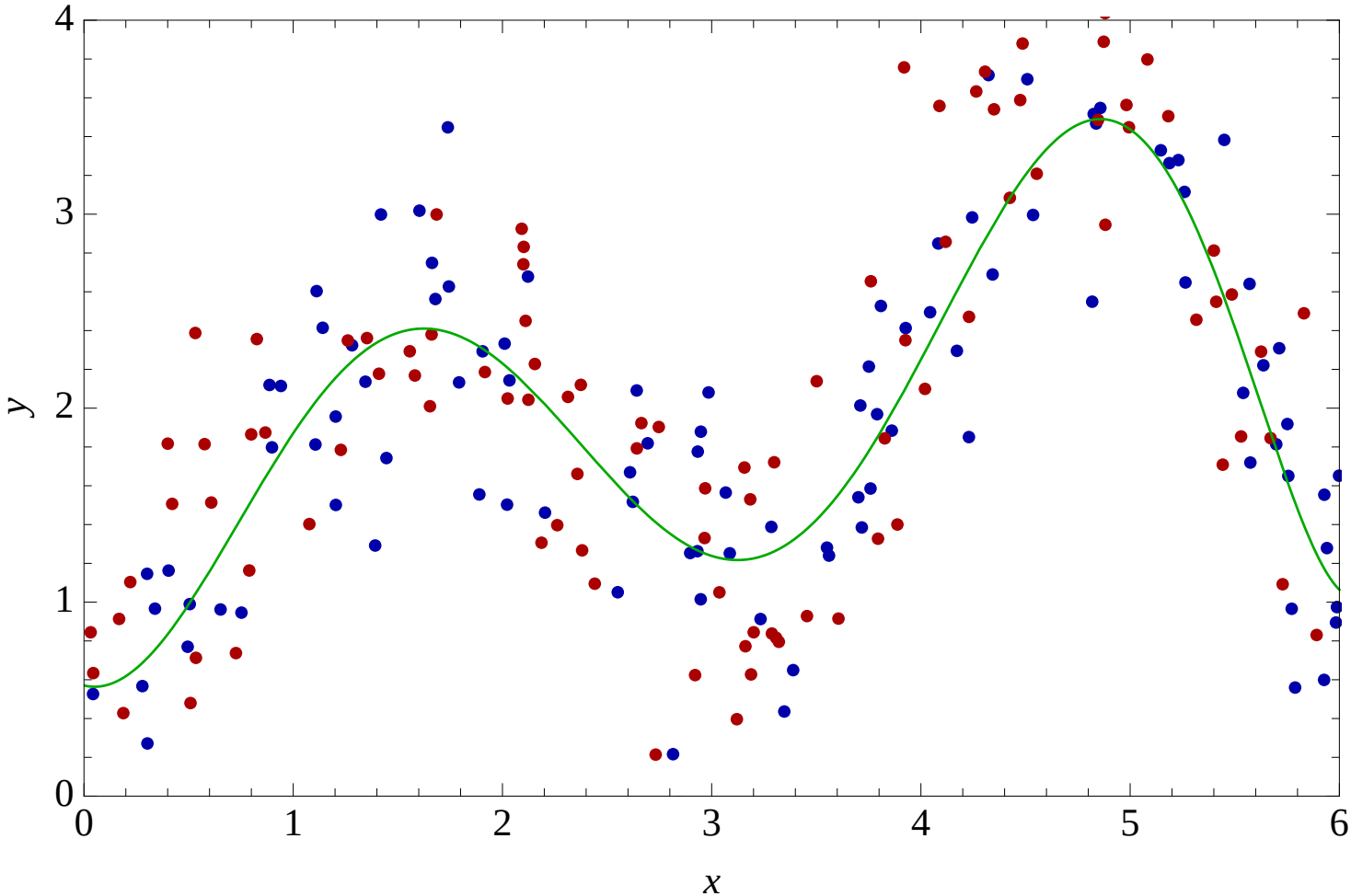

Training and test RMSE's for polinomial fits of different degrees
RMSE
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
d


Overfitting w = 10
Underfitting w = 80
Optimal w = 40
validation error
training error
RMSE
w
x
y

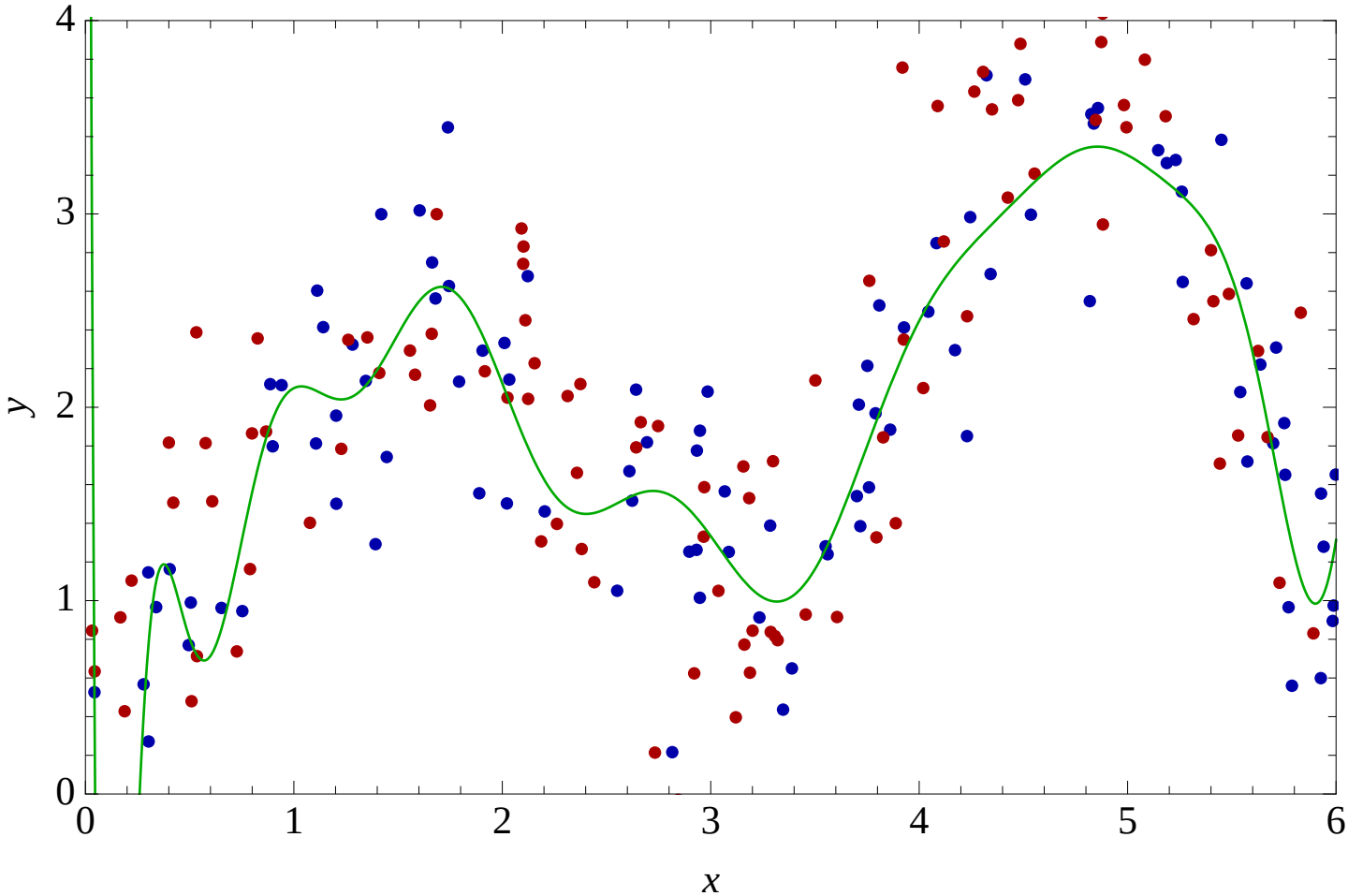

Data generated from an unknown function with unknown noise
y
x
0
1
2
3
4
0
1
2
3
4
5
6

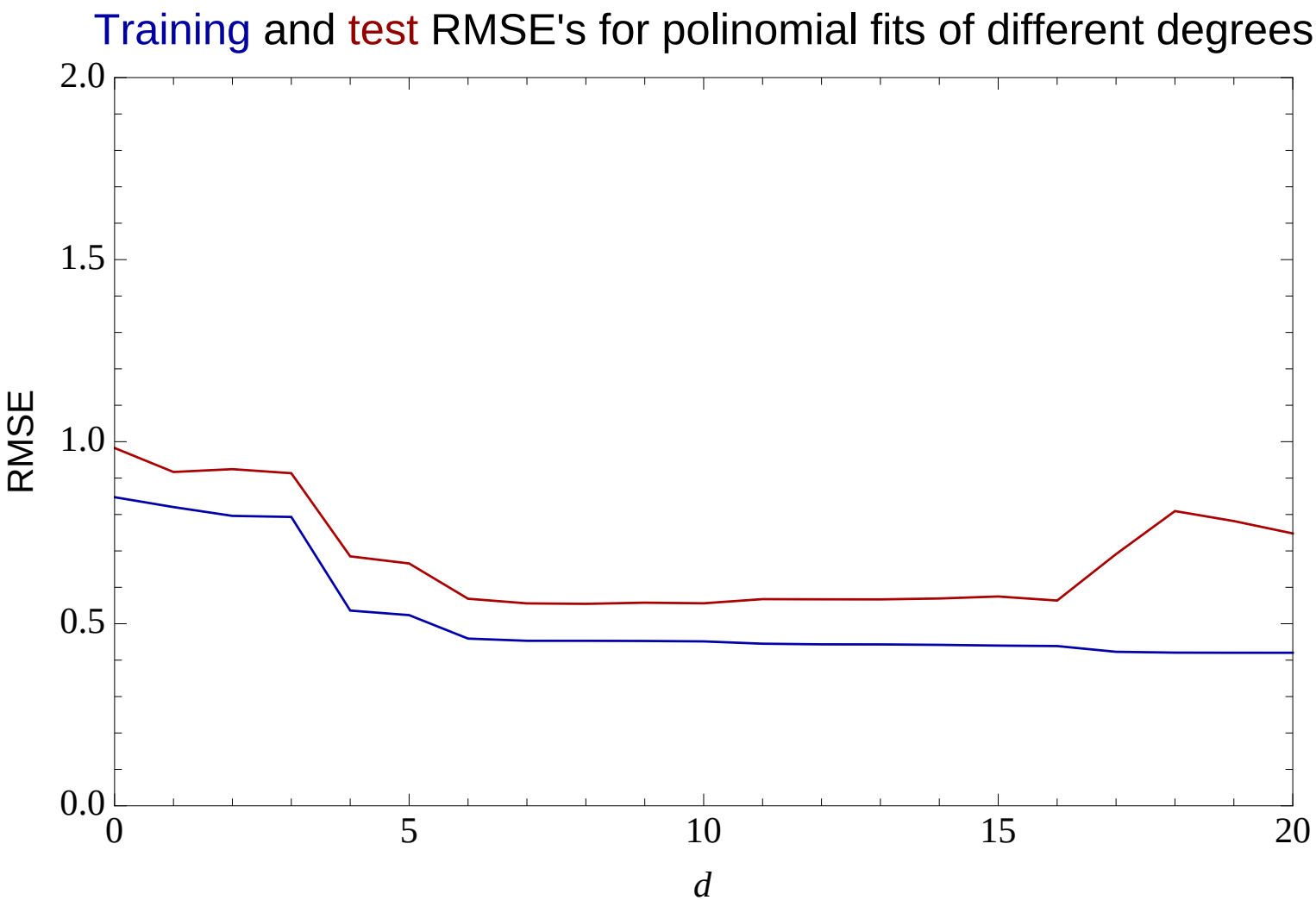

Cubic least squares fit, training RMSE = 0.793, test RMSE = 0.913
y
x
0
1
2
3
4
5
6

Poli(8) least squares fit, training RMSE = 0.453, test RMSE = 0.555



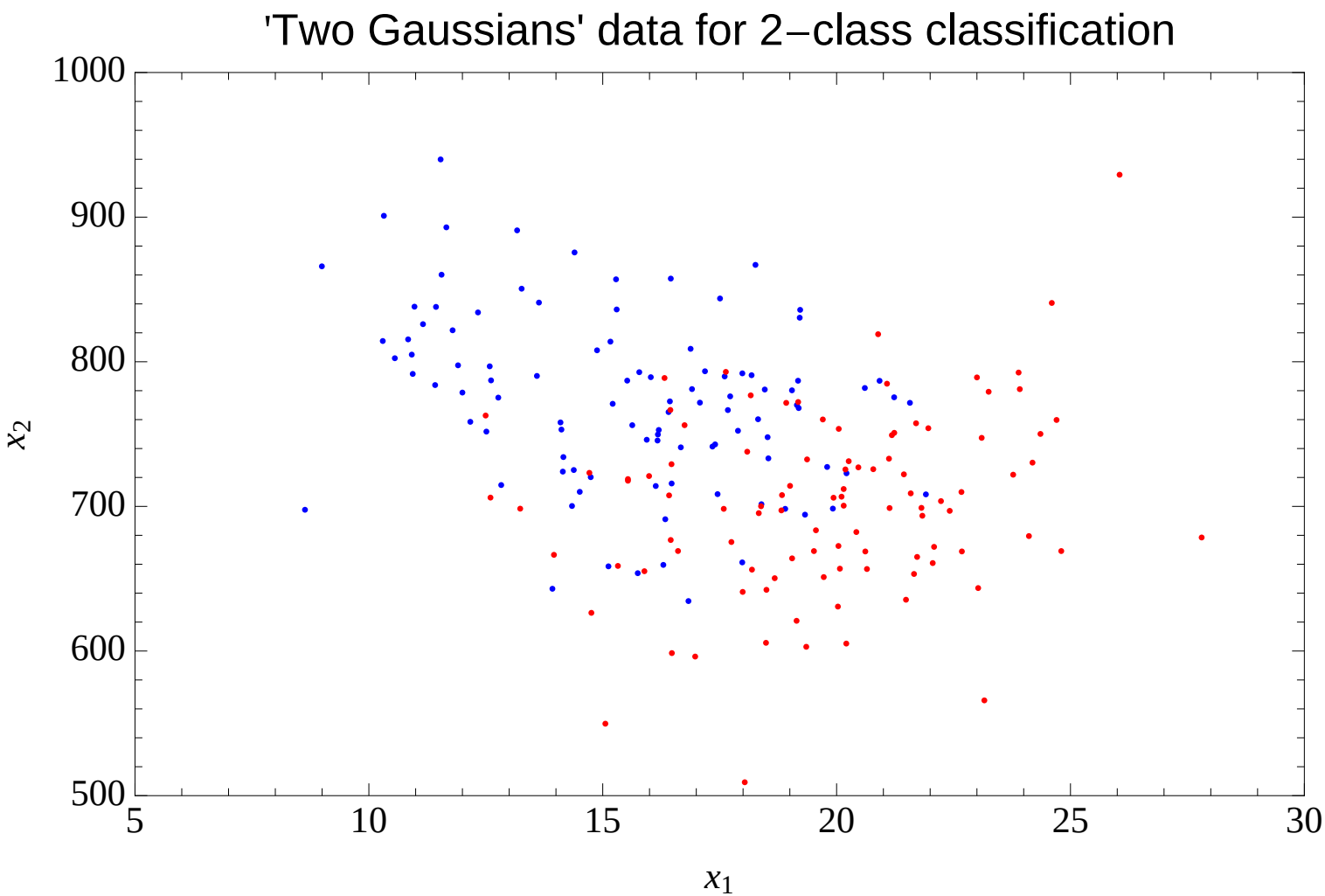
Poli(18) least squares fit, training RMSE = 0.421, test RMSE = 0.809

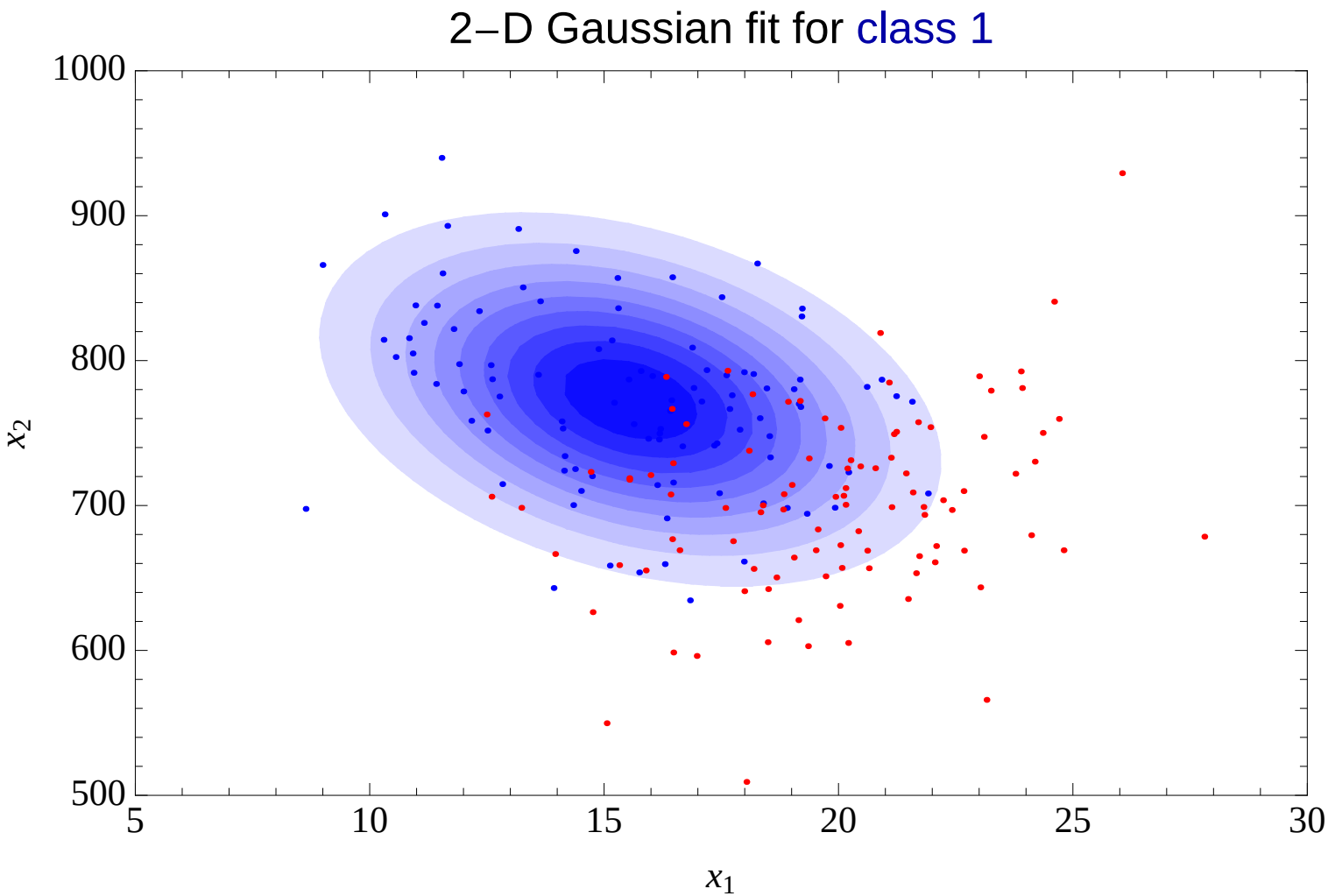


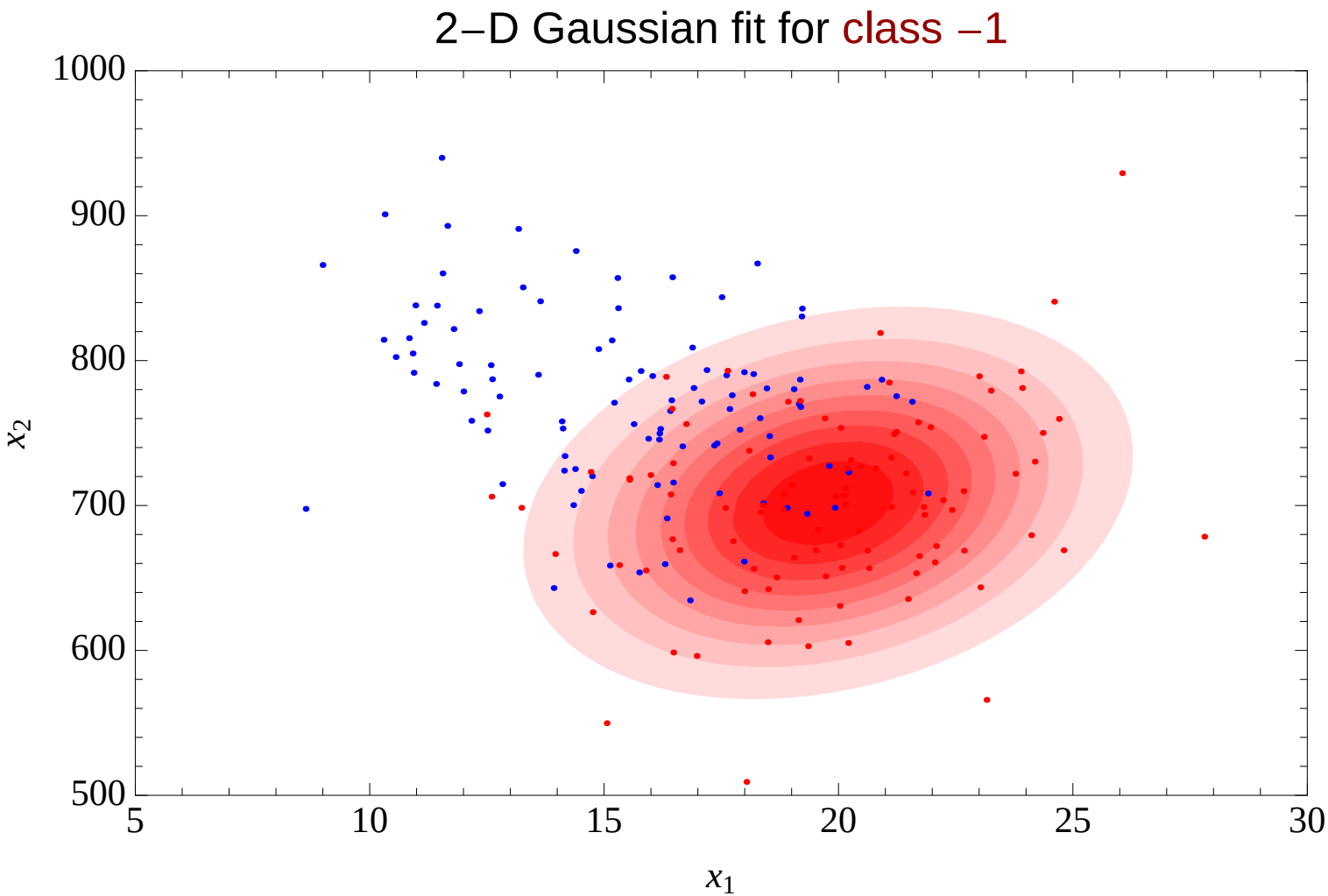

Training and test RMSE's for polinomial fits of different degrees
RMSE
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
d

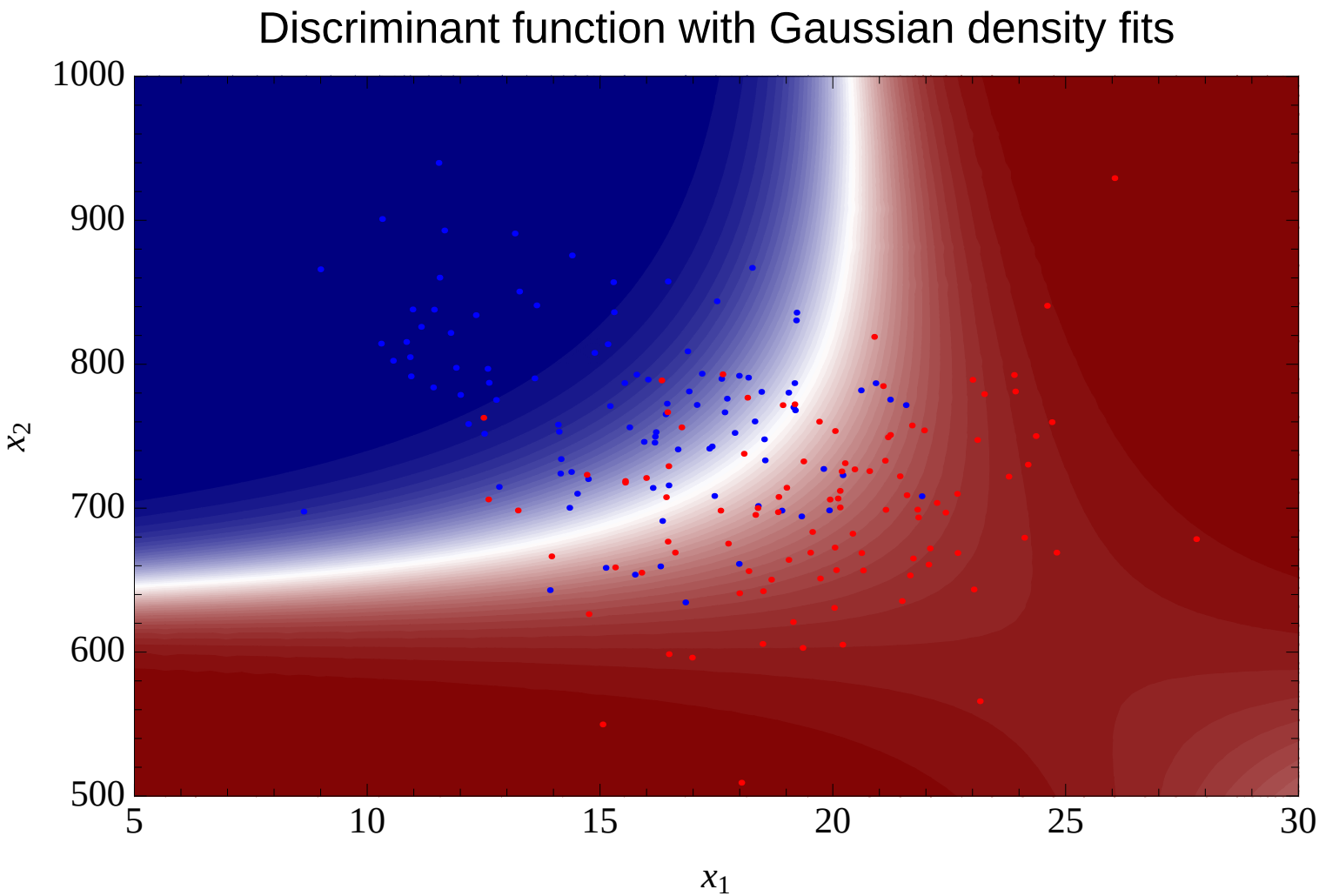
Non-parametric fitting

- Capacity control, regularization
  - trade-off between approximation error and estimation error
  - complexity grows with data size
  - no need to correctly guess the function class


'Two Gaussians' data for 2−class classification
1000
900
800
700
600
500
5
10
15
20
25
30
x_1
x_2


2−D Gaussian fit for class 1
1000
900
800
700
600
500
5
10
15
20
25
30
x_1
x_2


2−D Gaussian fit for class −1
1000
900
800
700
600
500
5
10
15
20
25
30
x_1
x_2


Discriminant function with Gaussian density fits
1000
900
800
700
600
500
x_2
5
10
15
20
25
30
x_1

Outline

- Non-parametric fitting: two simple examples
- The formal model for classification, learning principles
- Classification algorithms (from a user's point of view)
  - perceptron, neural networks (NN)
  - AdaBoost
  - the Support Vector Machine (SVM)
- Machine learning research motivated by HEP applications

Classification

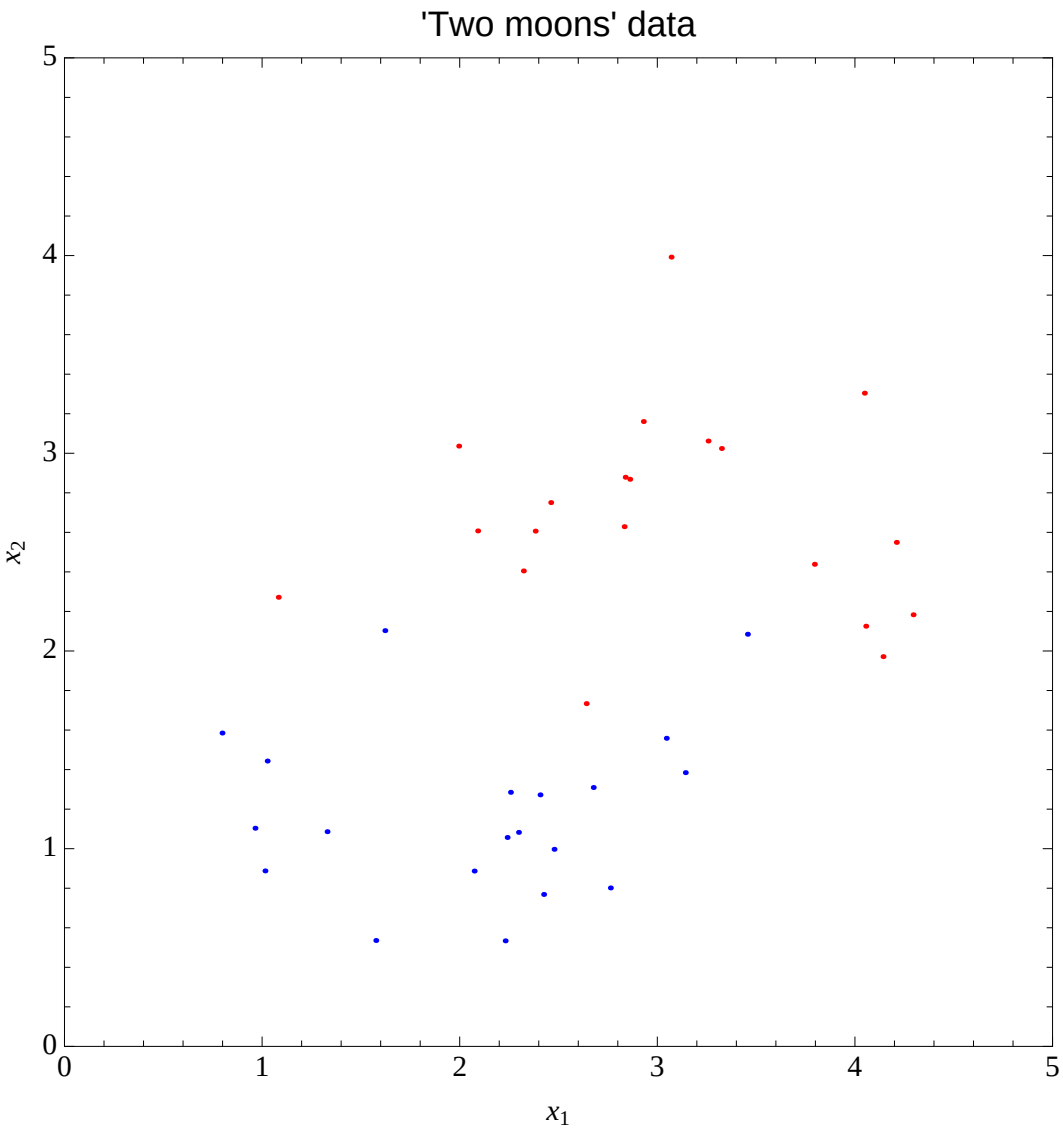
- Terminology
  - Conditional densities: $p(\mathbf{x} \mid Y = 1)$, $p(\mathbf{x} \mid Y = -1)$
  - Prior probabilities: $P(Y = 1)$, $P(Y = -1)$
  - Posterior probabilities: $P(Y = 1 \mid \mathbf{x})$, $P(Y = -1 \mid \mathbf{x})$

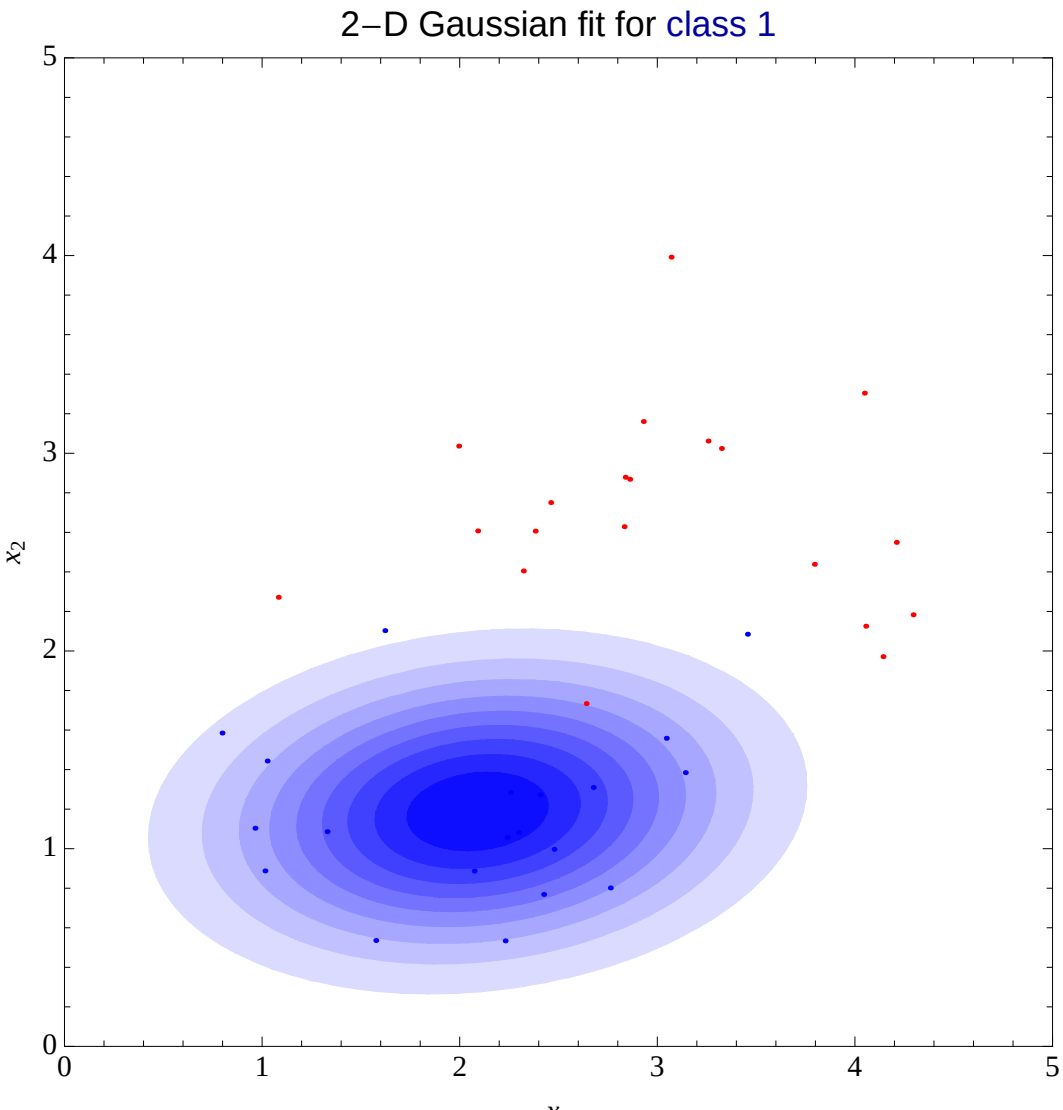
- Bayes theorem:

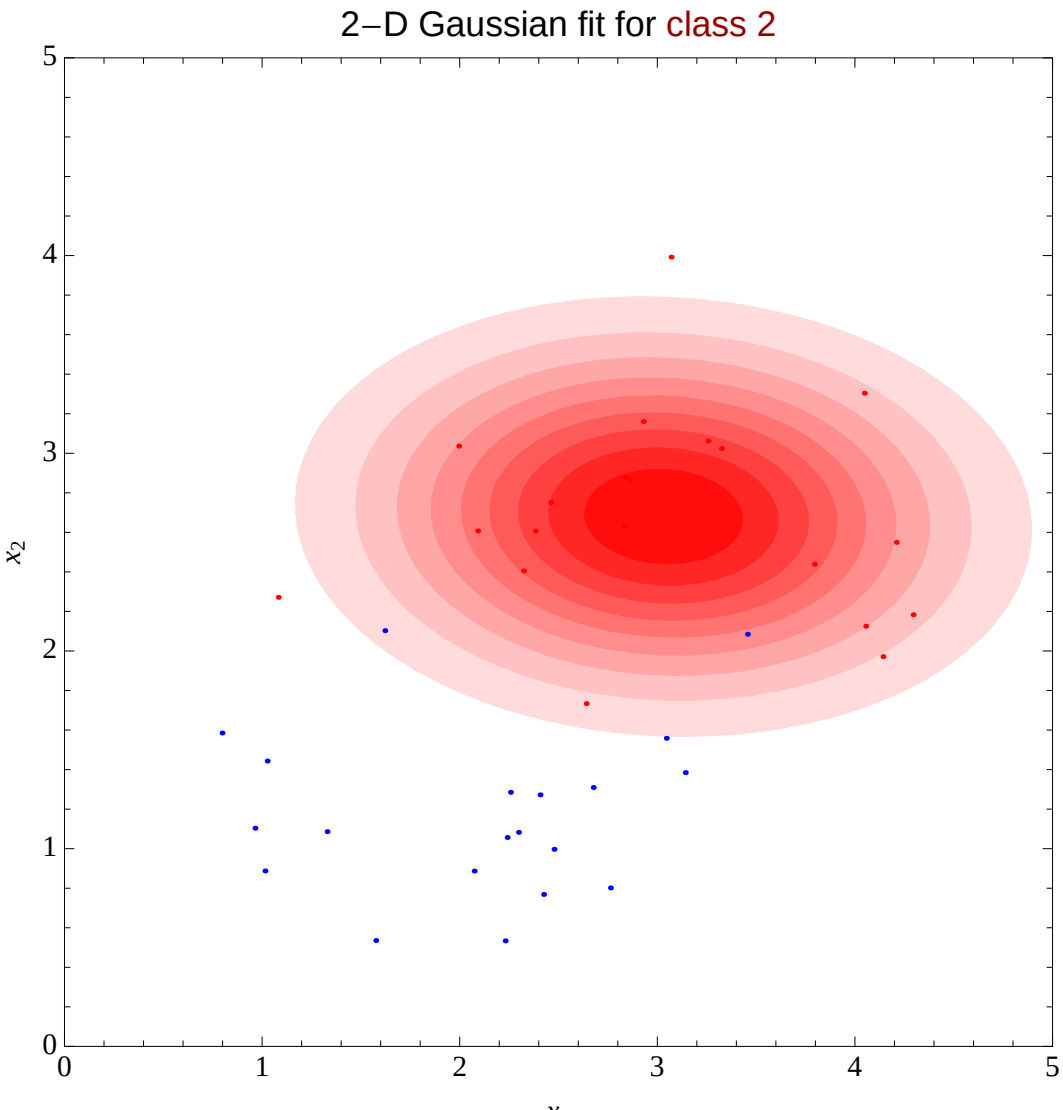
$$P(Y = 1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid Y = 1)P(Y = 1)}{p(\mathbf{x})} \propto p(\mathbf{x} \mid Y = 1)P(Y = 1)$$

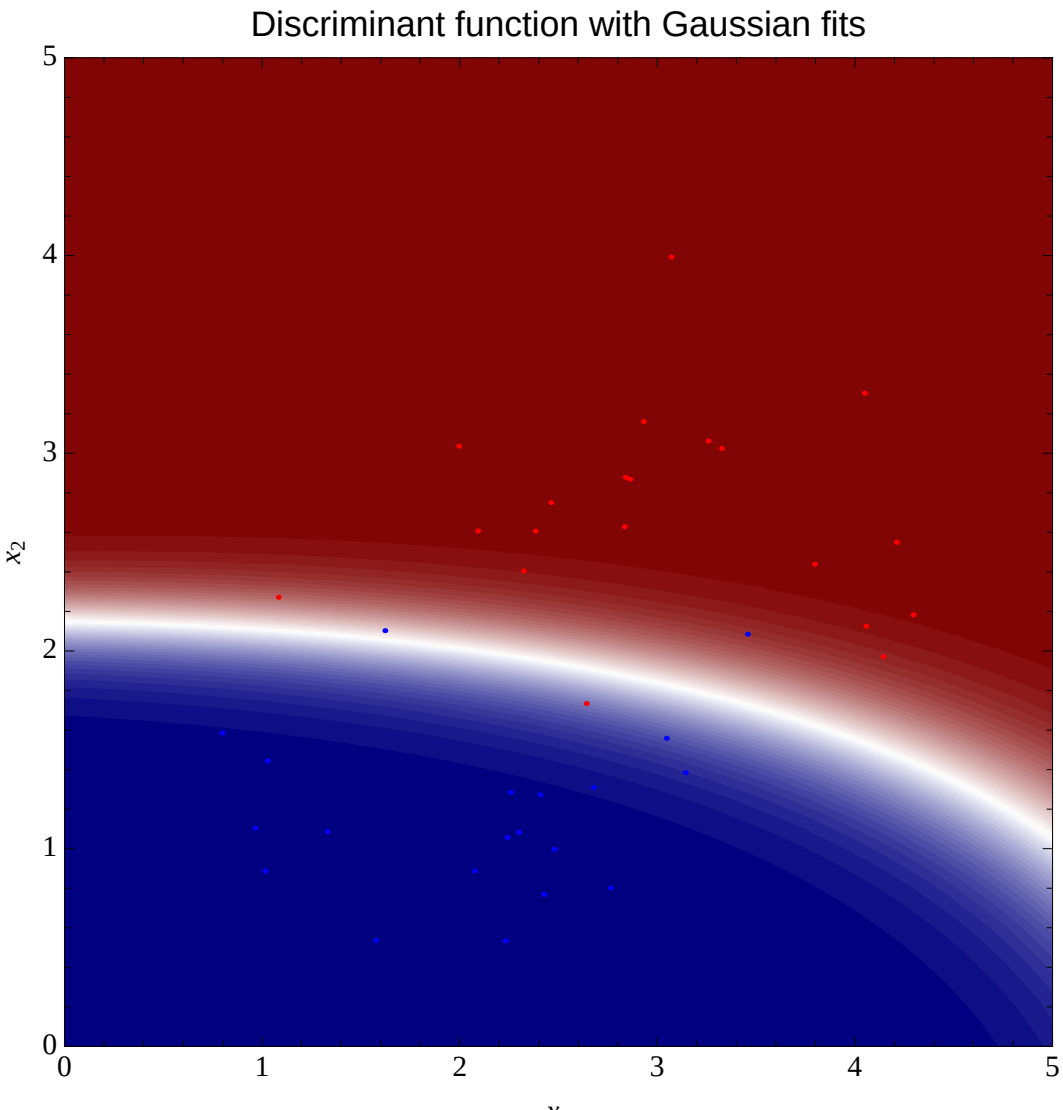
  - Decision:

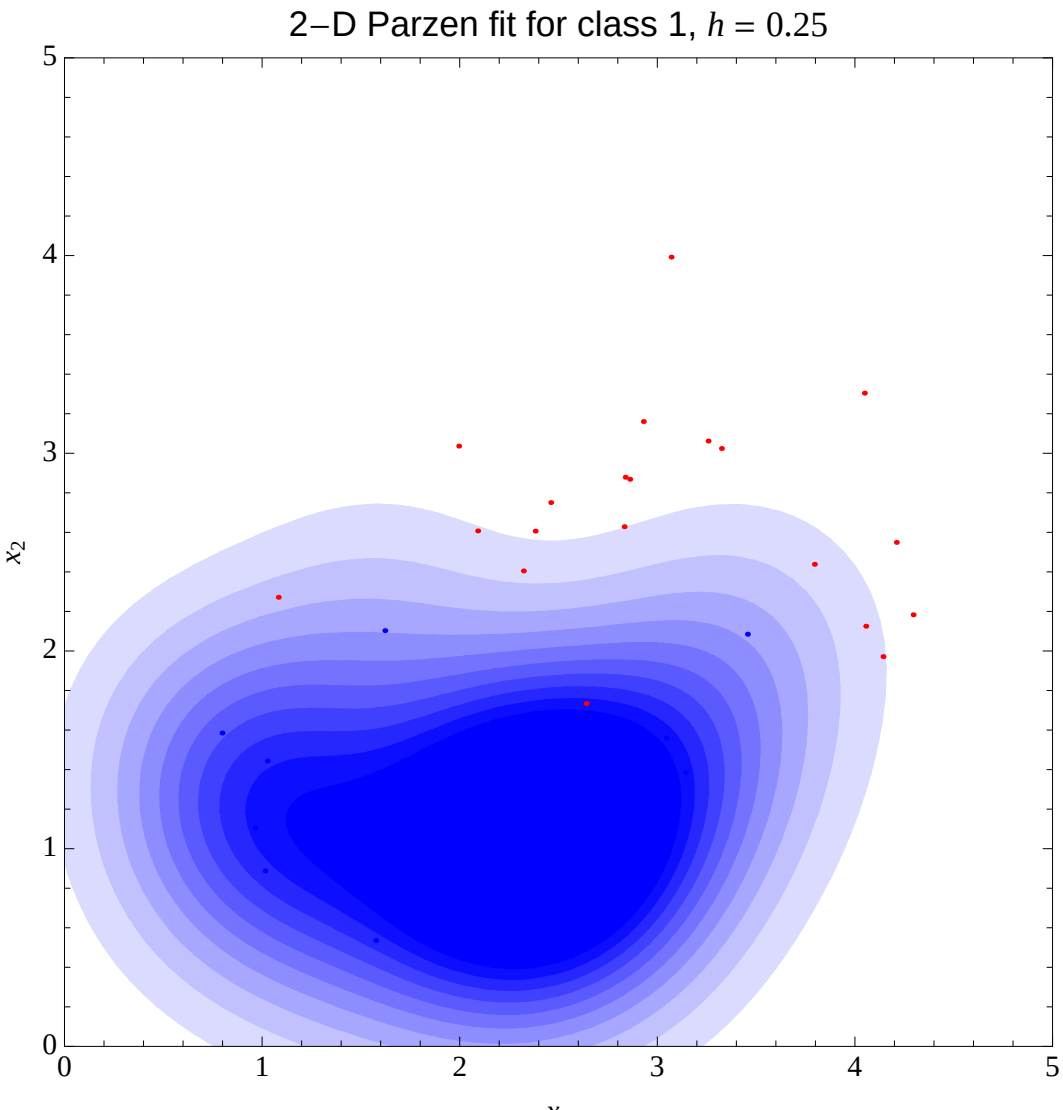
$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } \frac{p(\mathbf{x}|Y=1)P(Y=1)}{p(\mathbf{x}|Y=-1)P(Y=-1)} > 1, \\ -1 & \text{otherwise.} \end{cases}$$

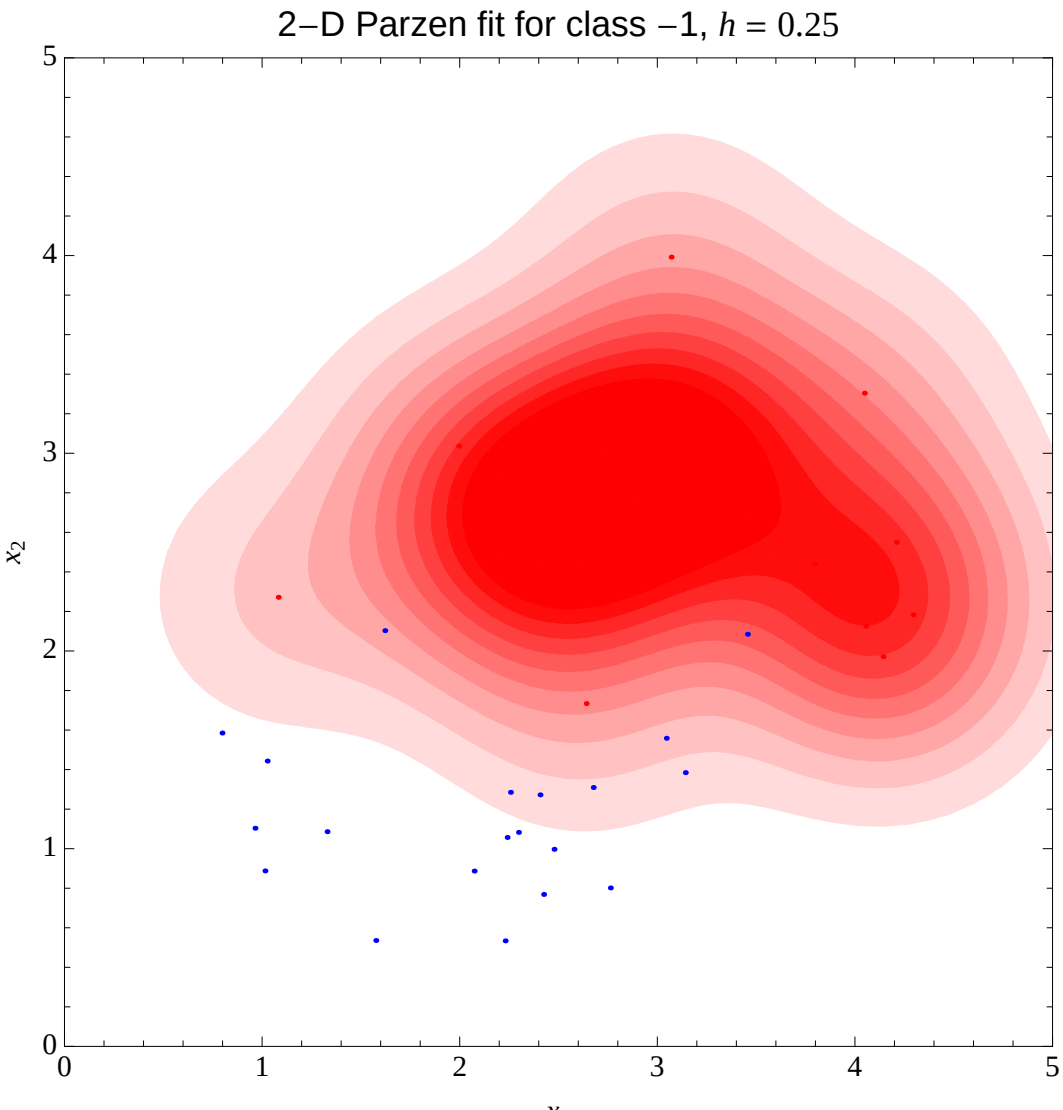

'Two moons' data
0
1
2
3
4
5
x_1
x_2

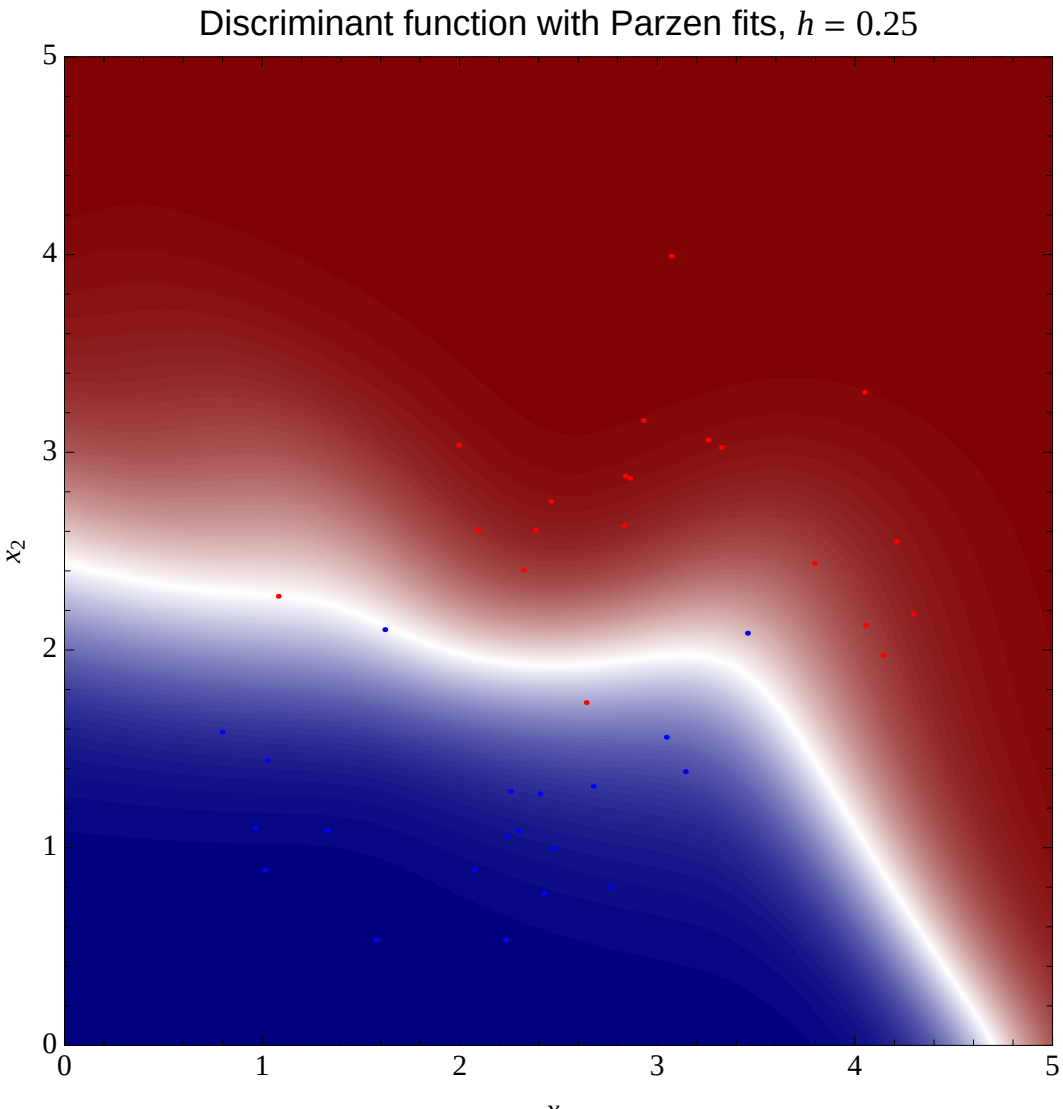

2−D Gaussian fit for class 1
0
1
2
3
4
5
x_1
x_2

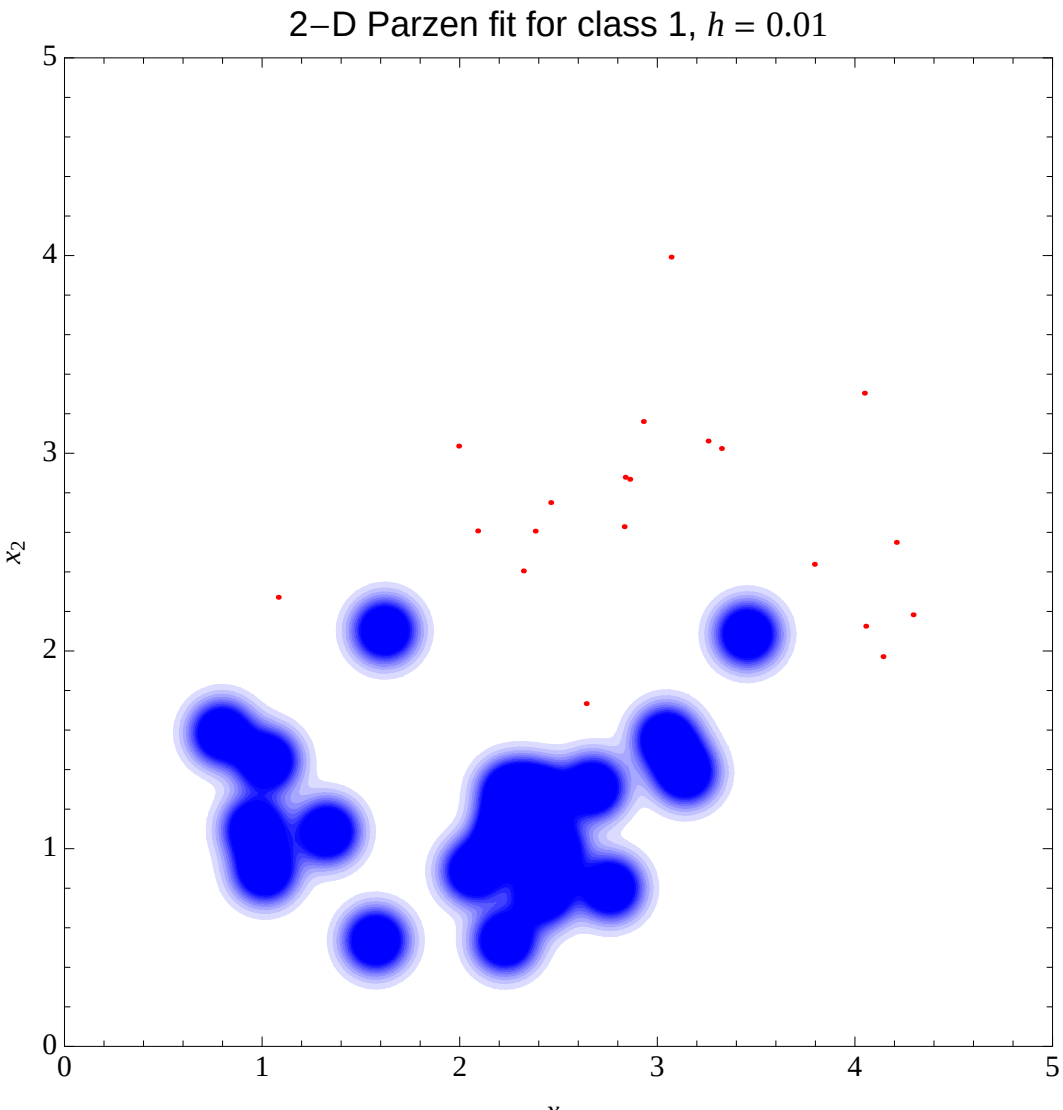

2−D Gaussian fit for class 2
0
1
2
3
4
5
x_1
x_2

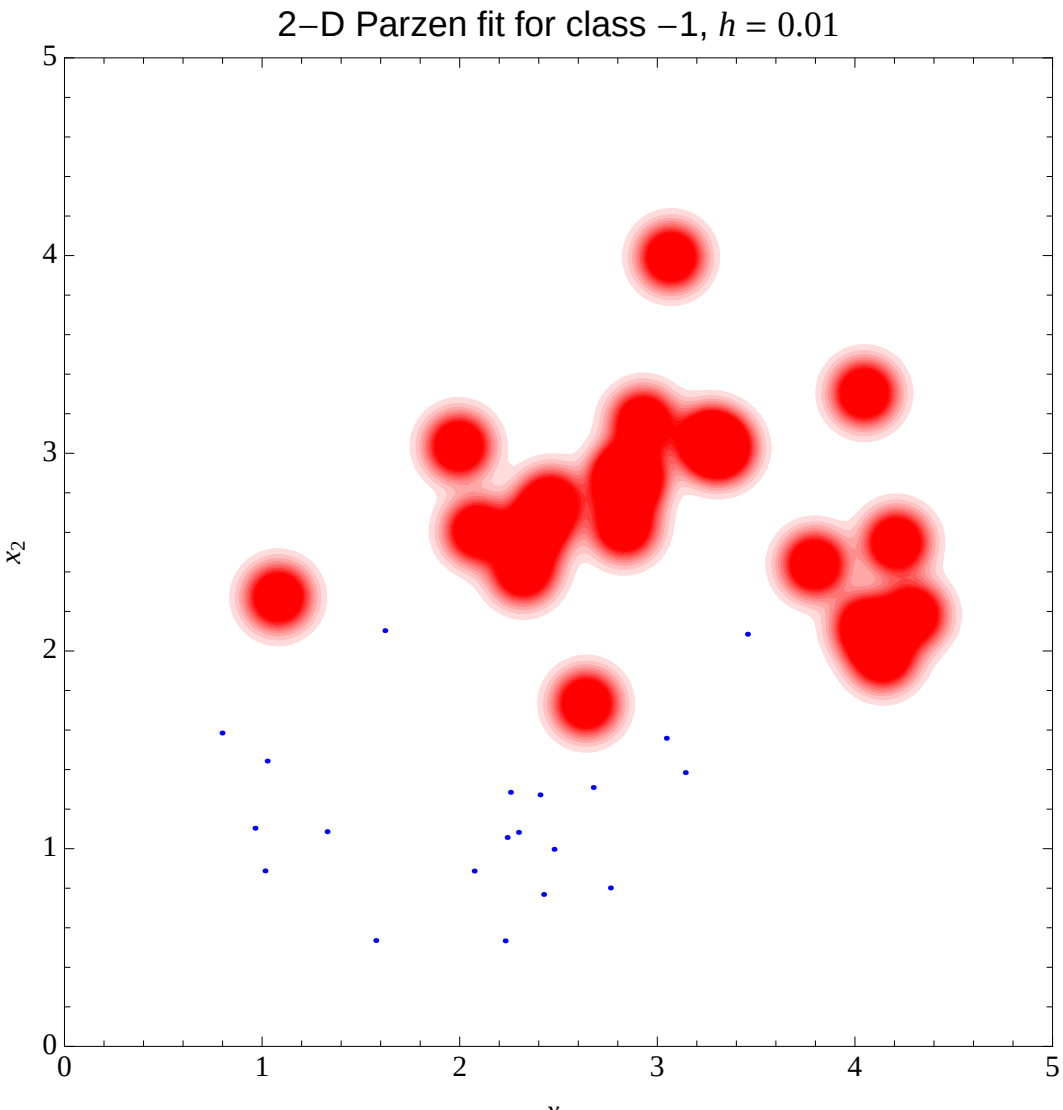

Discriminant function with Gaussian fits
0
1
2
3
4
5
x_1
x_2

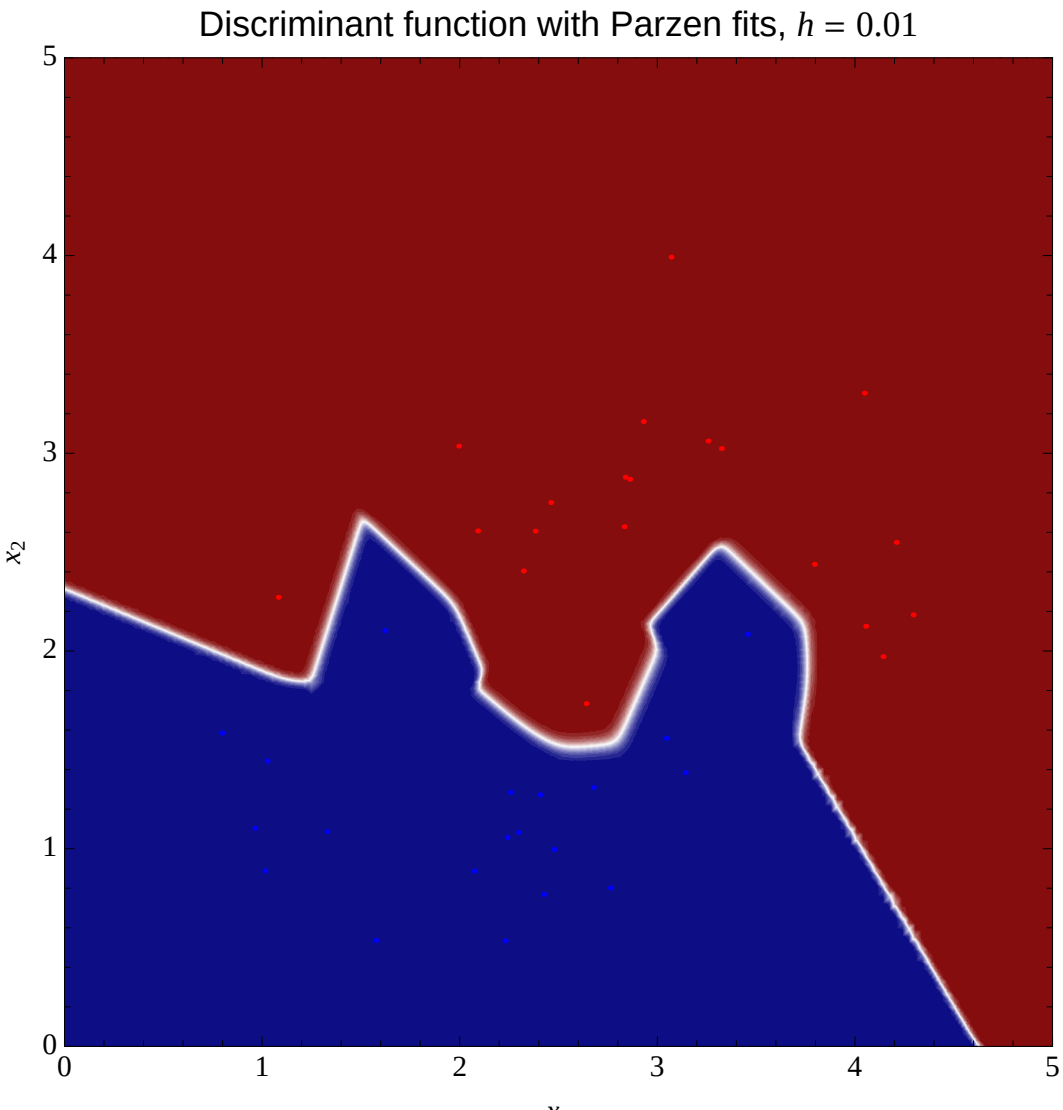

2−D Parzen fit for class 1, $h = 0.25$
x_1
x_2
0
1
2
3
4
5

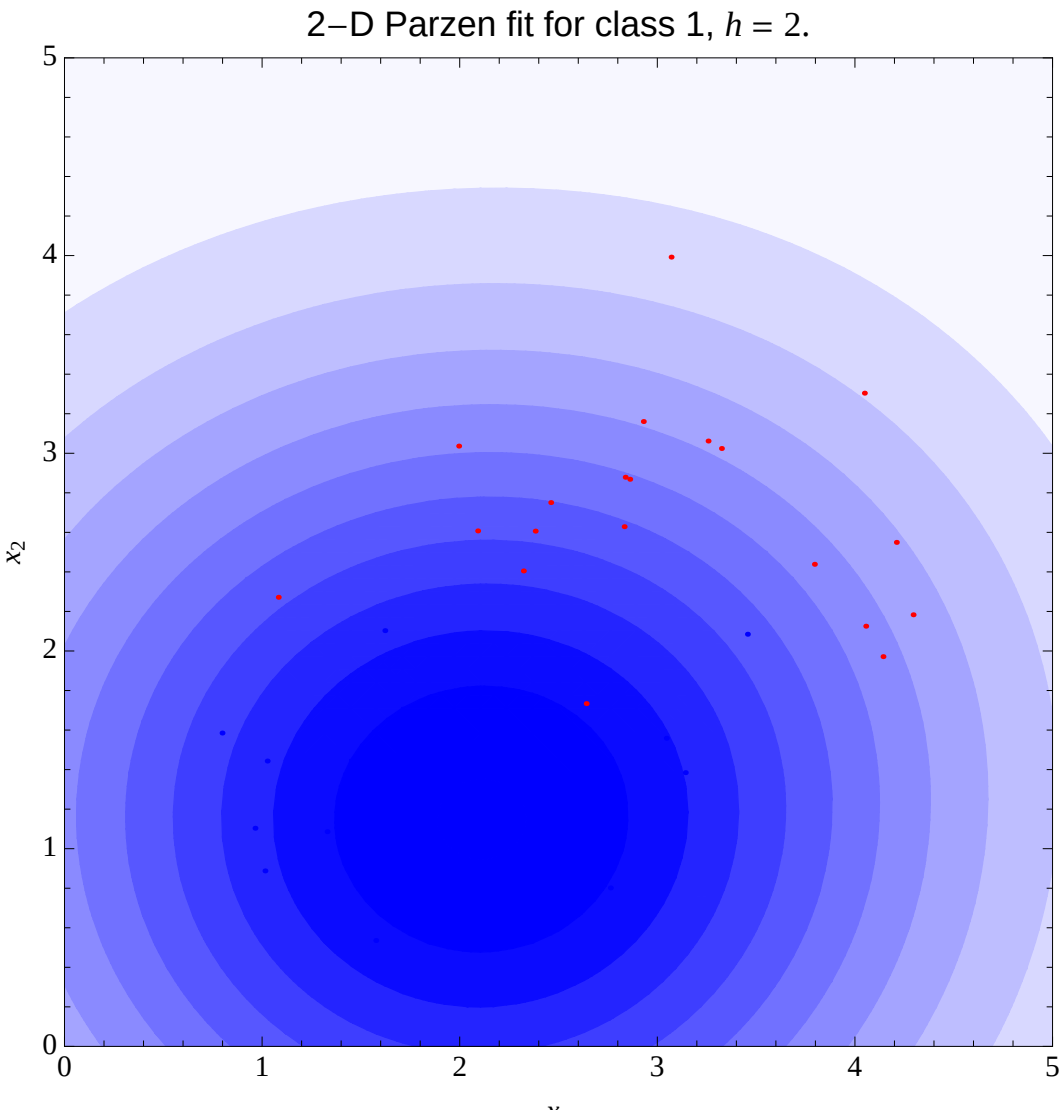

2−D Parzen fit for class −1, $h = 0.25$
0
1
2
3
4
5
x_1
x_2

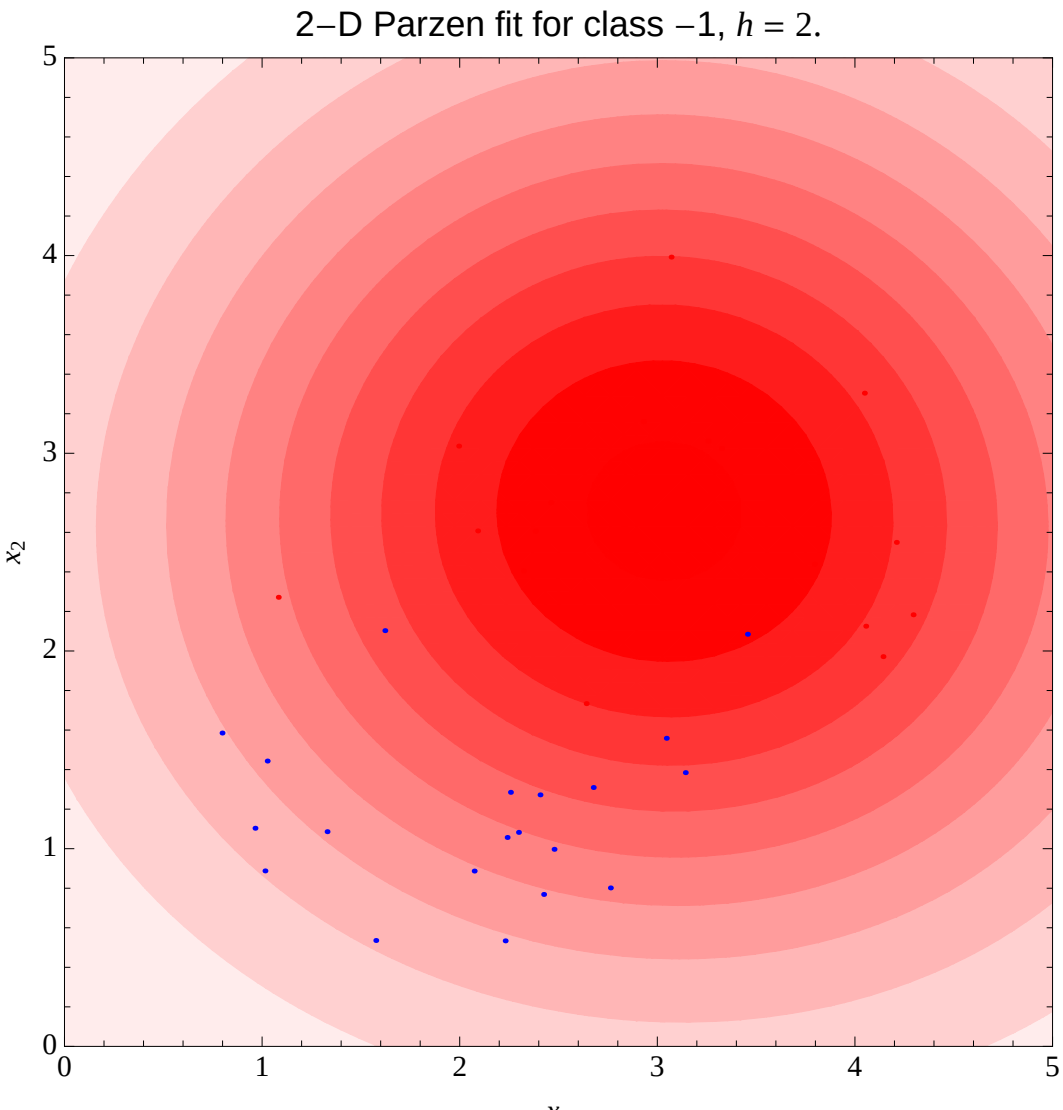

Discriminant function with Parzen fits, $h = 0.25$
x_1
x_2
0
1
2
3
4
5

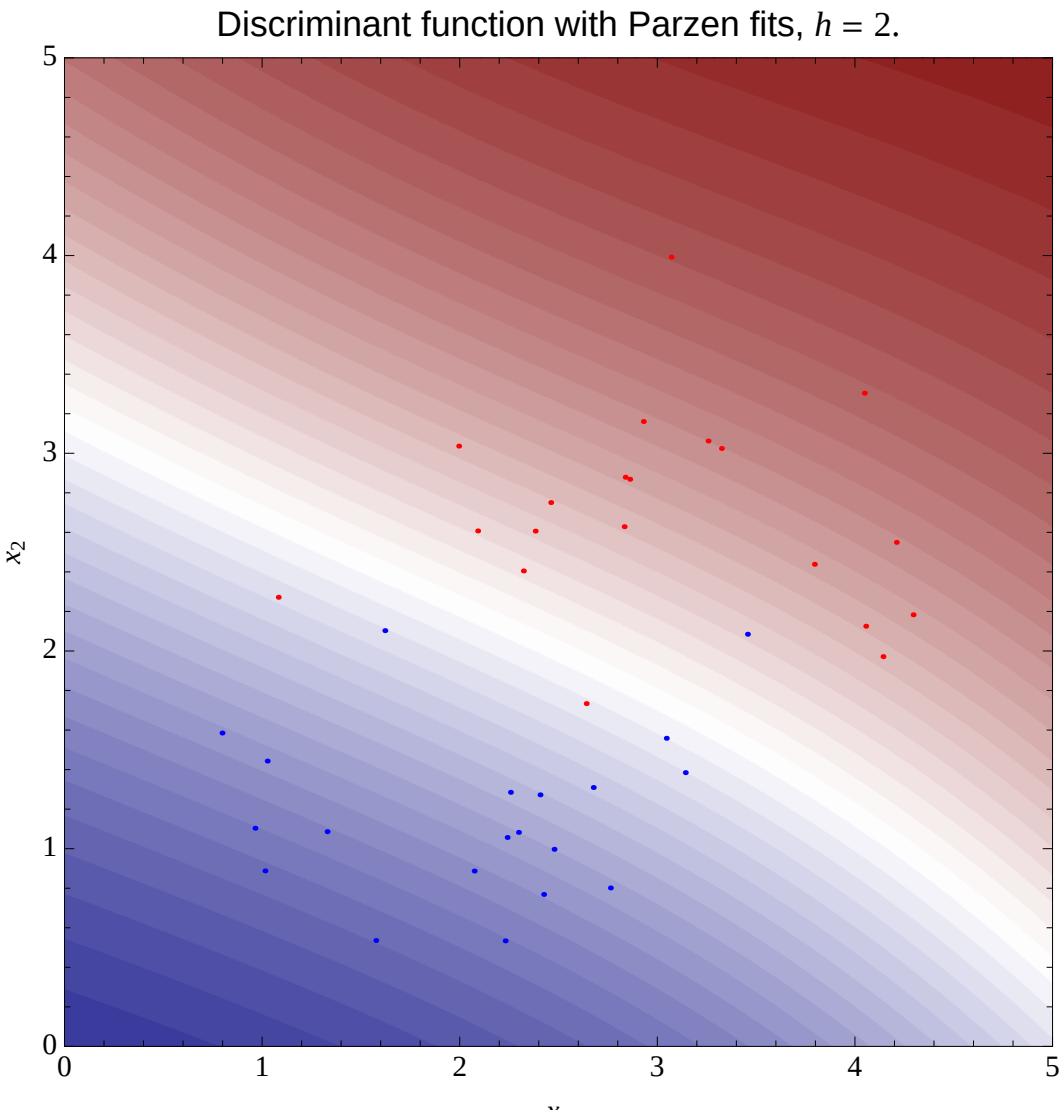

2−D Parzen fit for class 1, $h = 0.01$
x_1
x_2
0
1
2
3
4
5


2−D Parzen fit for class −1, $h = 0.01$
0
1
2
3
4
5
x_1
x_2


Discriminant function with Parzen fits, $h = 0.01$
0
1
2
3
4
5
x_1
x_2


2−D Parzen fit for class 1, $h = 2$.
0
1
2
3
4
5
x_1
x_2


2−D Parzen fit for class −1, $h = 2$.
0
1
2
3
4
5
x_1
x_2


Discriminant function with Parzen fits, $h = 2$.
x_1
x_2
0
1
2
3
4
5


Training and test error rates
0.20
0.15
0.10
0.05
0.00
error rate
Overfitting
Underfitting
0.05
0.10
0.50
1.00
5.00
h

Curse of dimensionality

- Capacity/complexity control becomes a real issue in high-dimensional spaces
  - in a 10000-dimensional space a linear function has 10000 parameters!
  - we need a lot of training points to “fill” the space
- Examples
  - images, music, language, text, bioinfo (genetics, proteomics)
- We prefer to learn the discriminant function directly, without going through the density estimation step

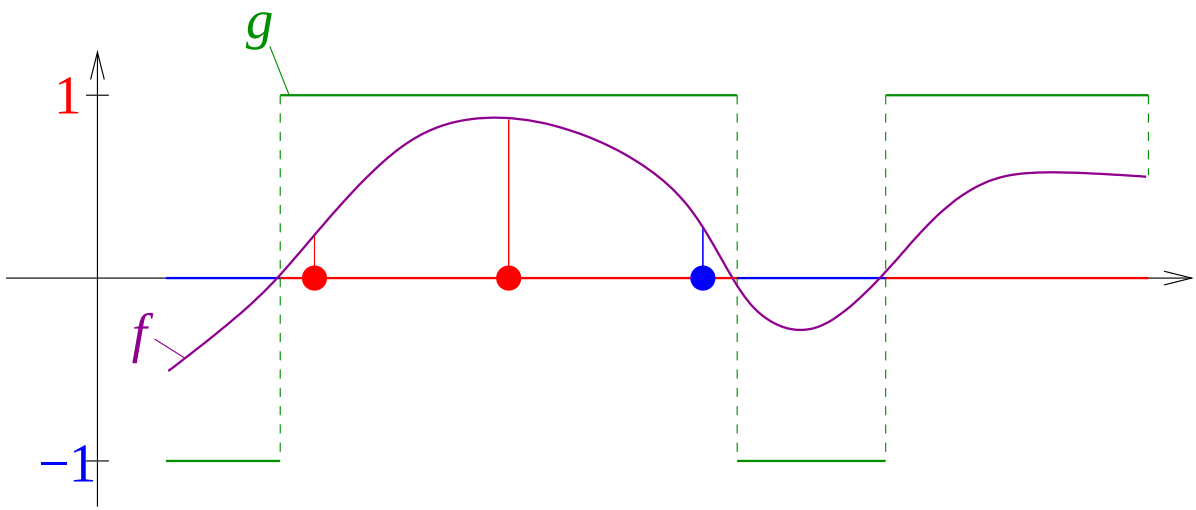
The classification model

- Observation vector: $\mathbf{x} \in \mathbb{R}^d$
- Class label: $y \in \{-1, 1\}$ (or $y \in \{1, \ldots, K\}$)
- Classifier: $g : \mathbb{R}^d \to \{-1, 1\}$
- Discriminant function: $f : \mathbb{R}^d \to [-1, 1]$
  - $\longrightarrow$ classifier

$$g(\mathbf{x}) = \begin{cases} 1, & \text{if } f(\mathbf{x}) \geq 0, \\ -1, & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

  - decision boundary: $\{\mathbf{x} : f(\mathbf{x}) = 0\}$

The classification model



- Discriminant function: $f : \mathbb{R}^d \to [-1, 1]$

- $\longrightarrow$ classifier

$$g(\mathbf{x}) = \begin{cases} 1, & \text{if } f(\mathbf{x}) \geq 0, \\ -1, & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

- decision boundary: $\{\mathbf{x} : f(\mathbf{x}) = 0\}$

The classification model

- Learning from data
  - training set : $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
  - function class : $\mathcal{F}$
  - learning algorithm : $\textsc{Algo} : \left(\mathbb{R}^d \times \{-1, 1\}\right)^n \to \mathcal{F}$

$$\textsc{Algo}(D_n) \mapsto f$$

  - goal: small generalization error $R(g) = P\left[g(\mathbf{X}) \neq Y\right] = P\left[f(\mathbf{X})Y \leq 0\right]$
  - learning principle: minimize the training error

$$\widehat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{g(\mathbf{x}_i) \neq y_i\}$$

The classification model

- goal: small generalization error

$$R(g) = P\left[g(\mathbf{X}) \neq Y\right]$$

- Learning principle: minimize the training error

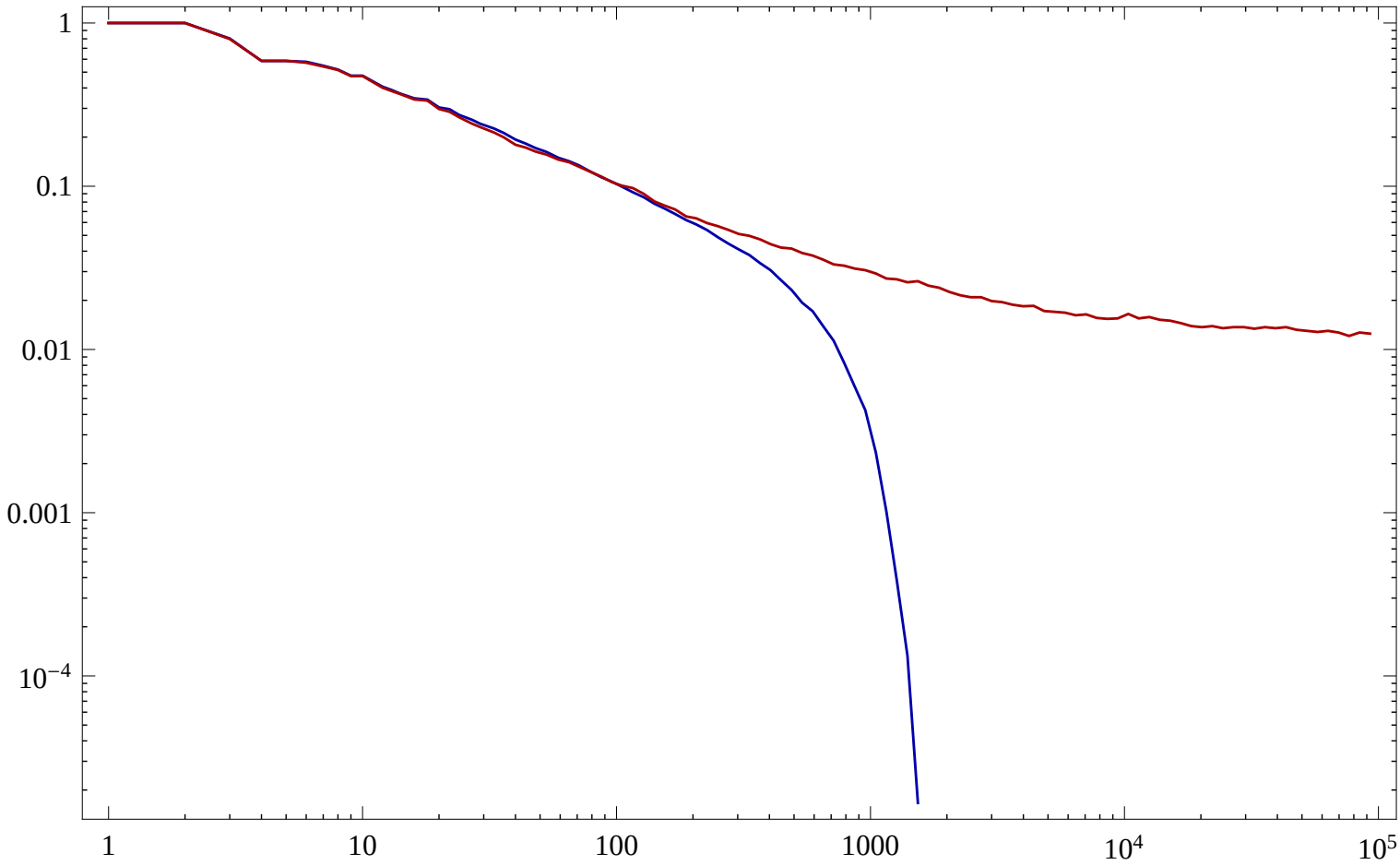
$$\widehat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{g(\mathbf{x}_i) \neq y_i\right\}$$

- Generalization error bounds

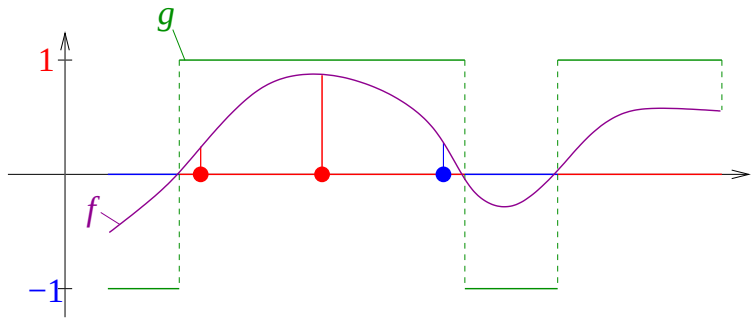
$$\text{“ } R(g) \leq \widehat{R}(g) + O\left(\frac{\dim_{\mathsf{VC}}(\mathcal{F})\log n}{n}\right) \text{ ”}$$

The classification model

AdaBoost on MNIST: training and test error rates



The classification model

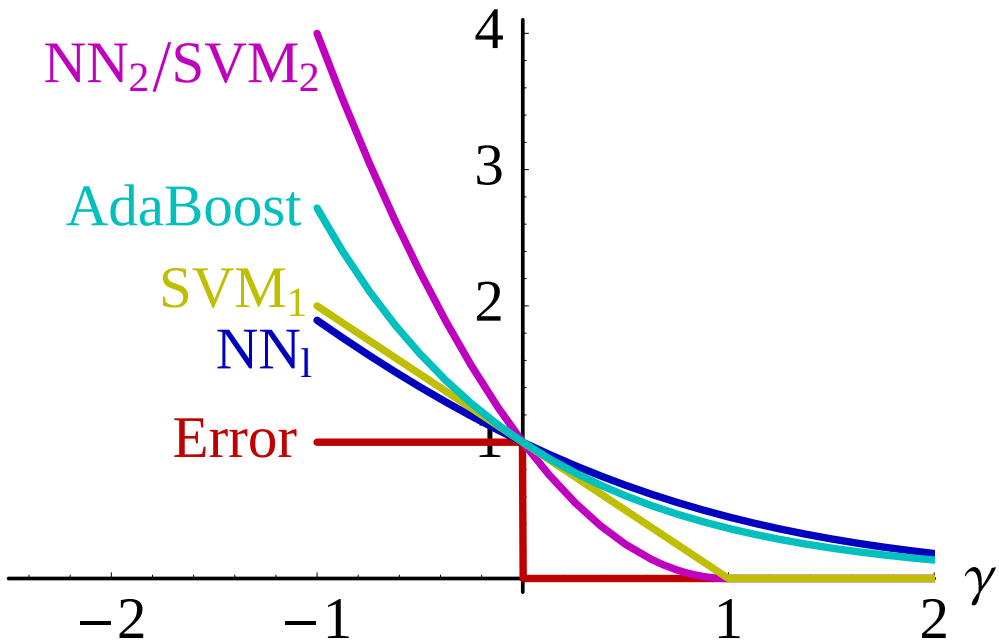


- Margin: $\gamma = y \cdot f(\mathbf{x})$
  - classification error $\equiv$ negative margin
  - the magnitude of a positive margin quantifies the confidence
  - learning principle: minimize a smooth loss function over the margin

$$\widehat{R}_\gamma(f) = \frac{1}{n}\sum_{i=1}^{n} L\big(f(\mathbf{x}_i)y_i\big)$$

The classification model

- Margin loss functions



Outline

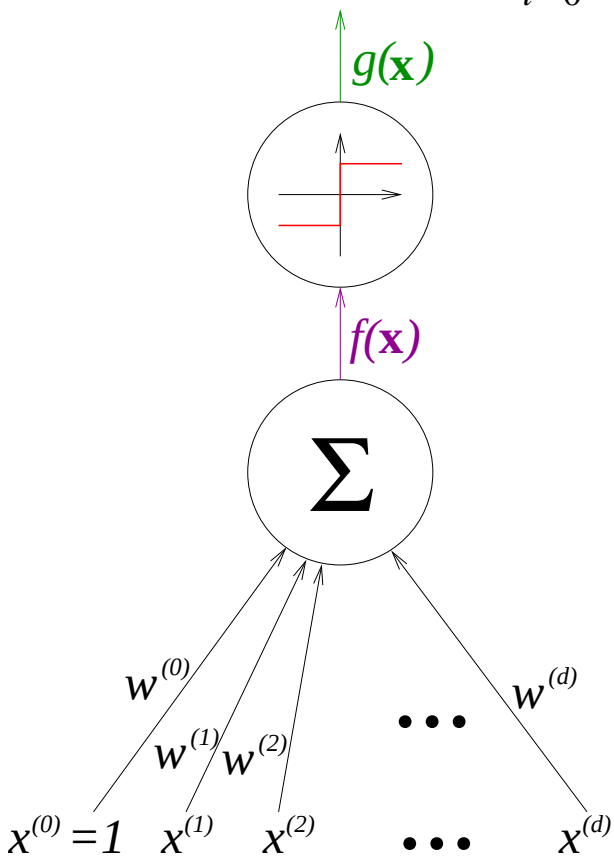
- Non-parametric fitting: two simple examples
- The formal model for classification, learning principles
- Classification algorithms (from a user's point of view)
  - perceptron, neural networks (NN)
  - AdaBoost
  - the Support Vector Machine (SVM)
- Machine learning research motivated by HEP applications

History

- Algorithms
  - 1958: Perceptron [Rosenblatt, ’58] – [Minsky–Papert ’69]
  - 1986: Multilayer perceptrons (neural networks) and the back-propagation algorithm [Rumelhart–Hinton–Williams, ’86]
  - 1995: Support vector machines [Boser–Guyon–Vapnik, ’92], [Cortes–Vapnik, ’95]
  - 1997: boosting, AdaBoost [Freund, ’95], [Freund–Schapire, ’97]

The perceptron

- Linear discriminant functions: $f(\mathbf{x}) = \sum_{i=0}^{d} w^{(i)} \cdot x^{(i)} = \langle \mathbf{w}, \mathbf{x} \rangle$



The perceptron

- Linear discriminant functions: $f(\mathbf{x}) = \sum_{i=0}^{d} w^{(i)} \cdot x^{(i)} = \langle \mathbf{w}, \mathbf{x} \rangle$

- Algorithm
  - simple iterative error correction
  - convergence if the data is linearly separable
  - oscillation for linearly non-separable data

Generalized linear discriminant functions

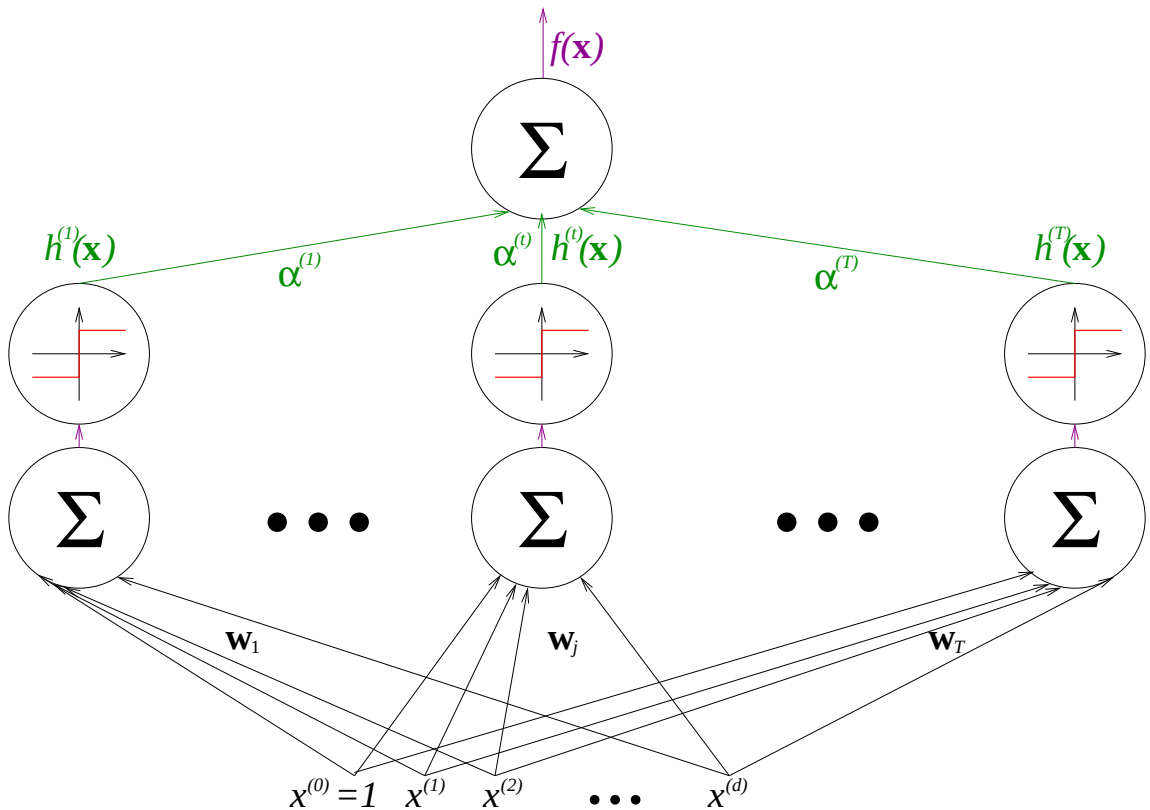
- Model:

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} h^{(j)}(\mathbf{x})$$

- $h^{(j)} : \mathbb{R}^d \to [-1, 1]$
  - simple classifiers/discriminant functions, features, experts
- $\alpha^{(j)} \in \mathbb{R}^+$
  - weight of the expert $h^{(j)}$ in the final vote

Multilayer perceptron (neural net)

- Model: $f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} \sigma(\langle \mathbf{w}_j, \mathbf{x} \rangle)$



Multilayer perceptron (neural net)

- Model: $f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} \sigma(\langle \mathbf{w}_j, \mathbf{x} \rangle)$

- Algorithm:
  - gradient descent optimization
  - differentiable error functions $\rightarrow$ margin loss
  - differentiable activation function σ: the sigmoid
  - local minima, “engineering”, parameters to tune
  - fast, works well if well-tuned
  - versatile: multi-class classification, regression, density estimation

AdaBoost

- Model:

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} h^{(j)}(\mathbf{x})$$

  - no restriction on the form of $h^{(j)}(\mathbf{x})$
  - often "decision stumps" :

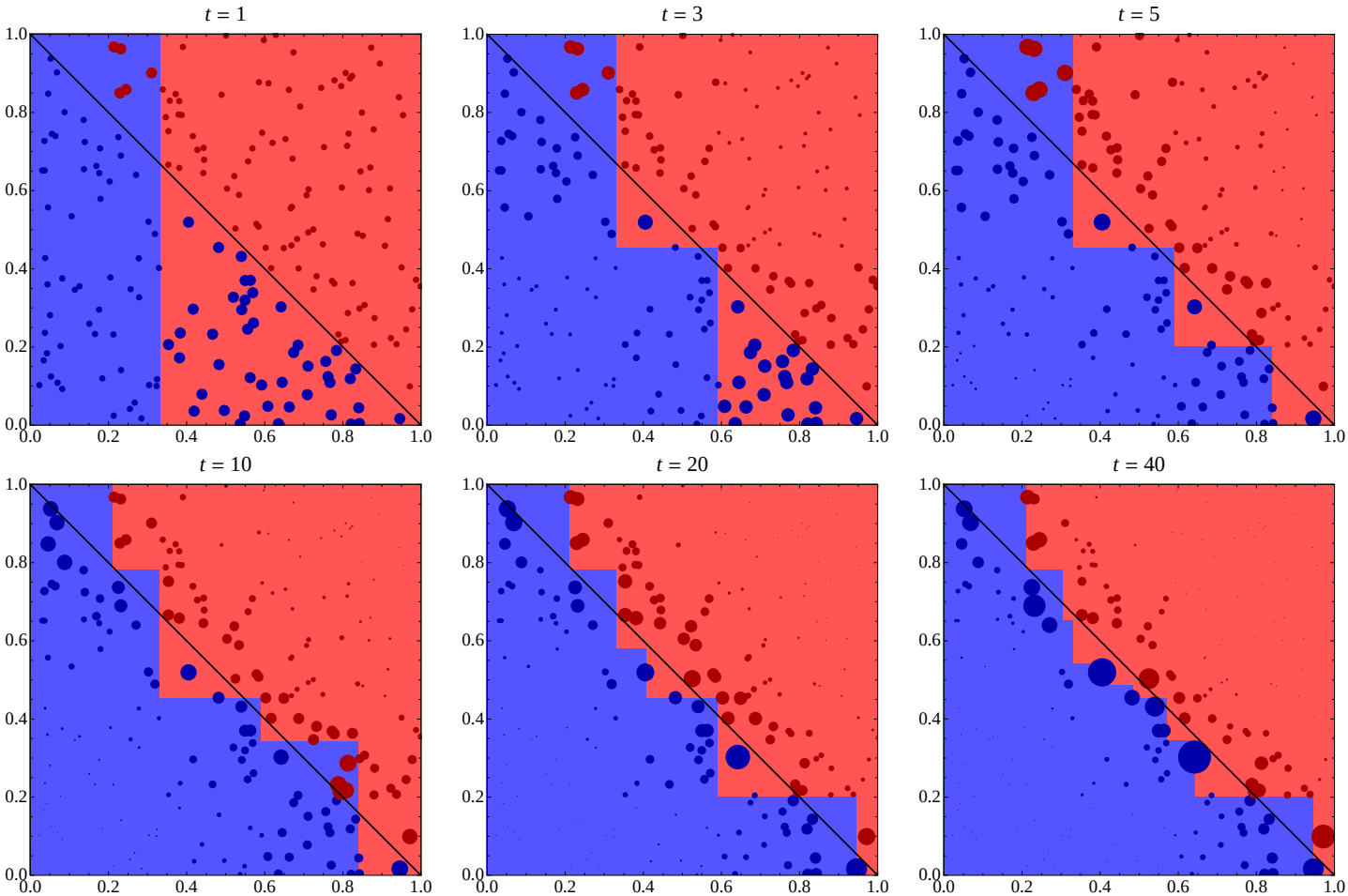
$$h_{\ell,\theta}(\mathbf{x}) = \begin{cases} +1 & \text{if } x^{(\ell)} \geq \theta, \\ -1 & \text{otherwise} \end{cases}$$

where $\mathbf{x} = \left(x^{(1)}, \ldots, x^{(d)}\right)$

AdaBoost

- Algorithm
  - add one expert/base classifier (stump or tree) at a time
  - choose the base classifier that "corrects" the errors of the previous ones – formally: maintain a weight distribution over the data points
  - coefficient α is proportional to the accuracy
  - weights are adjusted: decreased for correctly classified points, increased for incorrectly classified points

AdaBoost



AdaBoost

- Algorithm
  - extremely simple learning, limited parameter tuning
  - fast
  - the choice of the pool of experts captures the a-priori knowledge
  - no restriction on the form of the simple classifiers
  - multi-class classification is natural, regression is not

Support vector machine

- Model:

$$f(\mathbf{x}) = \sum_{j \in I_{\mathrm{sv}}} \alpha^{(j)} y_j K(\mathbf{x}_j, \mathbf{x})$$

  - $I_{\mathrm{sv}} \subset \{1, \ldots, n\}$ is the set of support vectors
  - $K(\cdot, \cdot)$ is a similarity function (kernel)

- Kernel:

  - $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle \longrightarrow f(\mathbf{x})$ is linear
  - $K(\mathbf{x}, \mathbf{x}') = \left(1 + \langle \mathbf{x}, \mathbf{x}' \rangle\right)^d \longrightarrow f(\mathbf{x})$ is a polynom of degree d
  - $K(\mathbf{x}, \mathbf{x}') = \exp\left(-1/h \|\mathbf{x} - \mathbf{x}'\|^2\right) \longrightarrow f(\mathbf{x})$ is a Gaussian mixture ($\rightarrow$ Parzen)

Support vector machine

- Model:

$$f(\mathbf{x}) = \sum_{j \in I_{\mathrm{sv}}} \alpha^{(j)} y_j K(\mathbf{x}_j, \mathbf{x})$$

- Algorithm:
  - goal: classification boundary equidistant from classes
  - “sophisticated nearest neighbor”
  - slow and complex quadratic programming optimization
  - turn-key algorithm, very limited parameter tuning
  - no multi-class, no regression

Outline

- Non-parametric fitting: two simple examples
- The formal model for classification, learning principles
- Classification algorithms (from a user's point of view)
  - perceptron, neural networks (NN)
  - AdaBoost
  - the Support Vector Machine (SVM)
- Machine learning research motivated by HEP applications

Discovery, cross section

- First

$$f(x) \sim P(\text{signal} \mid \mathbf{x}) = 1 - P(\text{bg} \mid \mathbf{x})$$

is estimated, then it is used to find an optimal region A such that

$$\arg\max_{A \in \mathbb{R}^d} \frac{\sum(\text{signal} \mid \mathbf{x} \in A)}{\sqrt{\sum(\text{bg} \mid \mathbf{x} \in A)}} = \frac{s}{\sqrt{b}} \text{ or } \arg\max_{A \in \mathbb{R}^d} \frac{\sum(\text{signal} \mid \mathbf{x} \in A)}{\sqrt{\sum(\text{signal or bg} \mid \mathbf{x} \in A)}} = \frac{s}{\sqrt{s+b}}$$

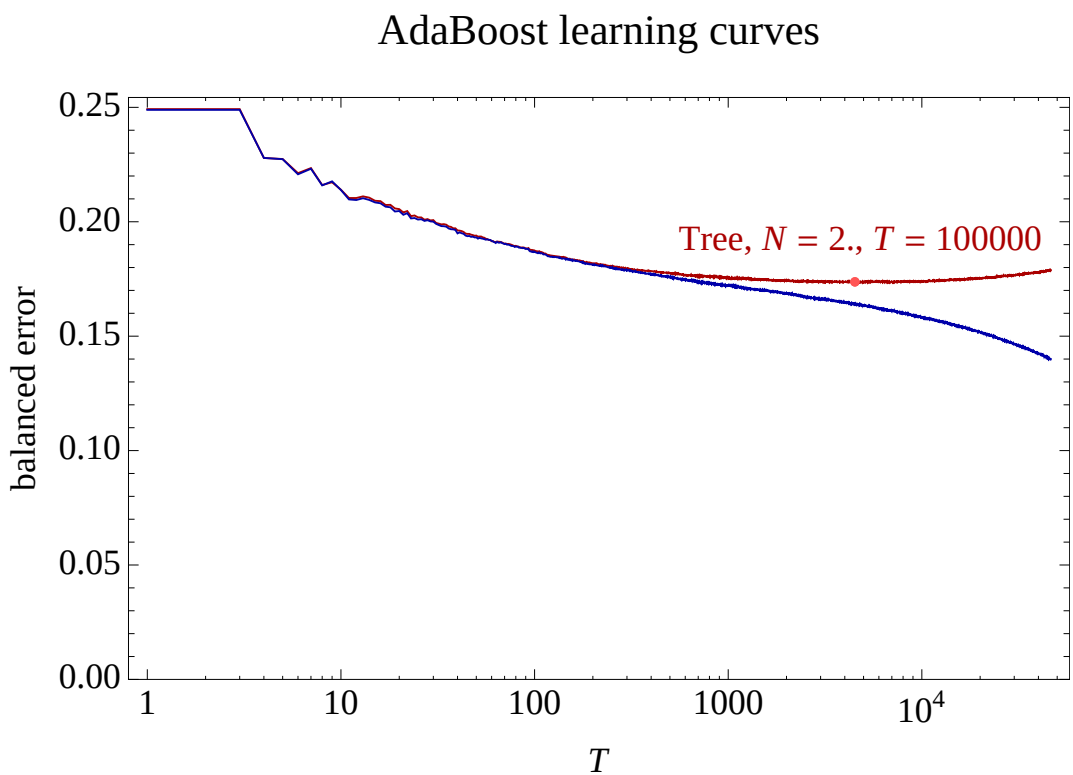
- Although $P(\text{signal} \mid \mathbf{x})$ is related to these goals, it is not the same

- Research goal: modify classification algorithms to directly optimize the modified criteria

- A Kaggle challenge on the Higgs tau-tau channel is running May - Sptember: `http://www.kaggle.com/c/higgs-boson`

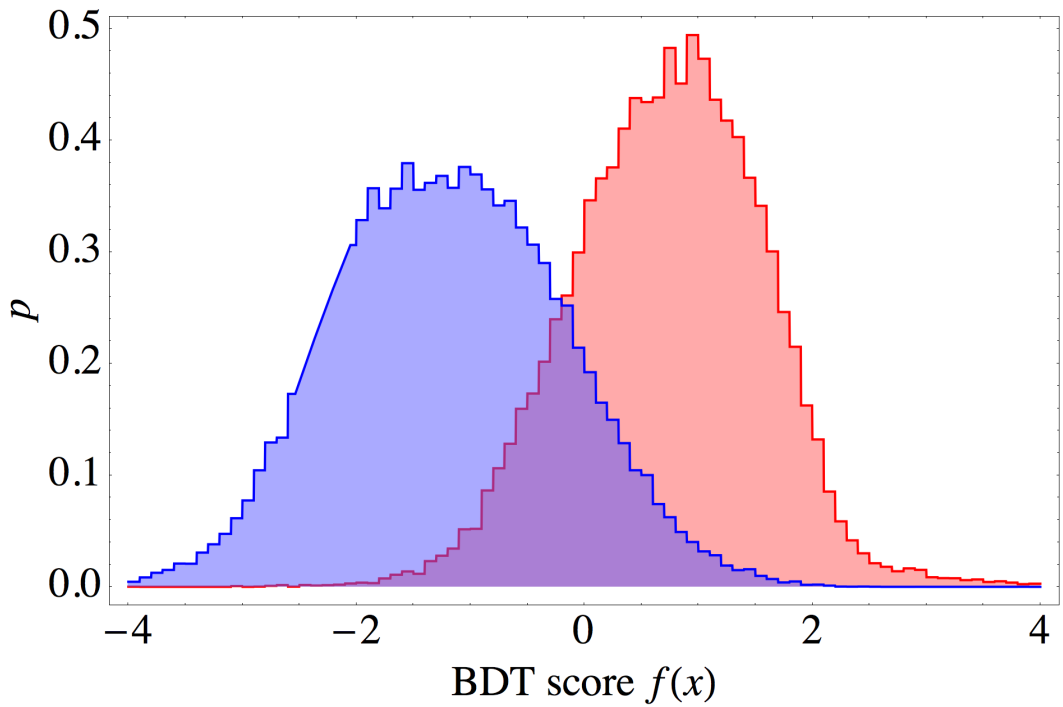
Discovery

- A two-stage approach
  1. optimize a discriminant function $f : \mathbb{R}^d \to \mathbb{R}$ for balanced classification error: $N'_{\mathsf{s}} = N'_{\mathsf{b}} = 0.5$



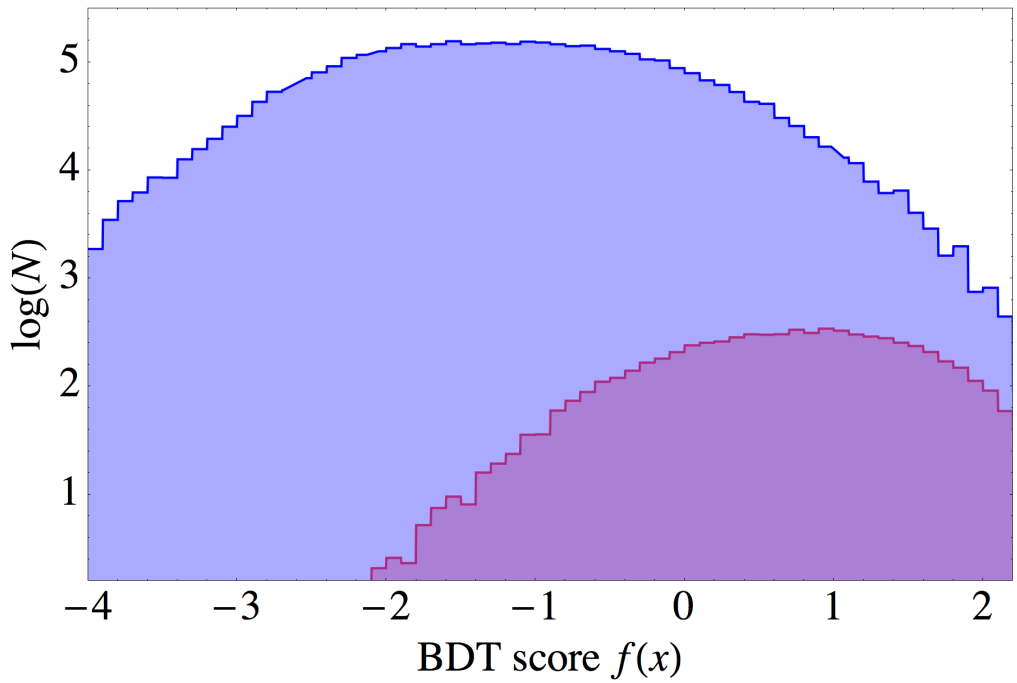
Discovery

- A two-stage approach
  1. optimize a discriminant function $f : \mathbb{R}^d \to \mathbb{R}$ for balanced classification error: $N'_{\mathsf{s}} = N'_{\mathsf{b}} = 0.5$



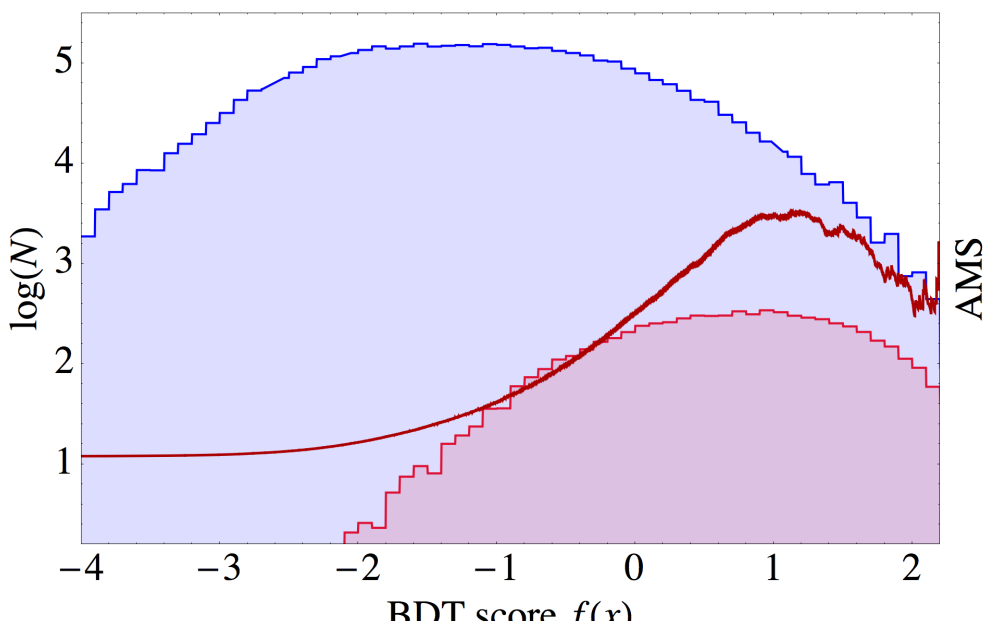
Discovery

- A two-stage approach
  1. optimize a discriminant function $f : \mathbb{R}^d \to \mathbb{R}$ for balanced classification error: $N'_{\mathsf{s}} = N'_{\mathsf{b}} = 0.5$



Discovery

- A two-stage approach
  1. optimize a discriminant function $f : \mathbb{R}^d \to \mathbb{R}$ for balanced classification error: $N'_{\mathrm{s}} = N'_{\mathrm{b}} = 0.5$
  2. define $g(\mathbf{x}) = \mathrm{sign}\big(f(\mathbf{x}) - \theta\big)$ and optimize θ for maximizing the AMS (with the original weights)



Trigger

- Real-time classification: the evaluation of $f(x)$ should be computationally efficient
- Competing goals of accuracy/speed
- Common although not mainstream in ML (object detection, web-page ranking)
- Classical design: cascade classification
- Research goal: explicitly include the accuracy/speed trade-off into training

Final thoughts

- If you can build a generative model for the class densities, do it
- All algorithms are optimized for classification, not hypothesis testing, discovery, trigger, etc.
  - doesn't mean they don't work, but they are probably not optimal
- Non-parametric methods also exist for other objectives (e.g., regression, density estimation)

Software

- mloss.org
- scikit-learn.org
- www.cs.waikato.ac.nz/ml/weka
- multiboost.org
- svmlight.joachims.org
- www.csie.ntu.edu.tw/∼cjlin/libsvm
- www.torch.ch

Other resources

- http://www.epj-conferences.org/index.php?option=com_toc&url=/articles/epjconf/abs/2013/16/contents/contents.html
- http://higgsml.lal.in2p3.fr/documentation/

Thank you!