# Multivariate classification

Balázs Kégl

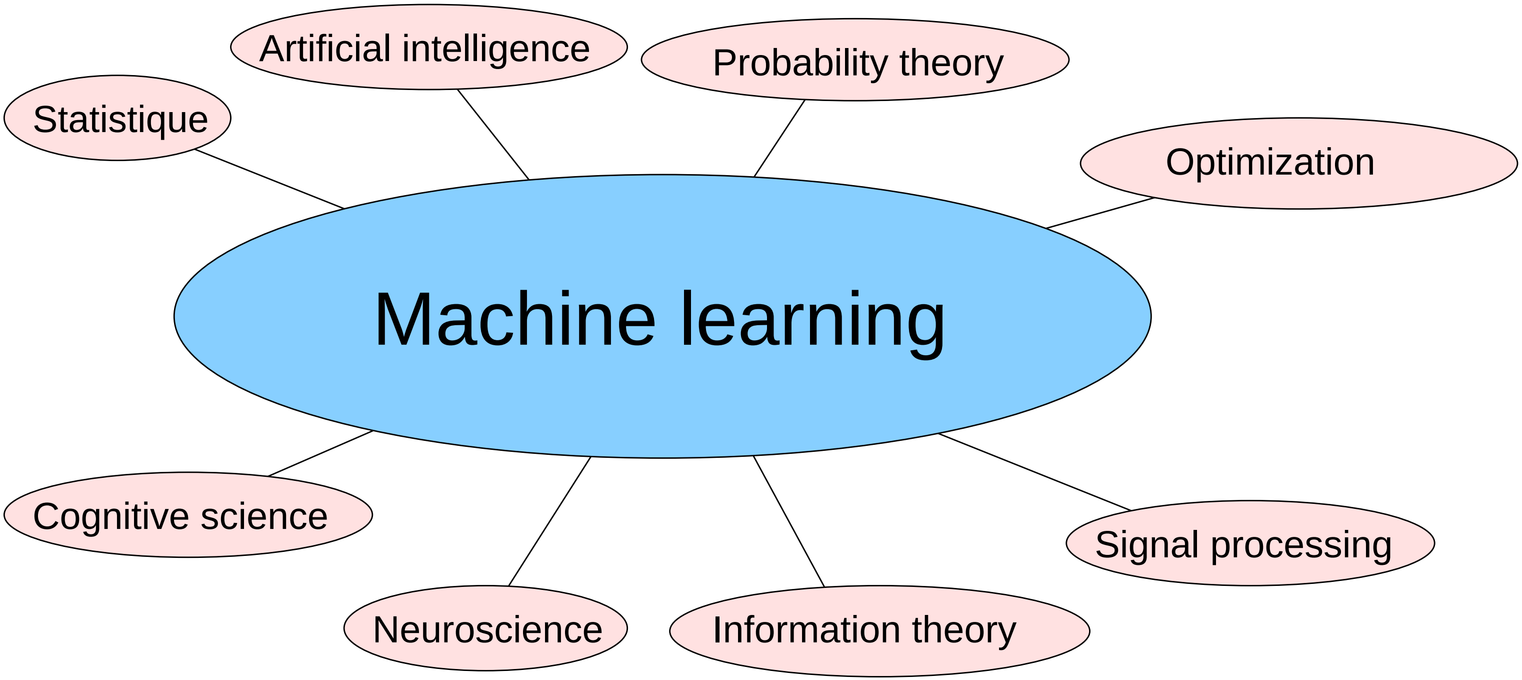
LAL / University of Paris-Sud

School of Statistics
20 May, 2010

# 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)
- Most known algorithms
  - Classification: neural network, boosting, support vector machine
  - Regression: neural network, Gaussian process
  - Density estimaton: mixture of Gaussians
  - Clustering, dimensionality reduction: k-means, spectral clustering, local linear embedding, ISOMAP, nonlinear PCA, kernel PCA

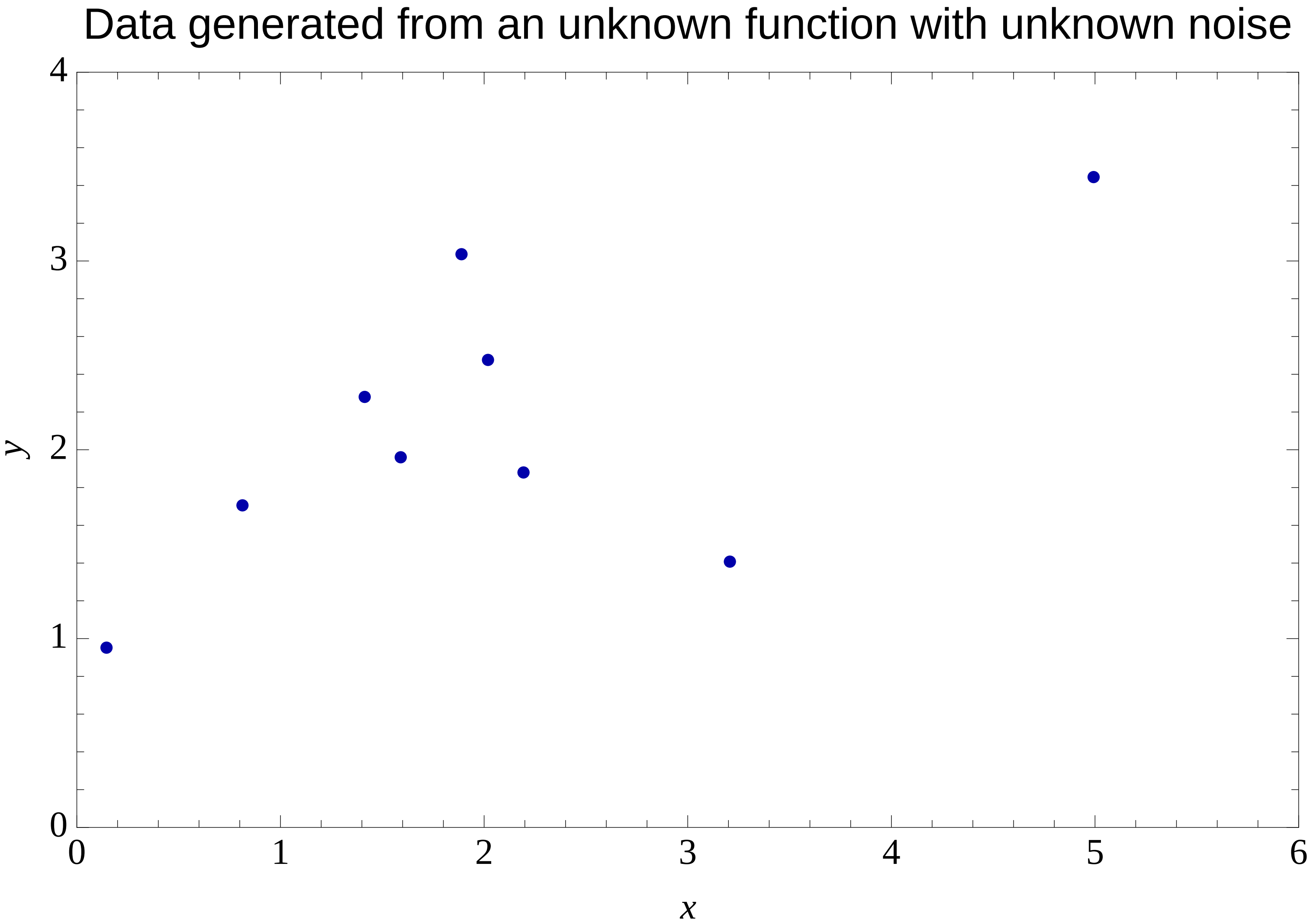
# Machine learning at the crossroads

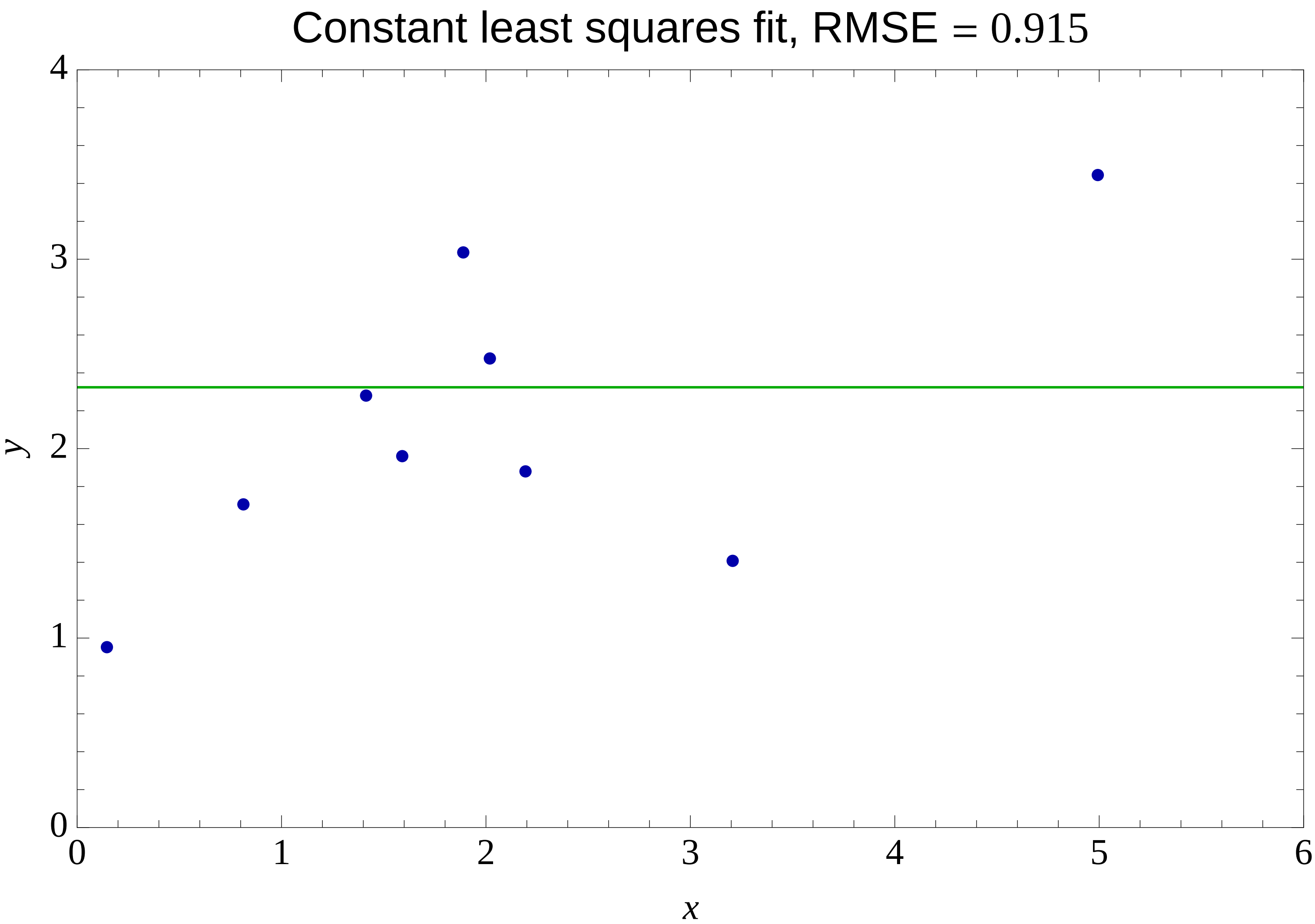


# Outline

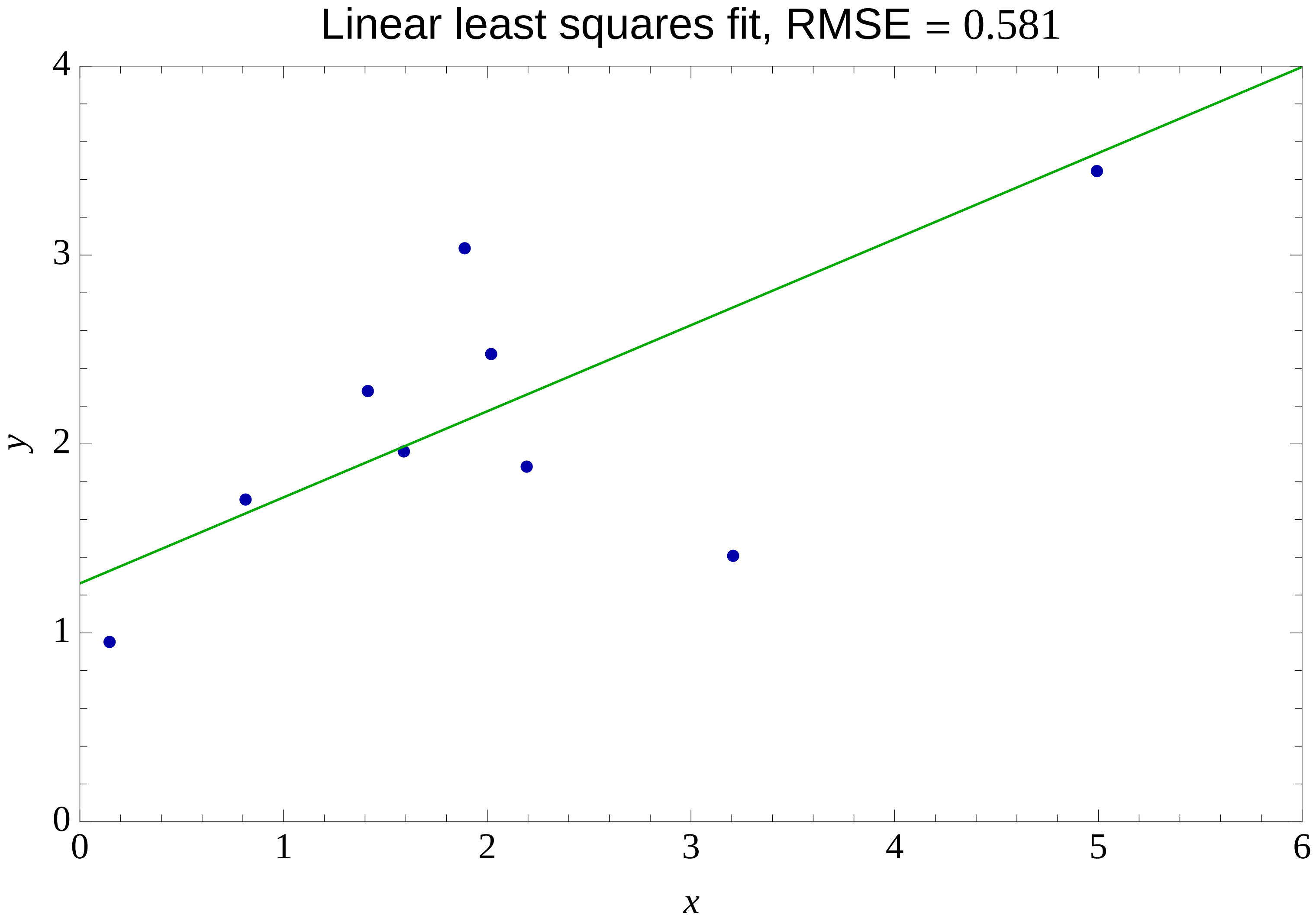


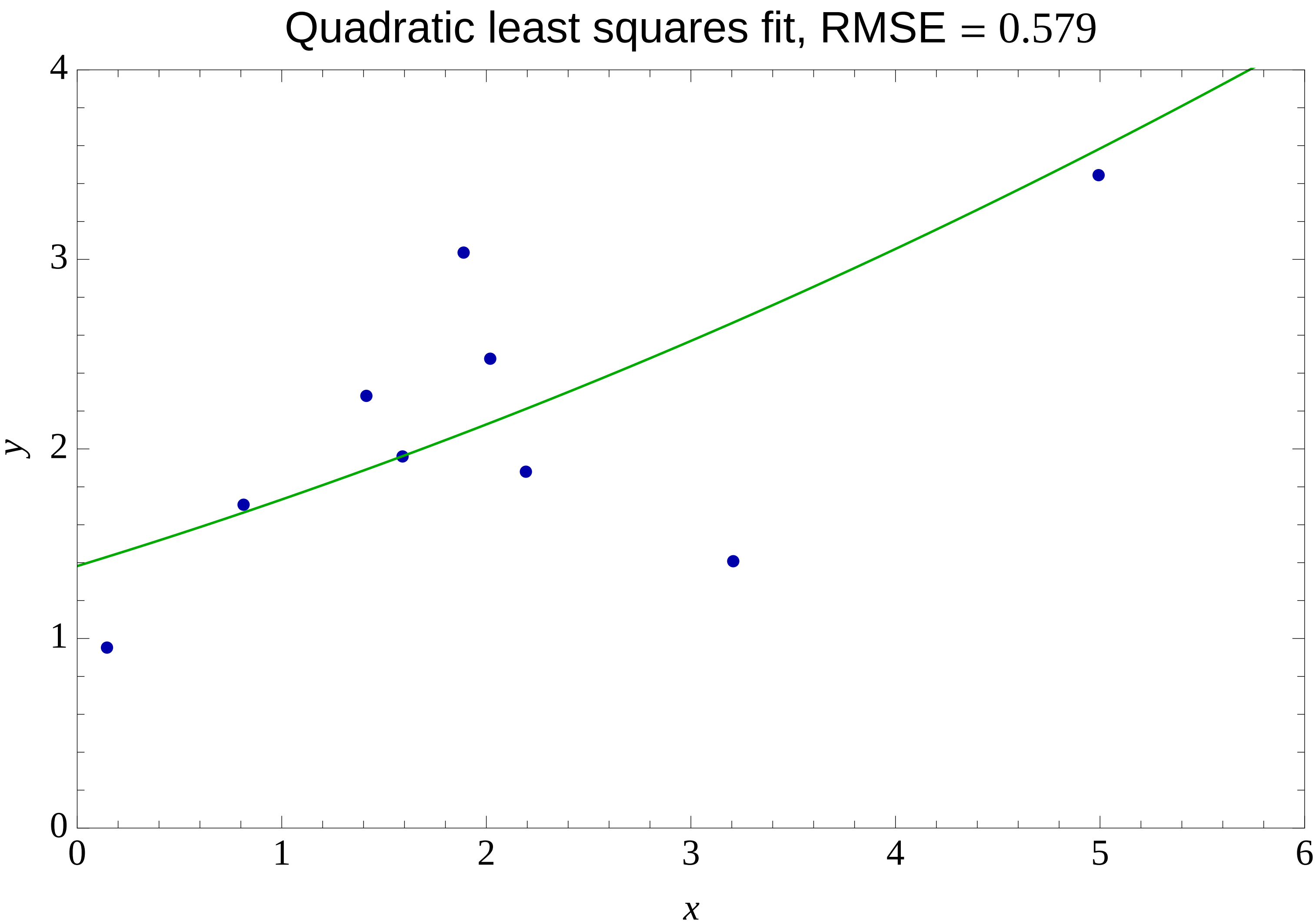
- Non-parametric fitting: two simple examples
- The formal model for classification, learning principles
- Classification algorithms (from a user's point of view)
  - perceptron, neural networks (NN)
  - AdaBoost
  - the Support Vector Machine (SVM)

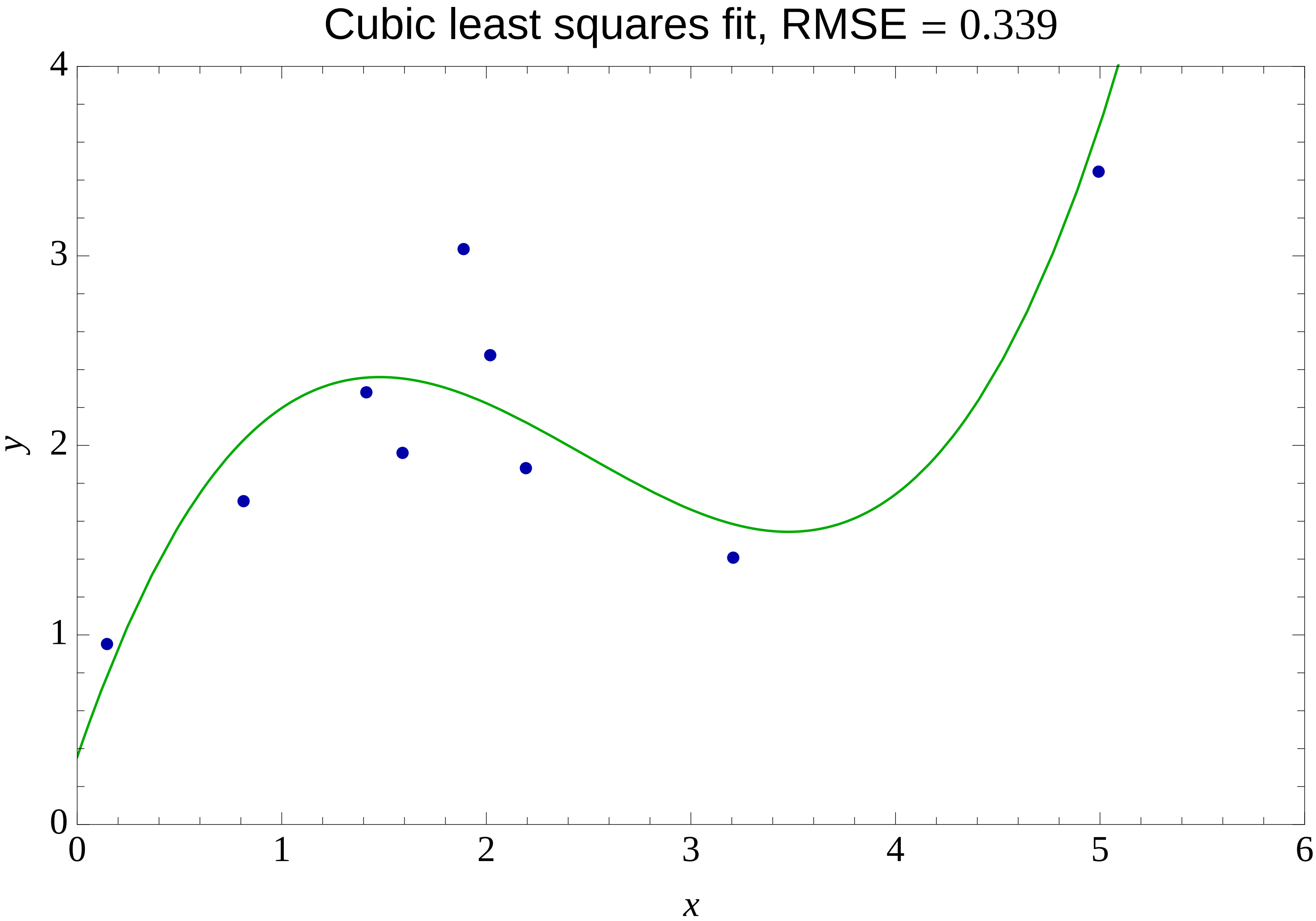

Data generated from an unknown function with unknown noise
y
x
0
1
2
3
4
0
1
2
3
4
5
6

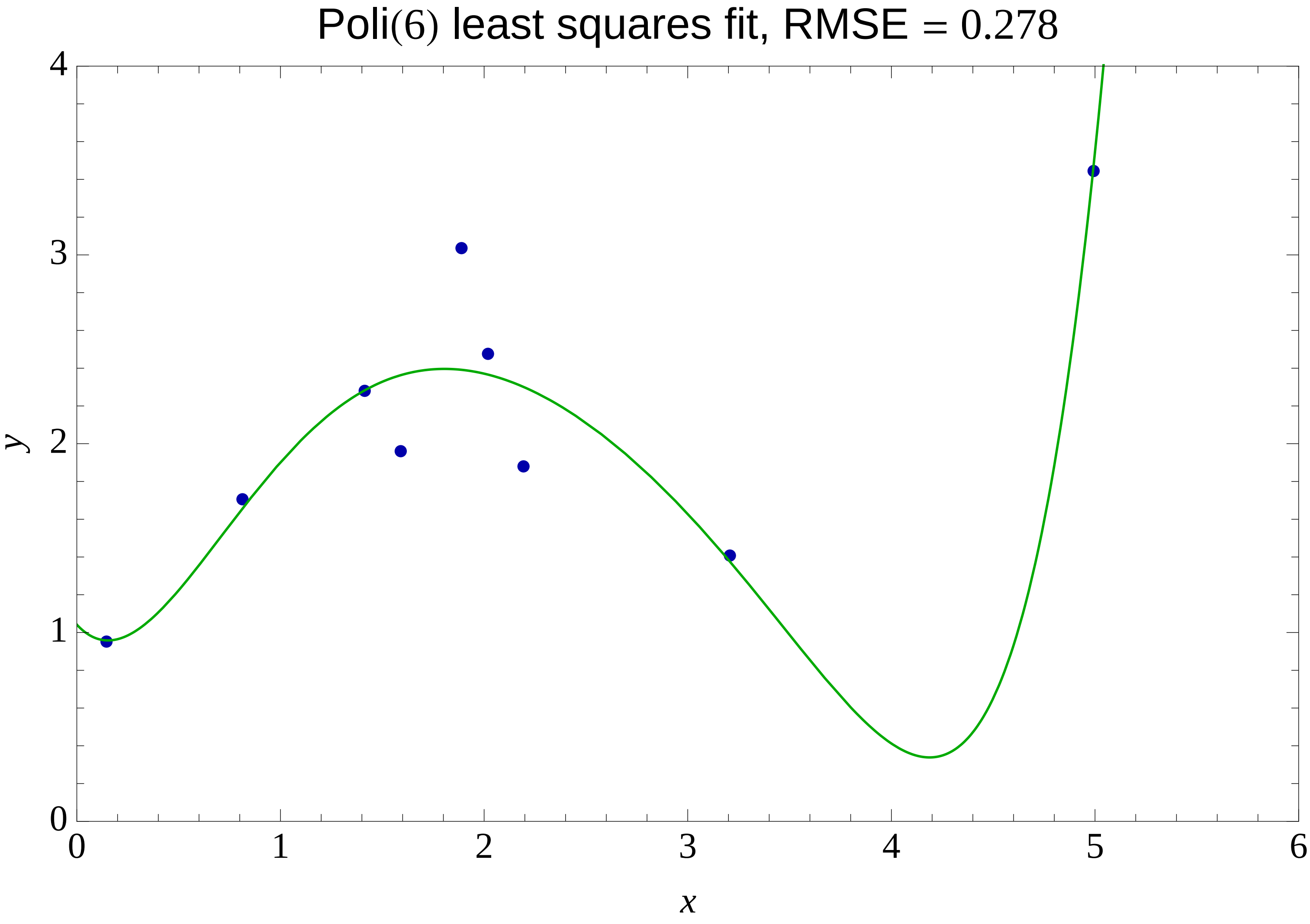


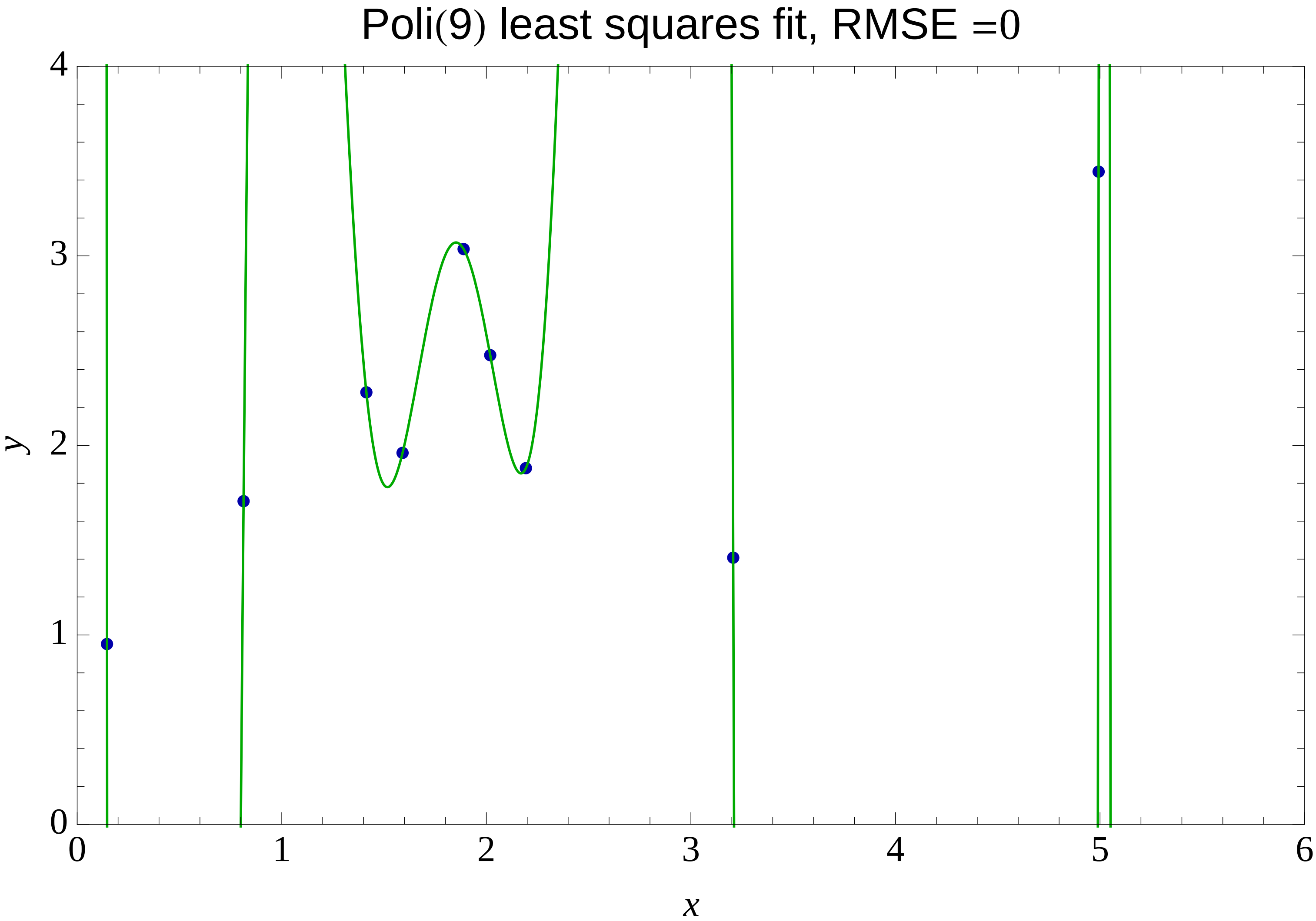
Constant least squares fit, RMSE = 0.915
4
3
2
1
0
y
0
1
2
3
4
5
6
x


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


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


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


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


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

# Non-parametric fitting

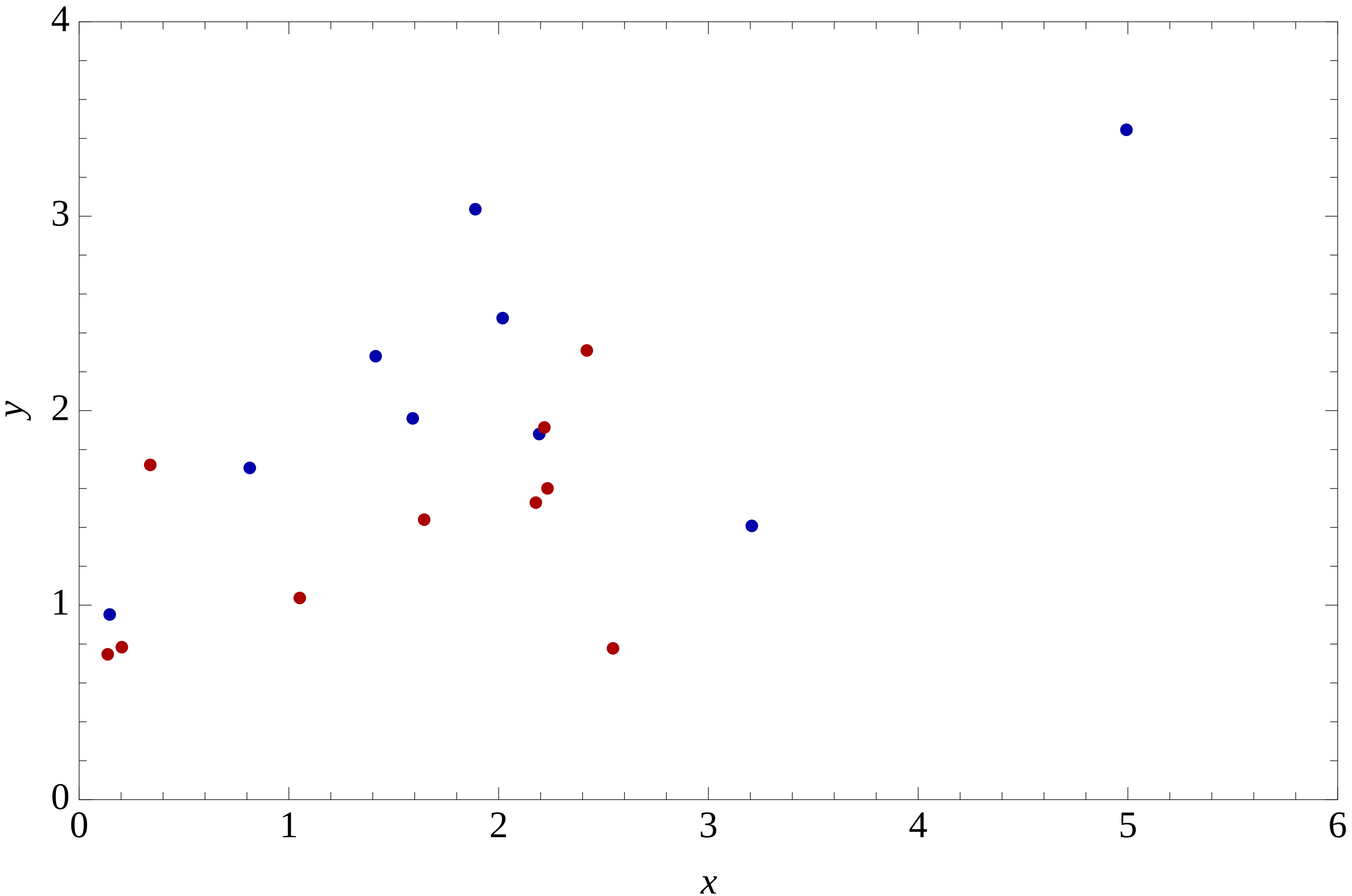


- Minimizing the training cost (RMSE) does not work if the function class is not fixed beforehand

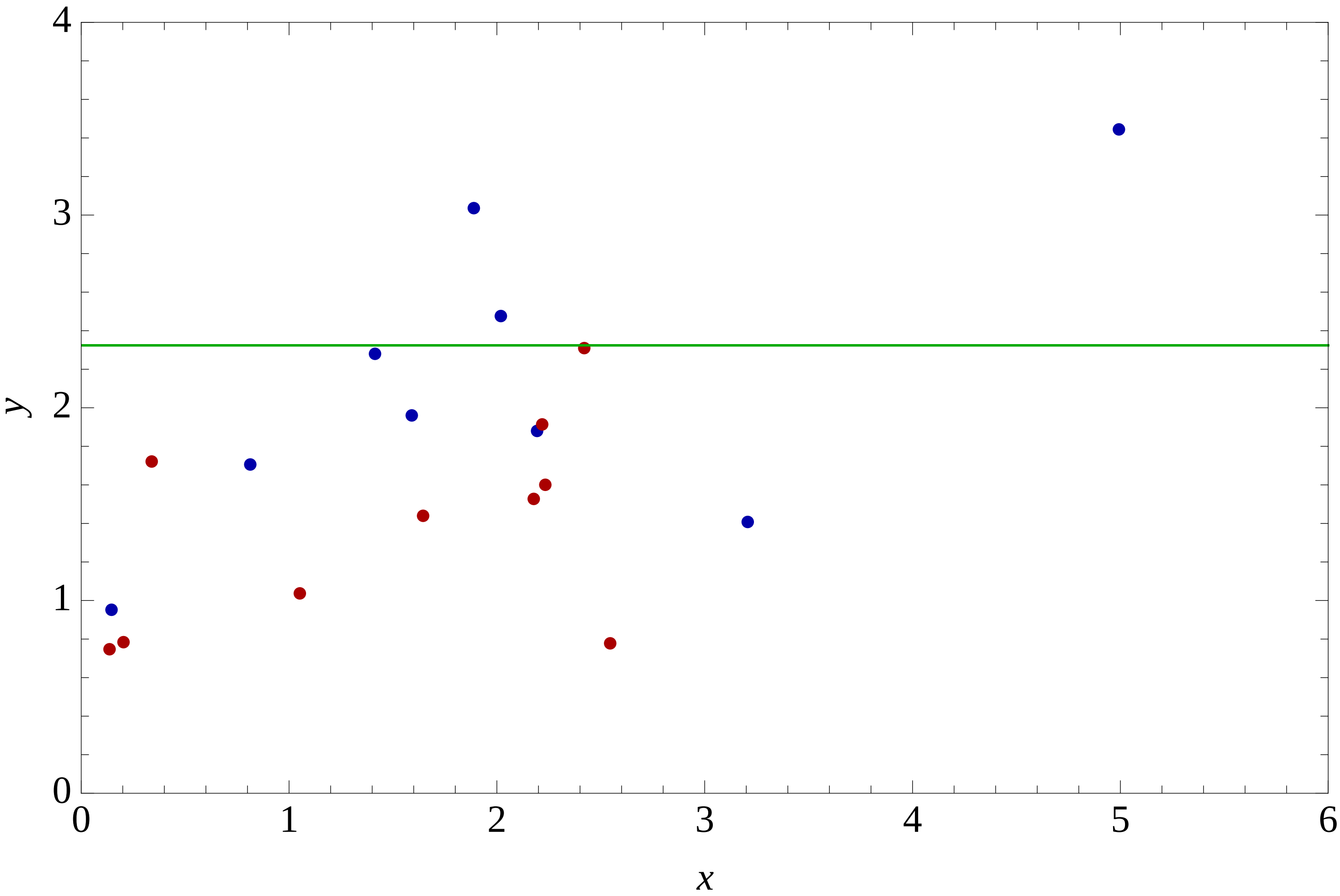
- If you do not know the correct function class, you should not fix it

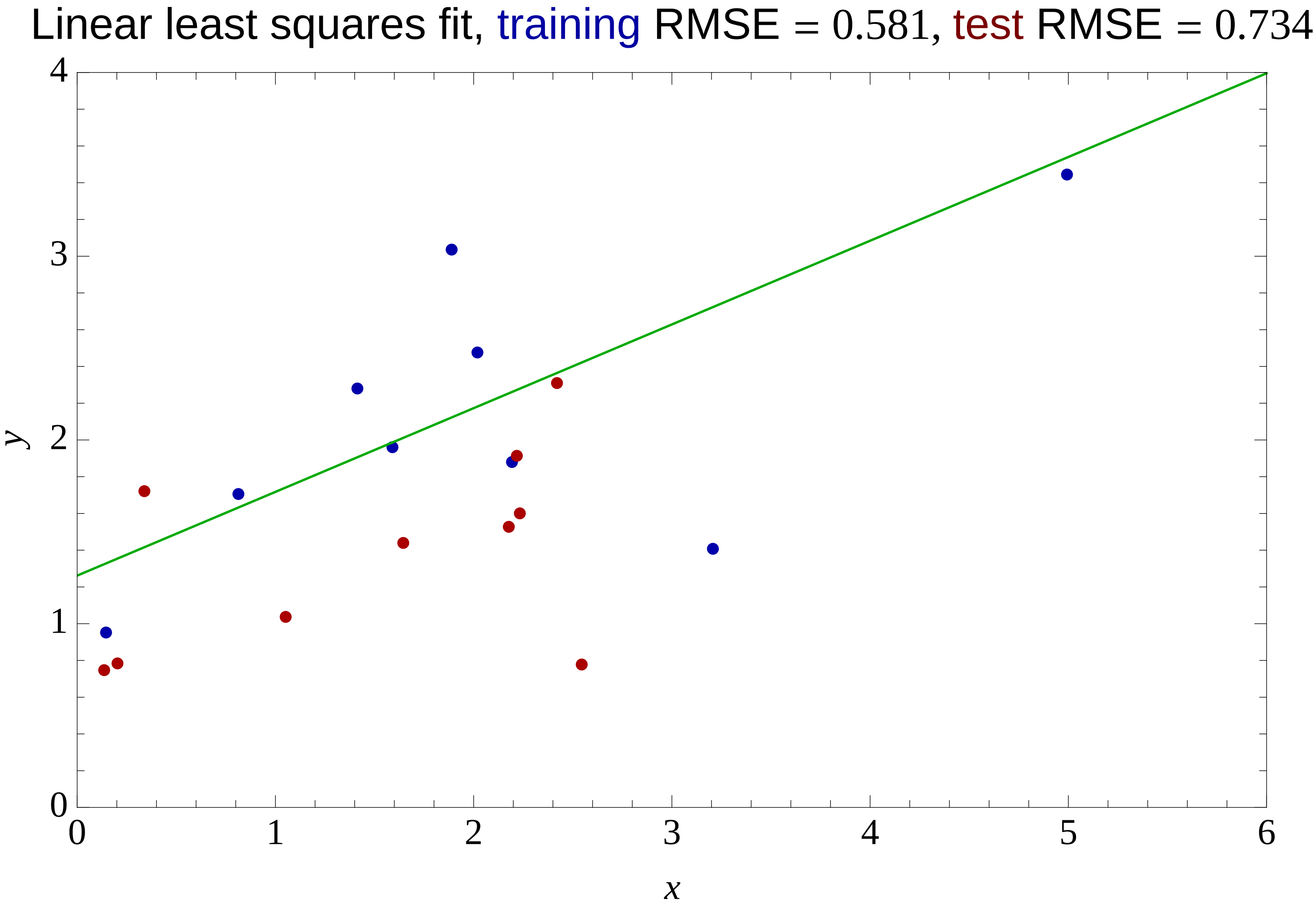
- Capacity control, regularization
  - theoretical results, data-independent, based on sample size and measures of complexity
  - use independent (test) set

# Data generated from an unknown function with unknown noise

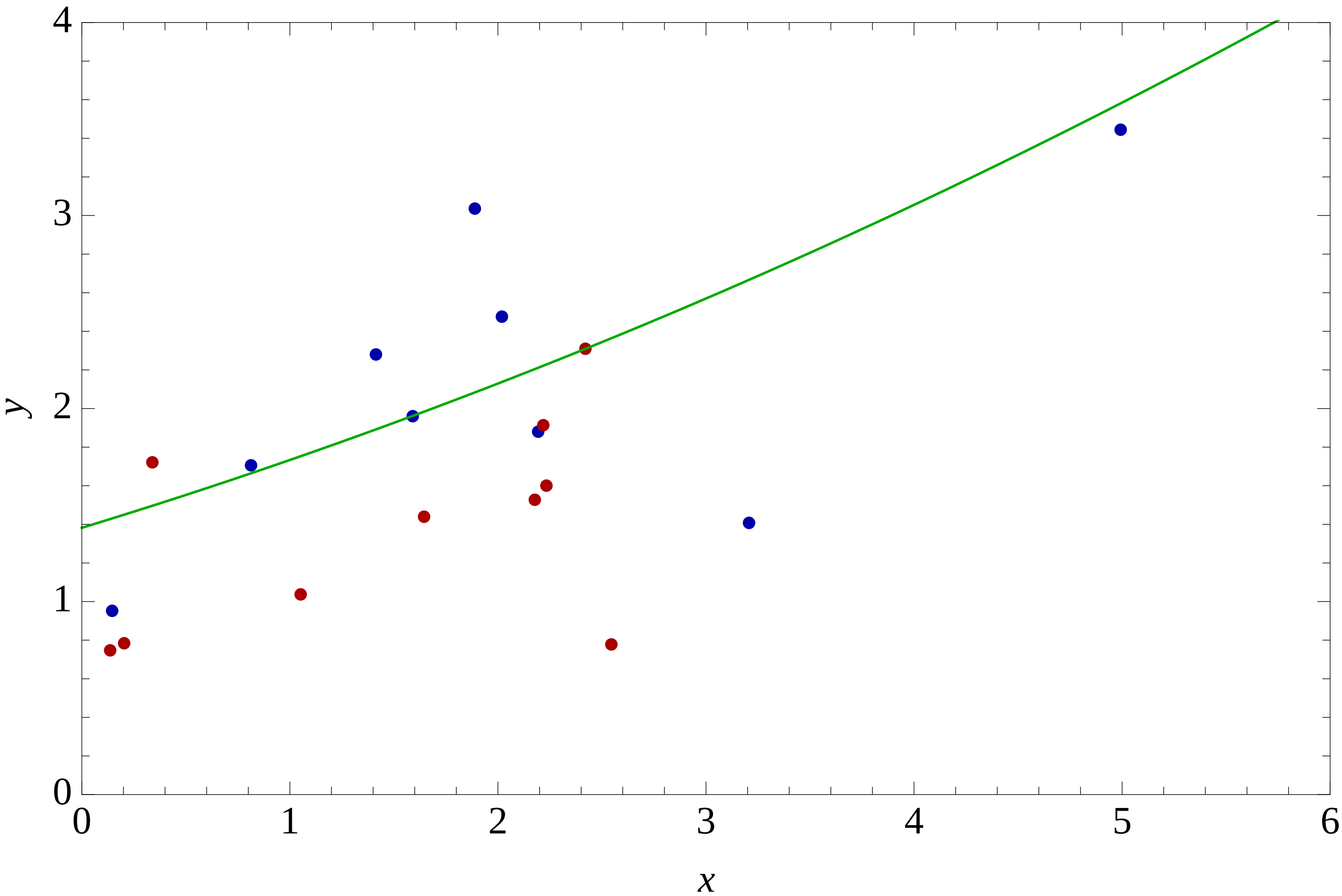


Const. least squares fit, training RMSE = 0.915, test RMSE = 1.067

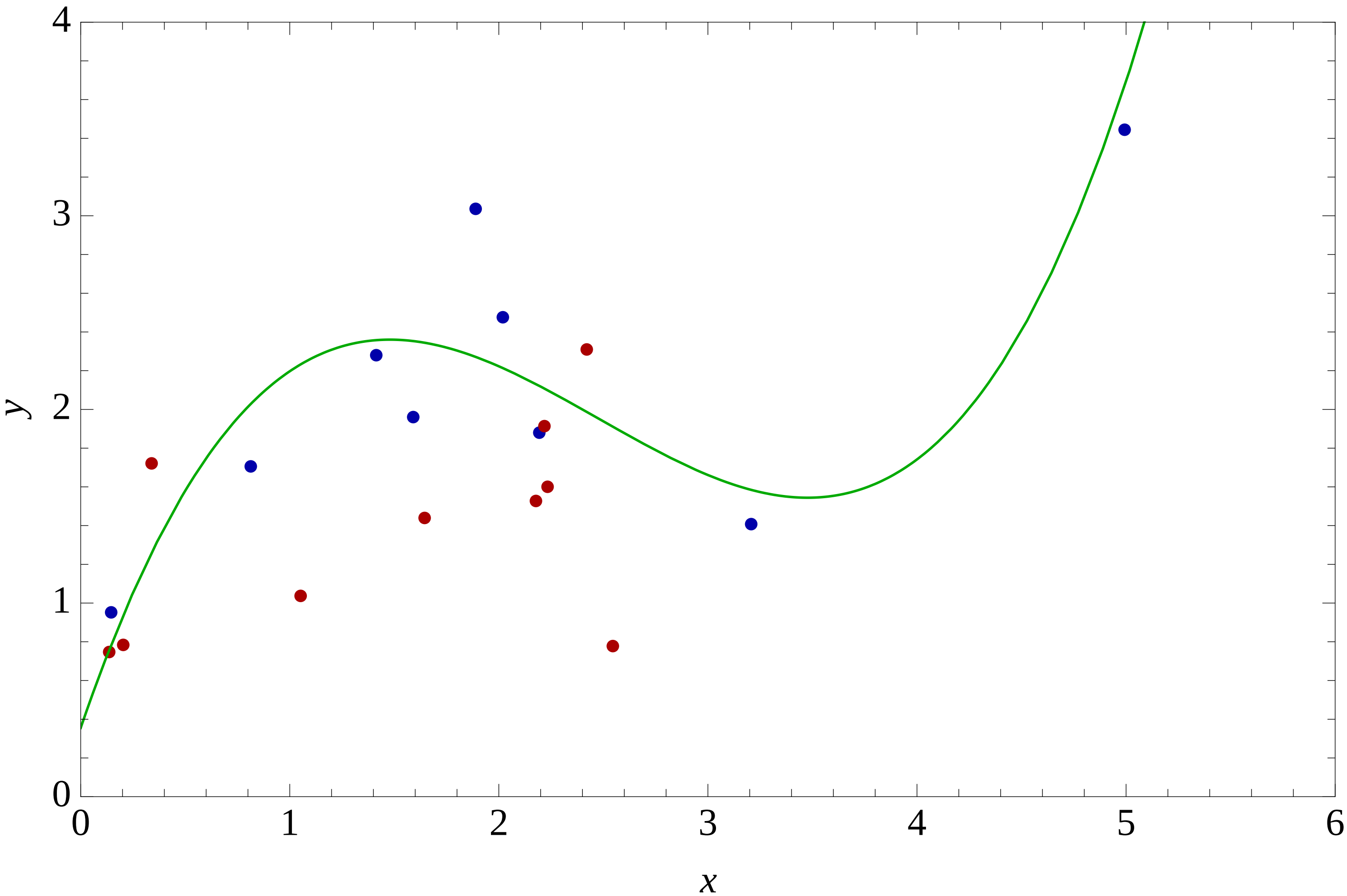



Linear least squares fit, training RMSE = 0.581, test RMSE = 0.734
y
x
0
1
2
3
4
5
6

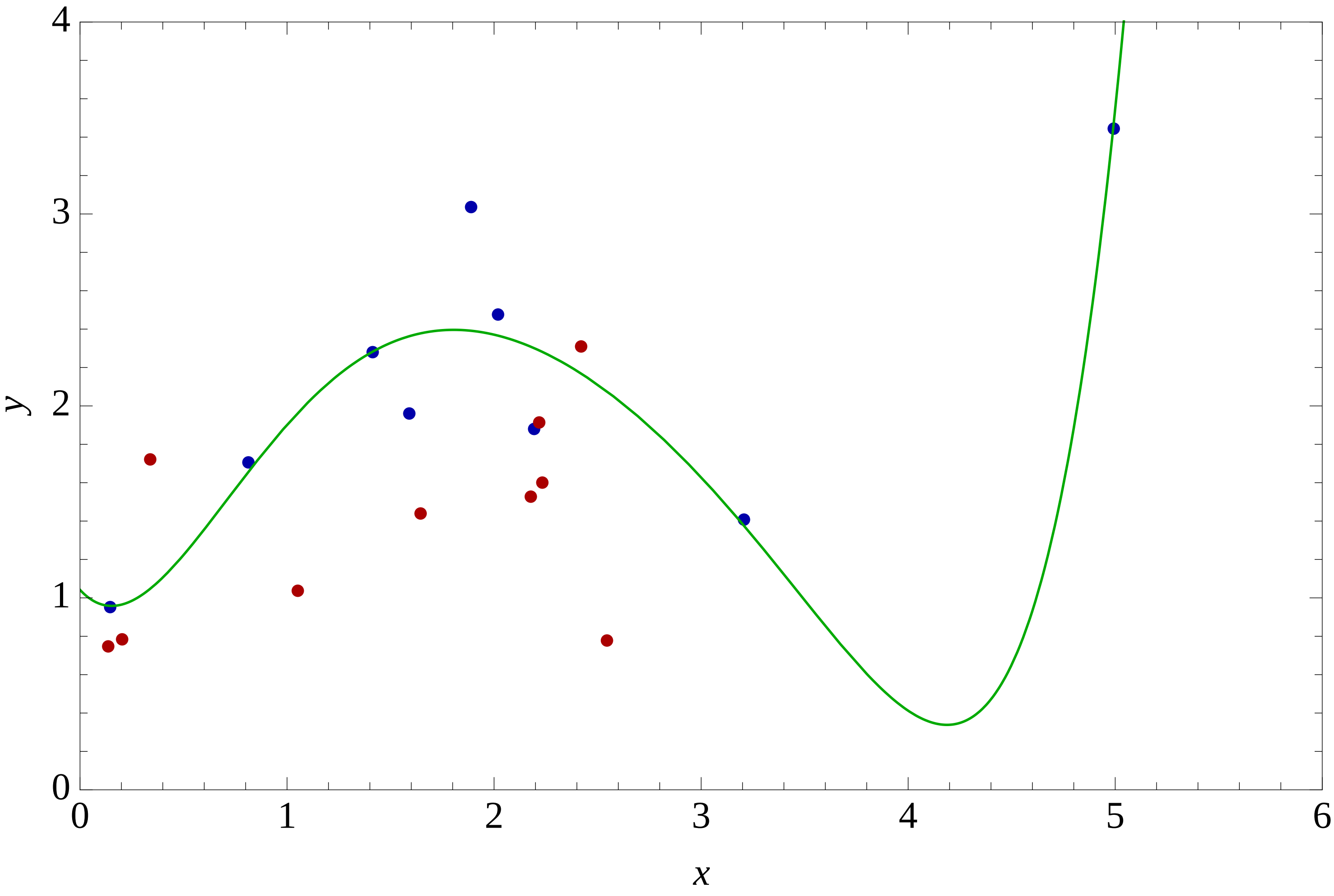
Quadr. least squares fit, training RMSE = 0.579, test RMSE = 0.723



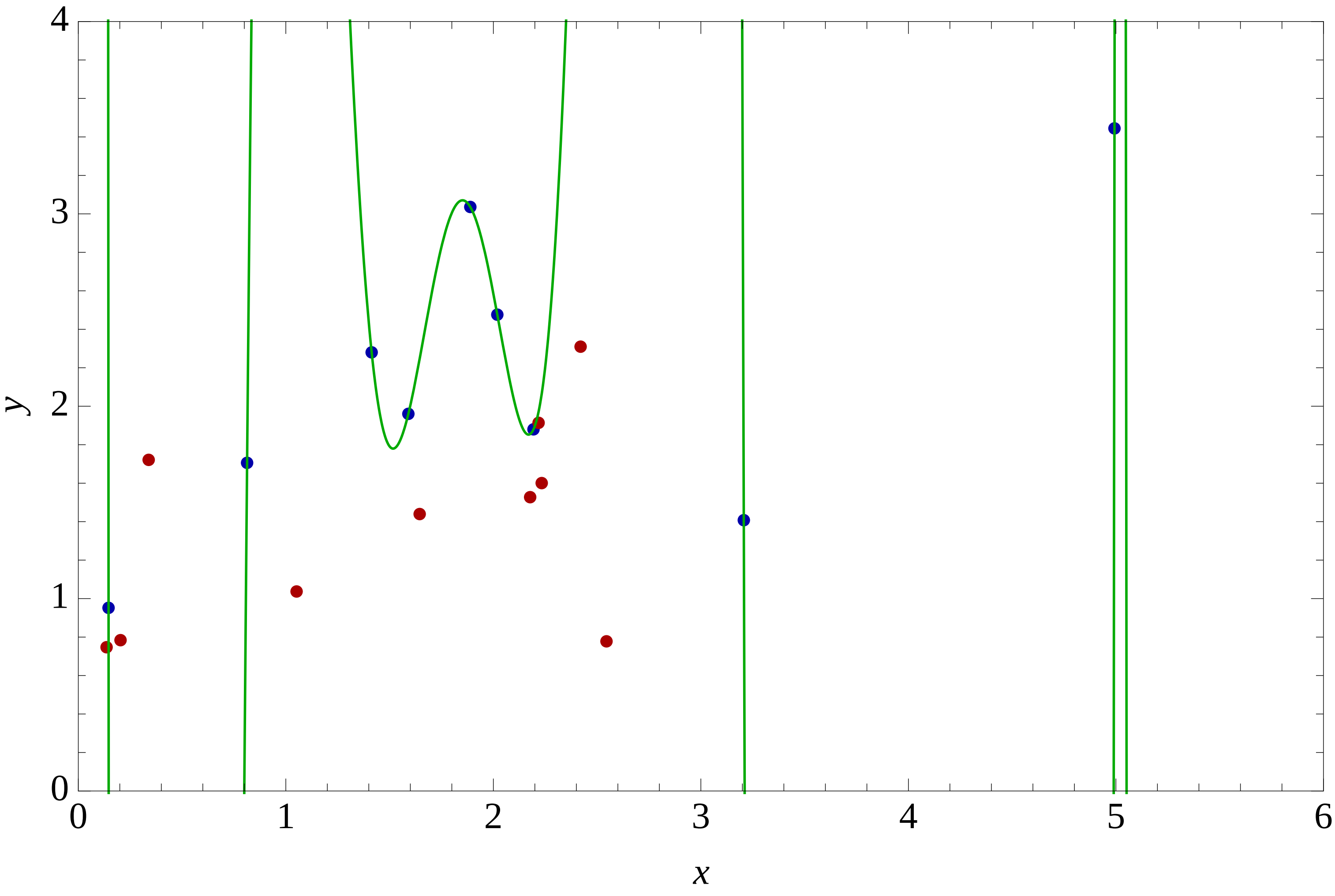
Cubic least squares fit, training RMSE = 0.339, test RMSE = 0.672



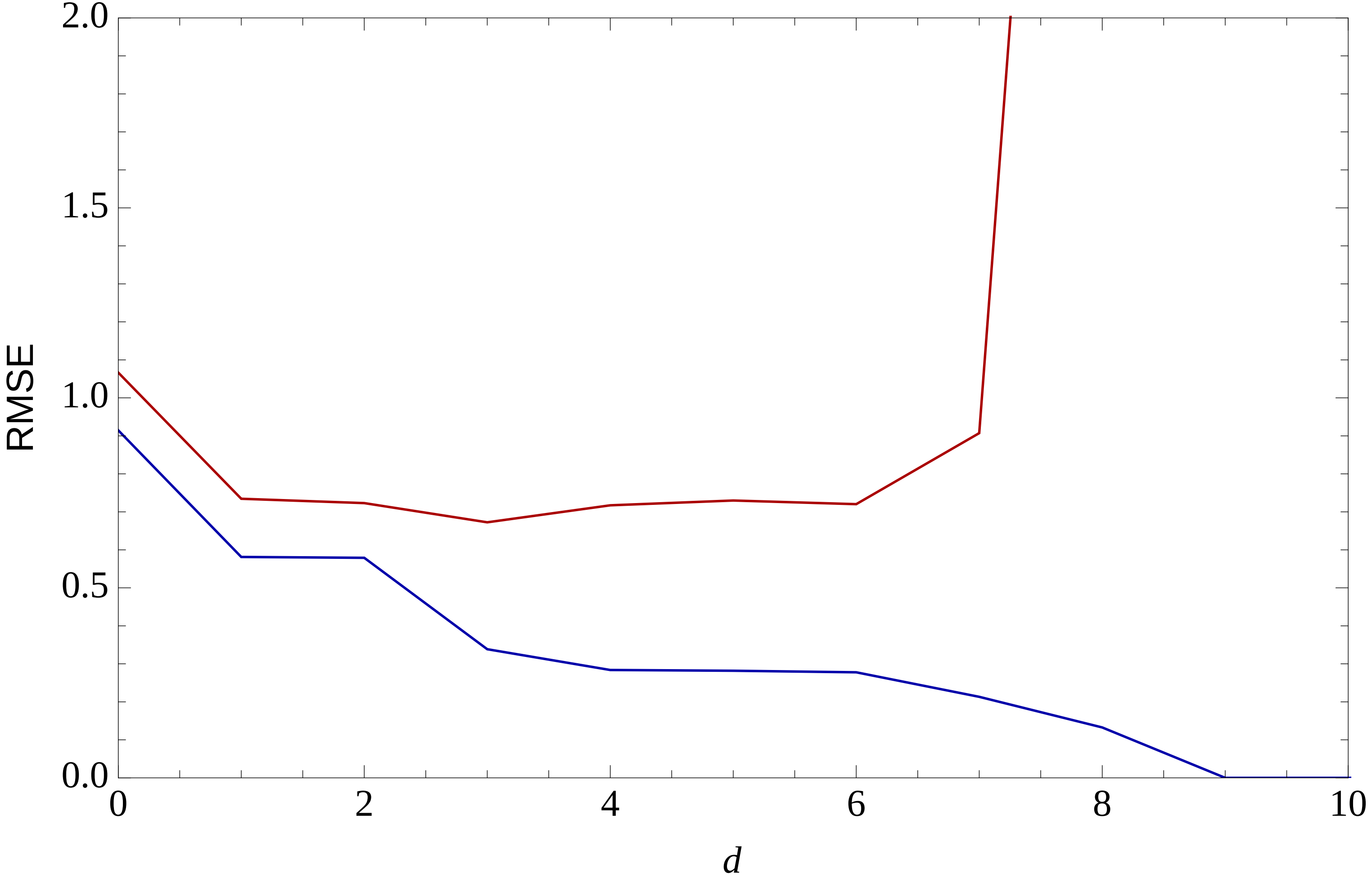
Poli(6) least squares fit, training RMSE = 0.278, test RMSE = 0.72



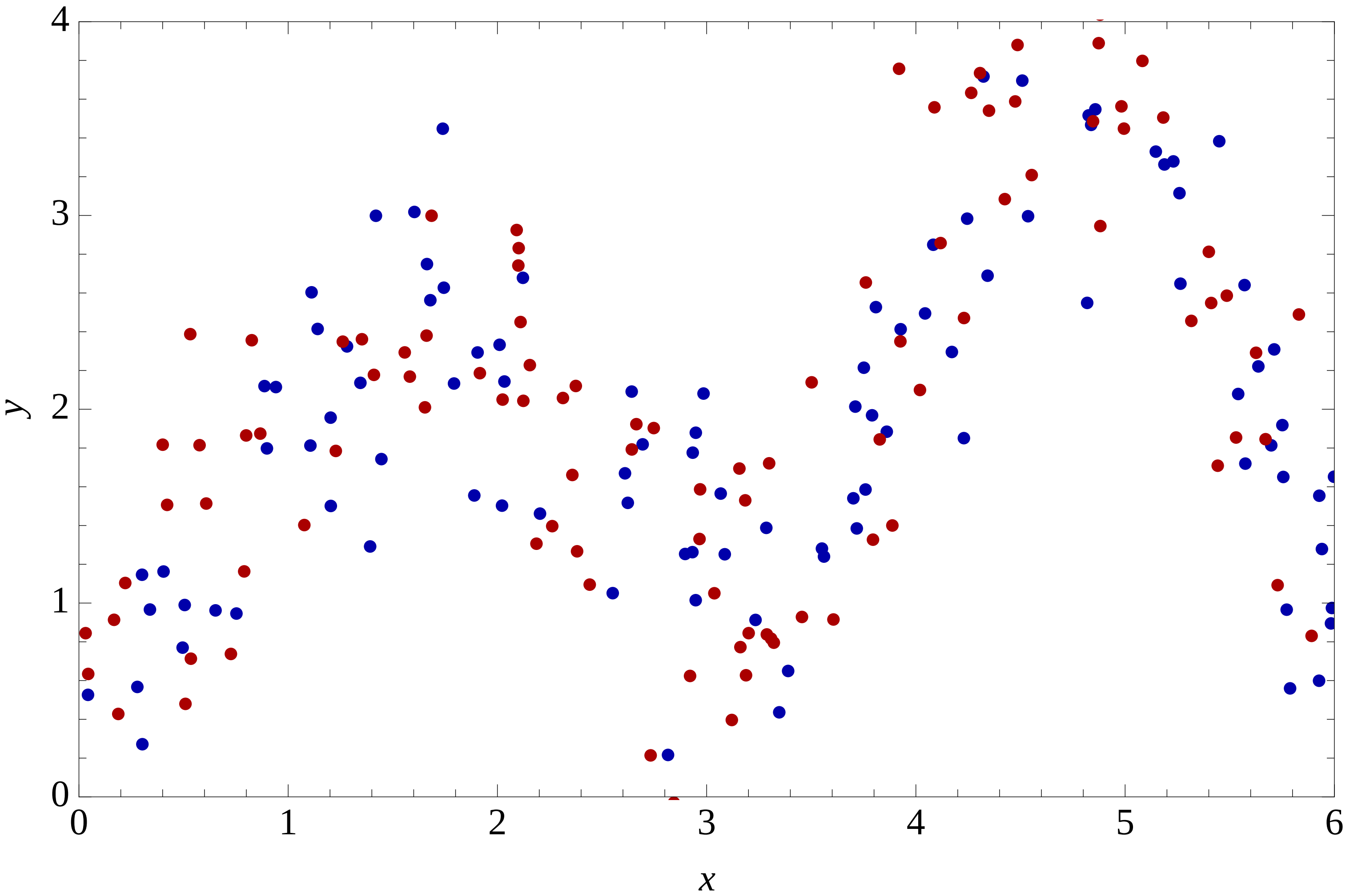
Poli(9) least squares fit, training RMSE = 0, test RMSE = 46.424

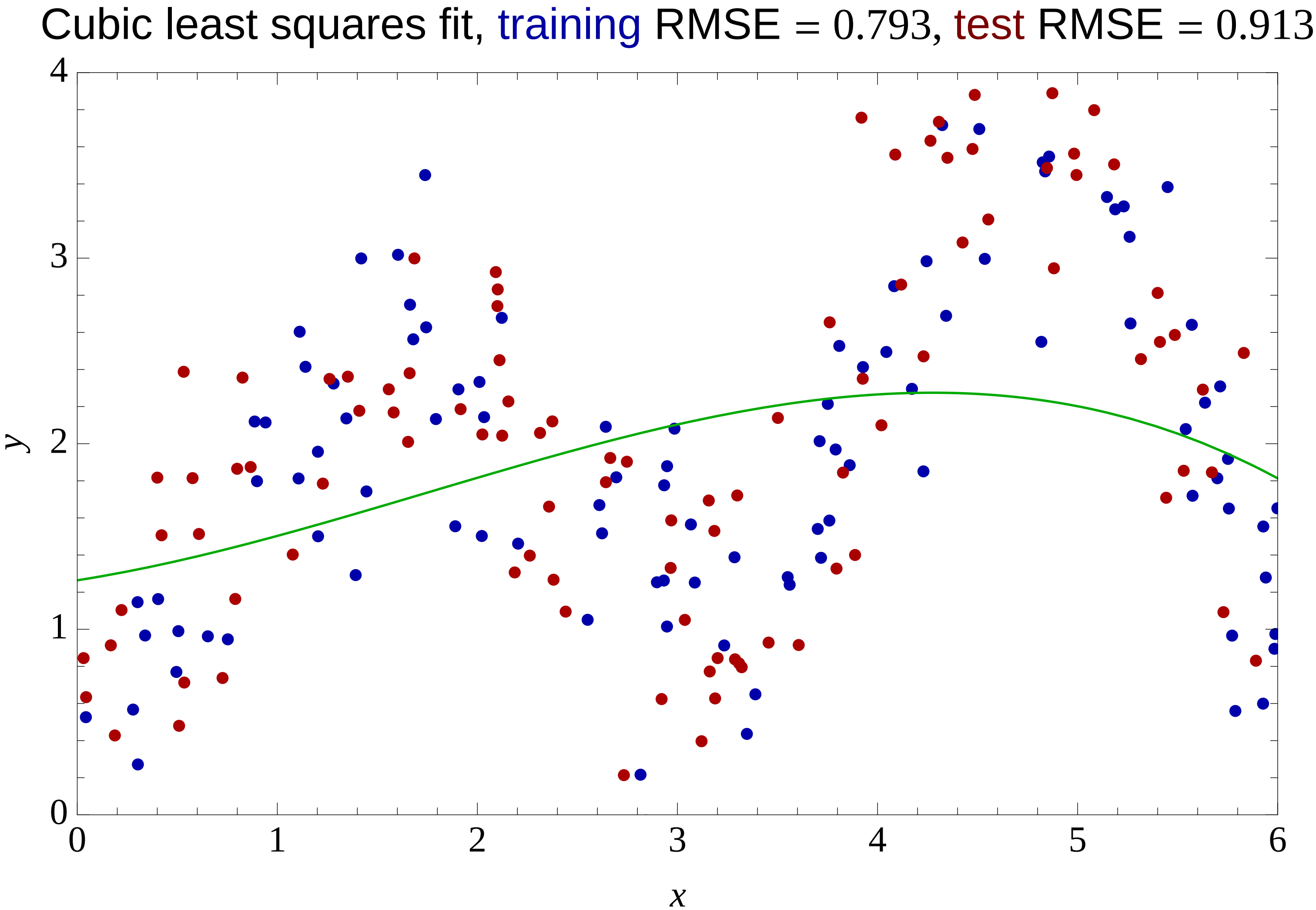


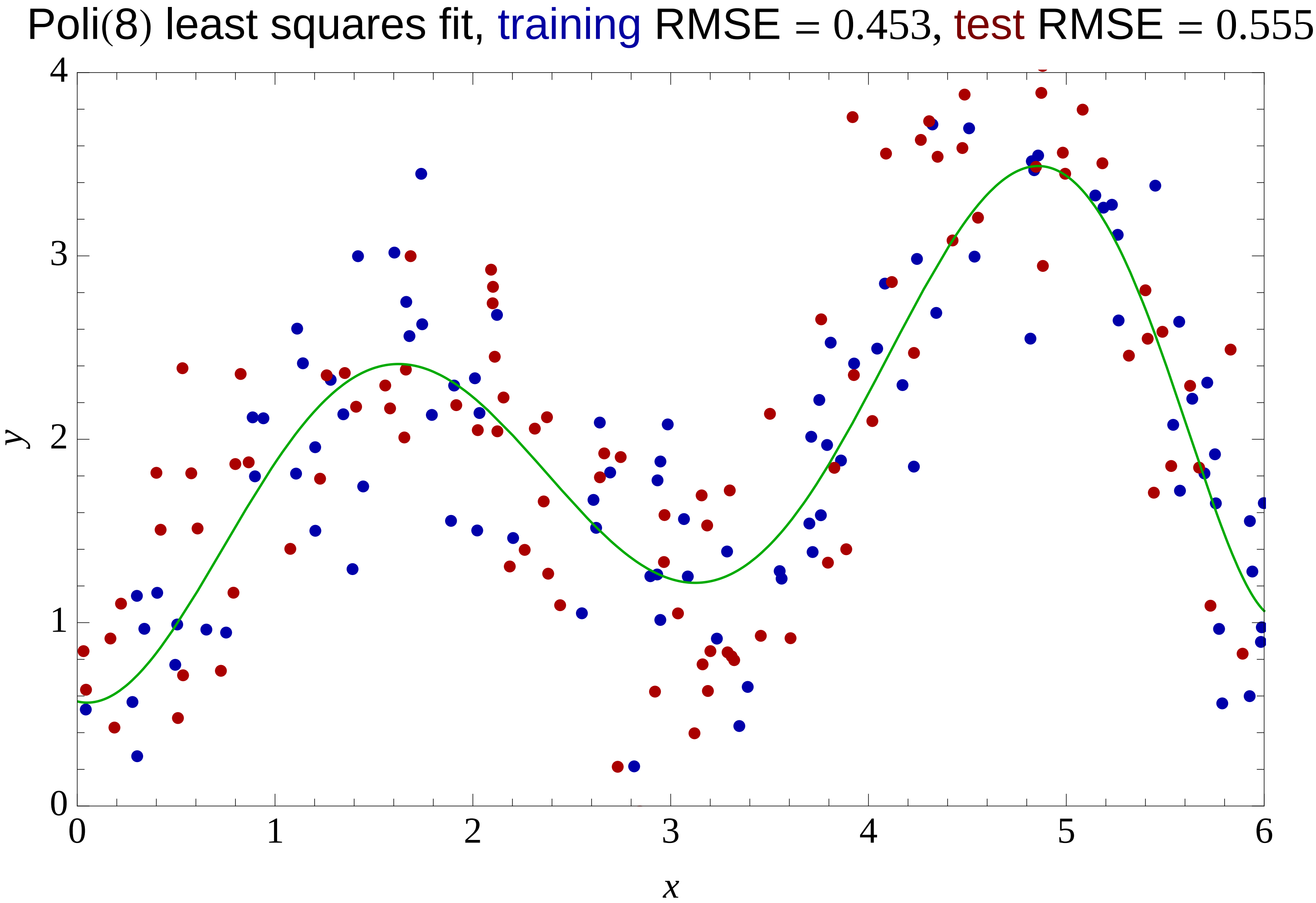
## Training and test RMSE's for polinomial fits of different degrees



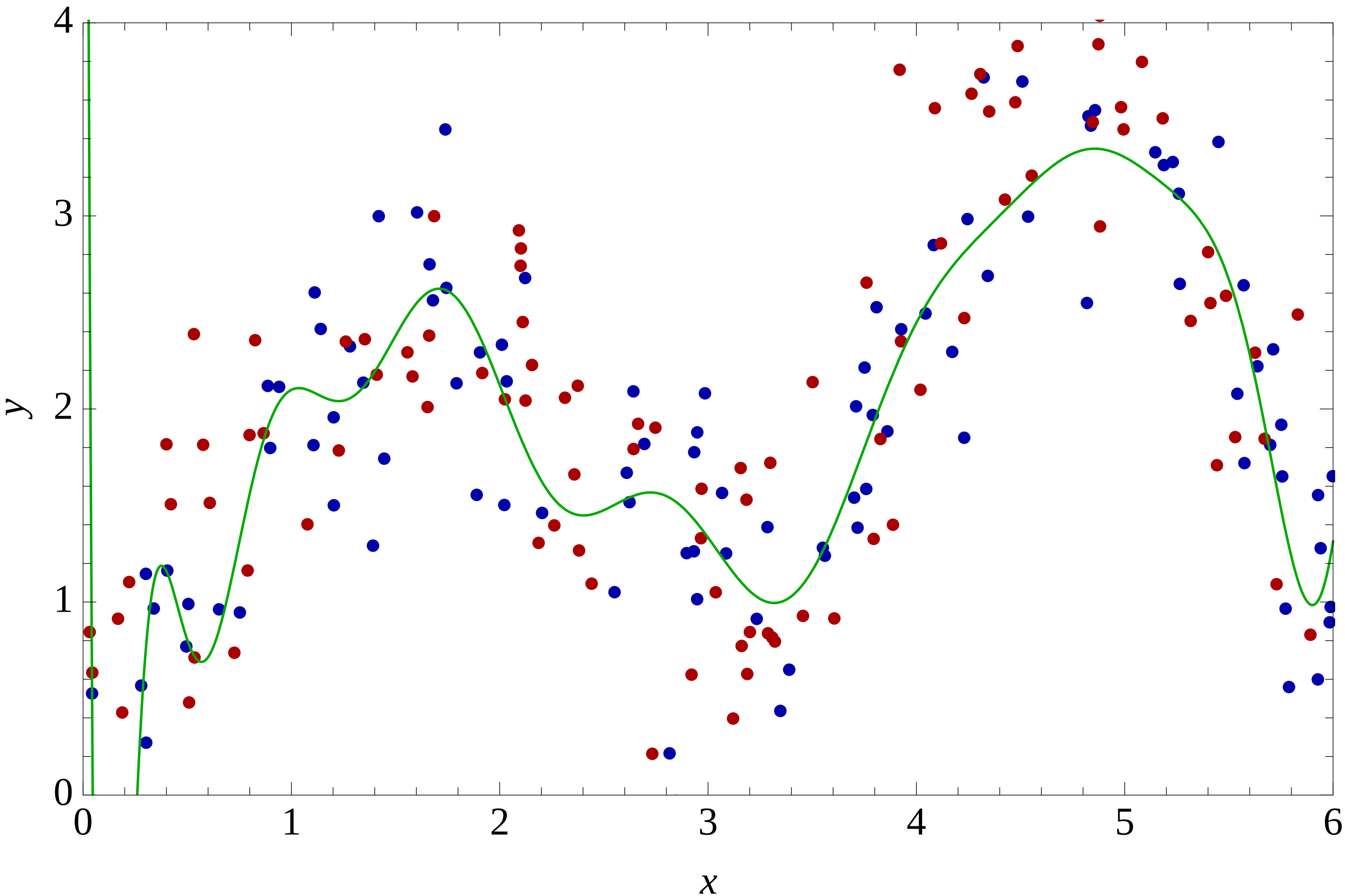
# Data generated from an unknown function with unknown noise



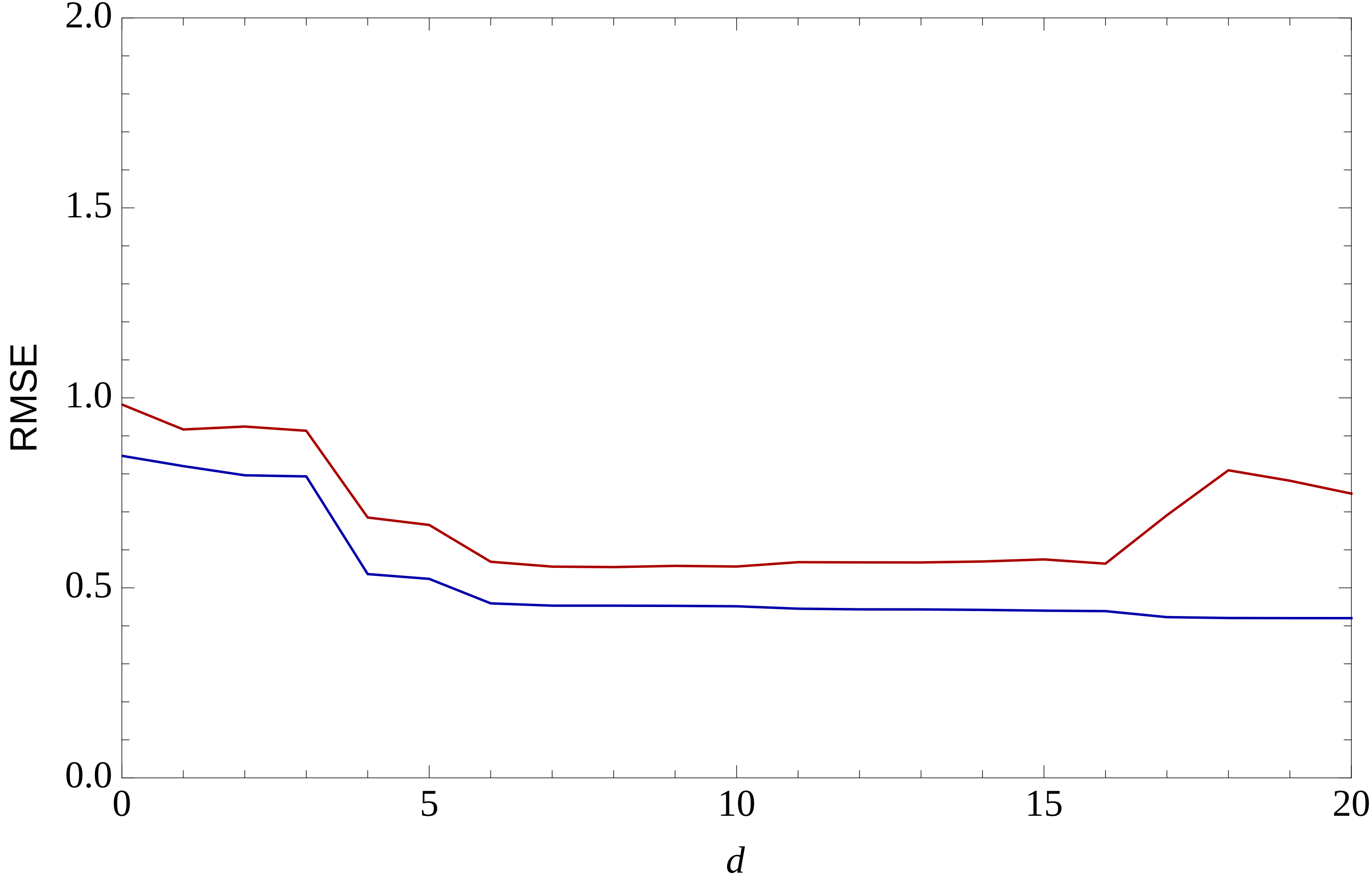

Cubic least squares fit, training RMSE = 0.793, test RMSE = 0.913
y
x
0
1
2
3
4
5
6


Poli(8) least squares fit, training RMSE = 0.453, test RMSE = 0.555
y
x
0
1
2
3
4
5
6

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



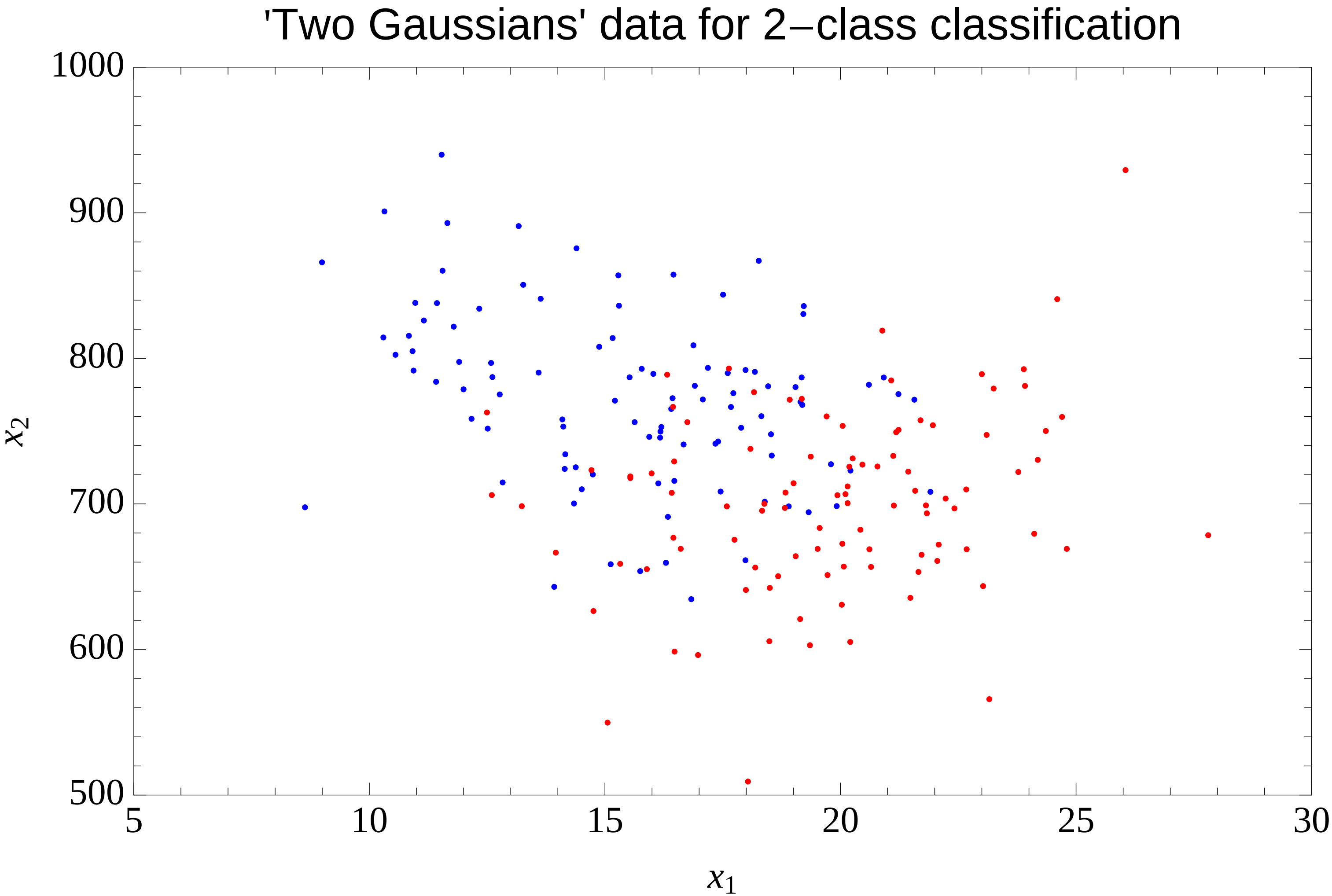
Training and test RMSE's for polinomial fits of different degrees



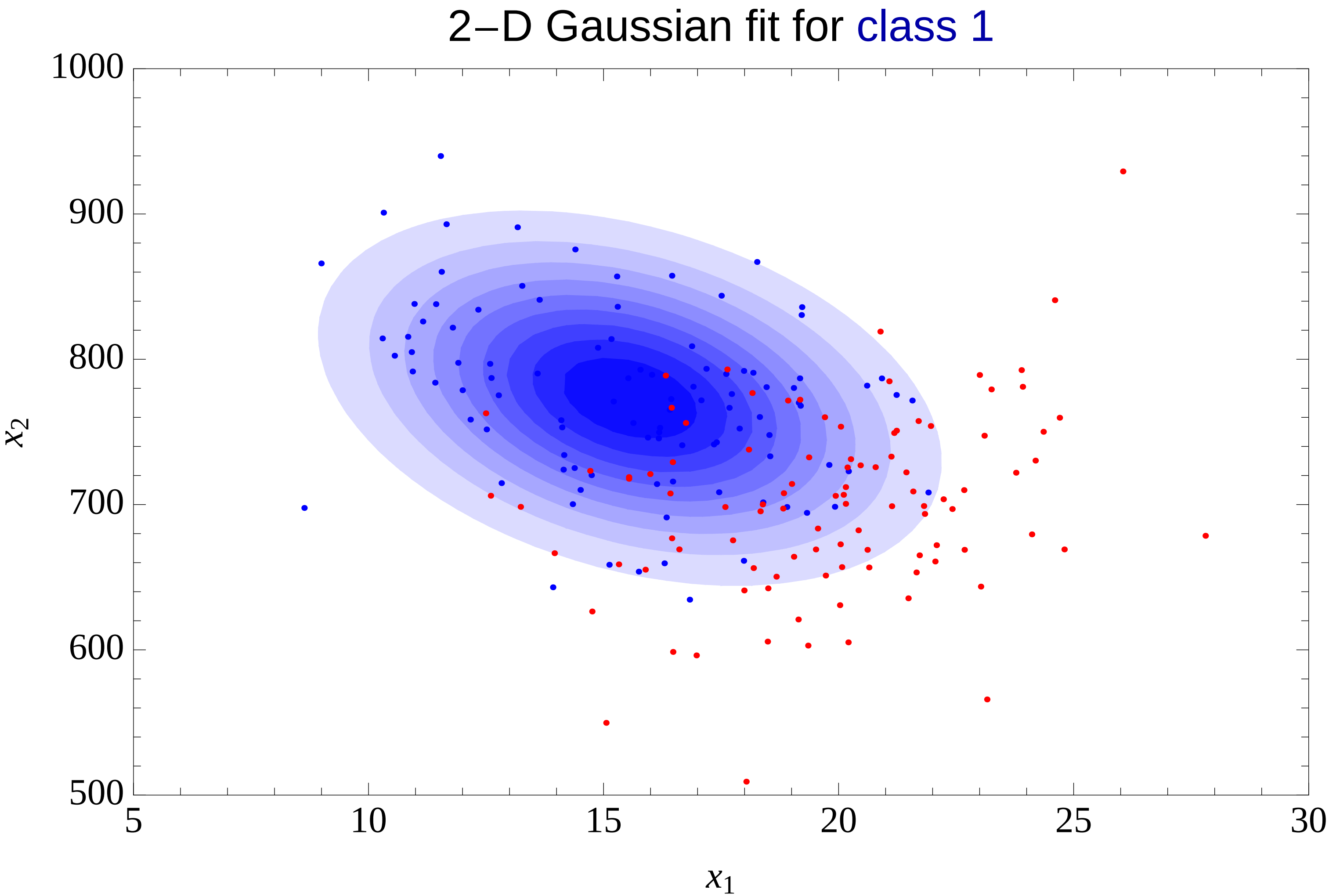
# Non-parametric fitting

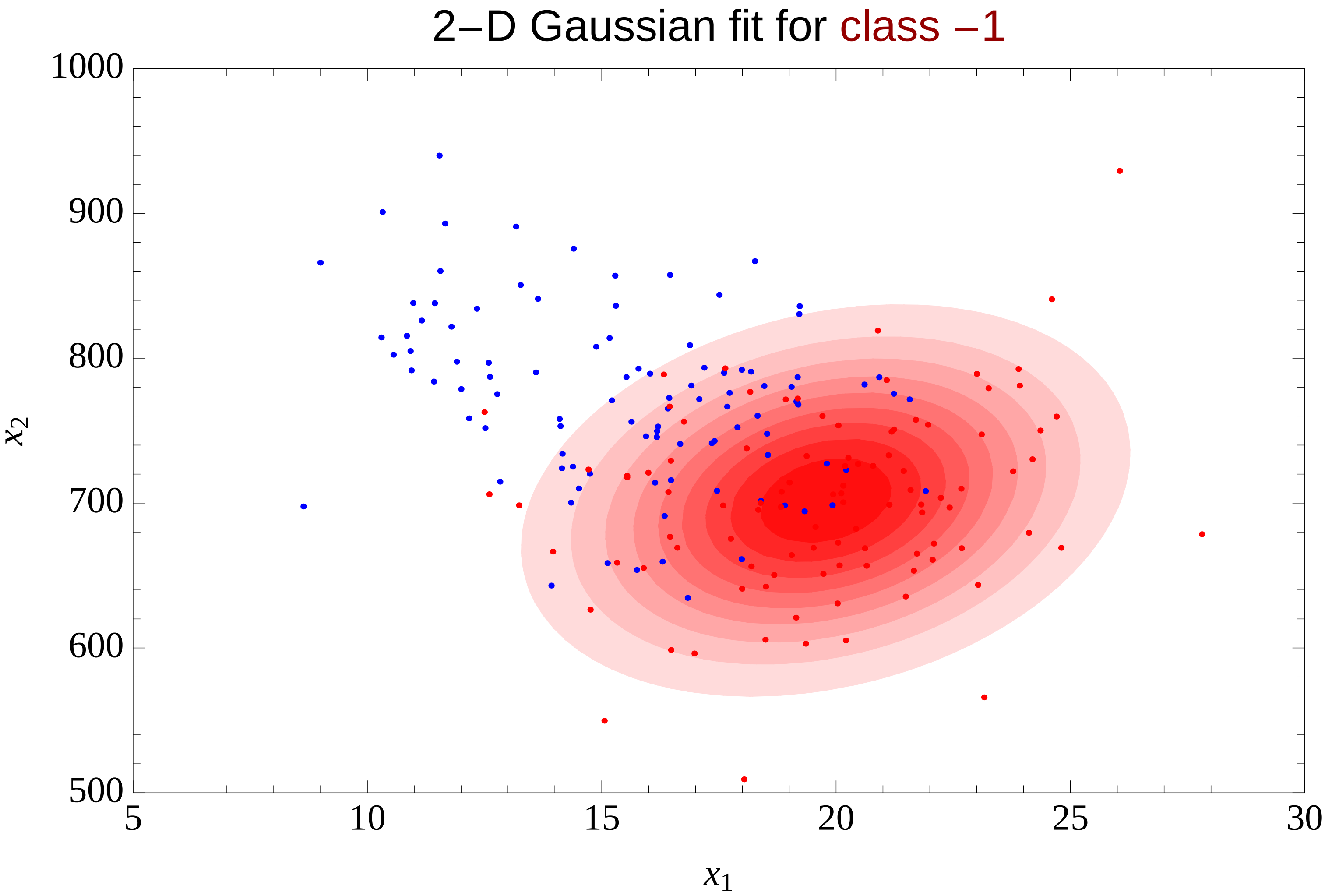


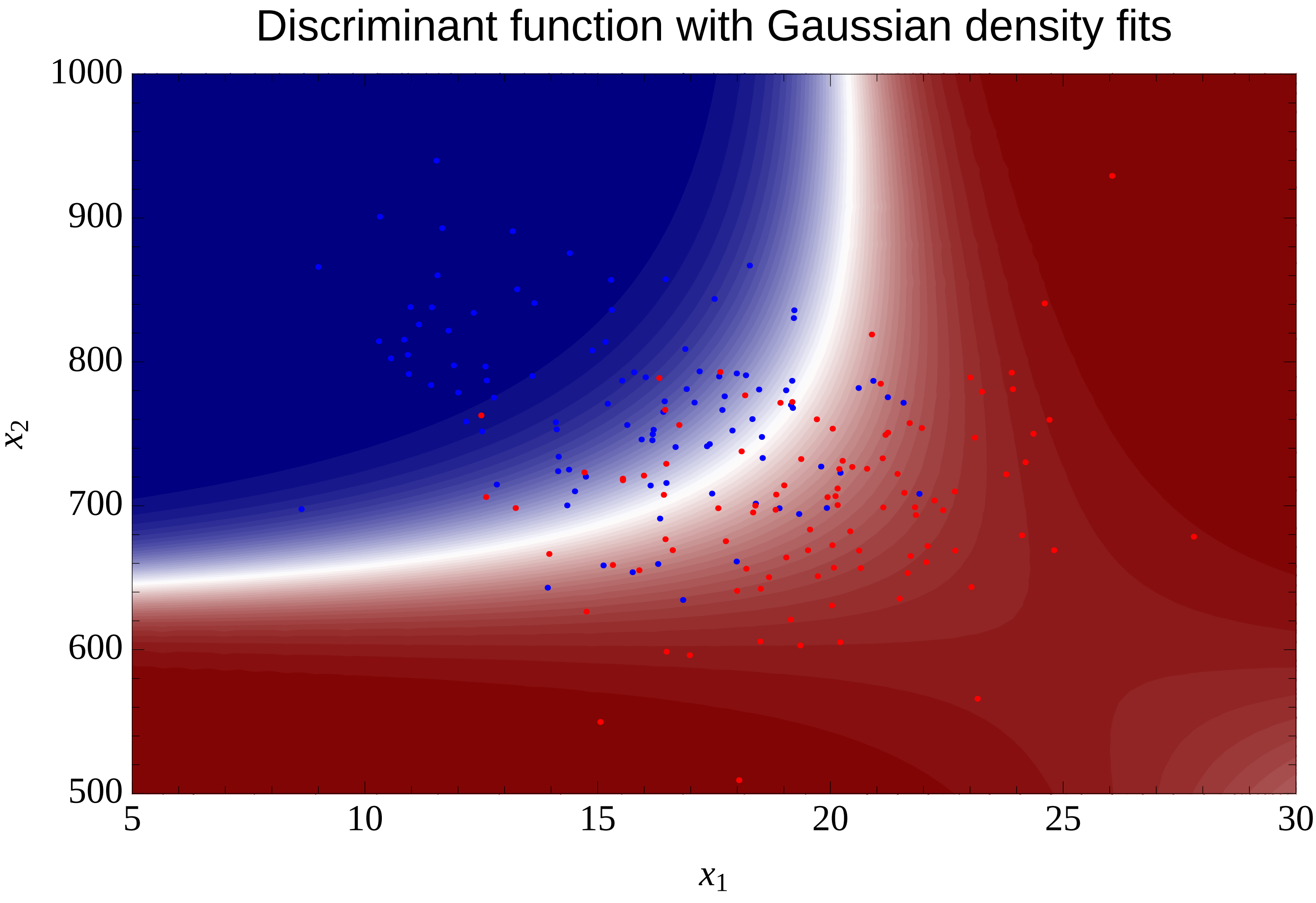
- Capacity control, regularization
  - trade-off between approximation error and estimation error
  - complexity grows with data size
  - no need to correctly guess the function class



'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
5
10
15
20
25
30
$x_1$
$x_2$

# Classification



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

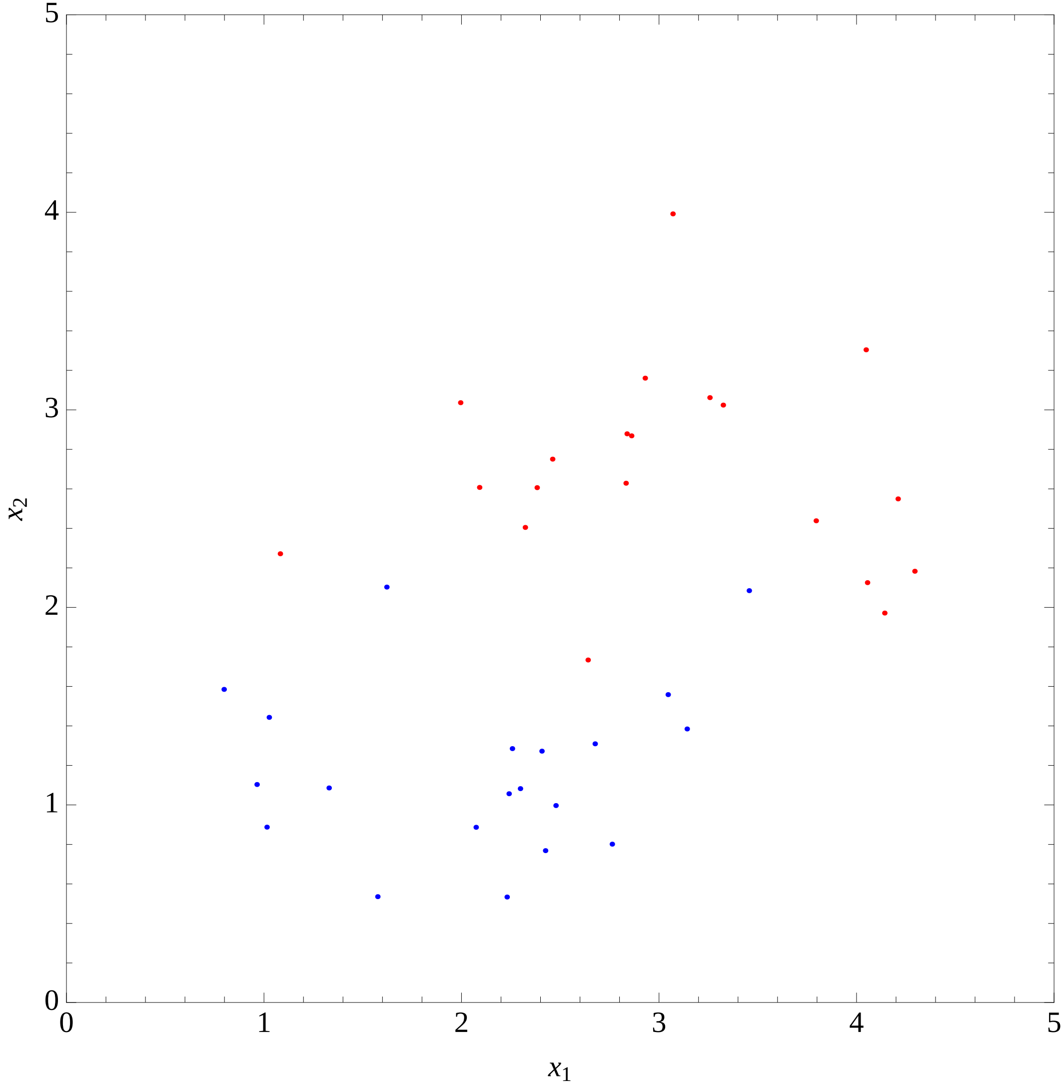
- Bayes theorem:

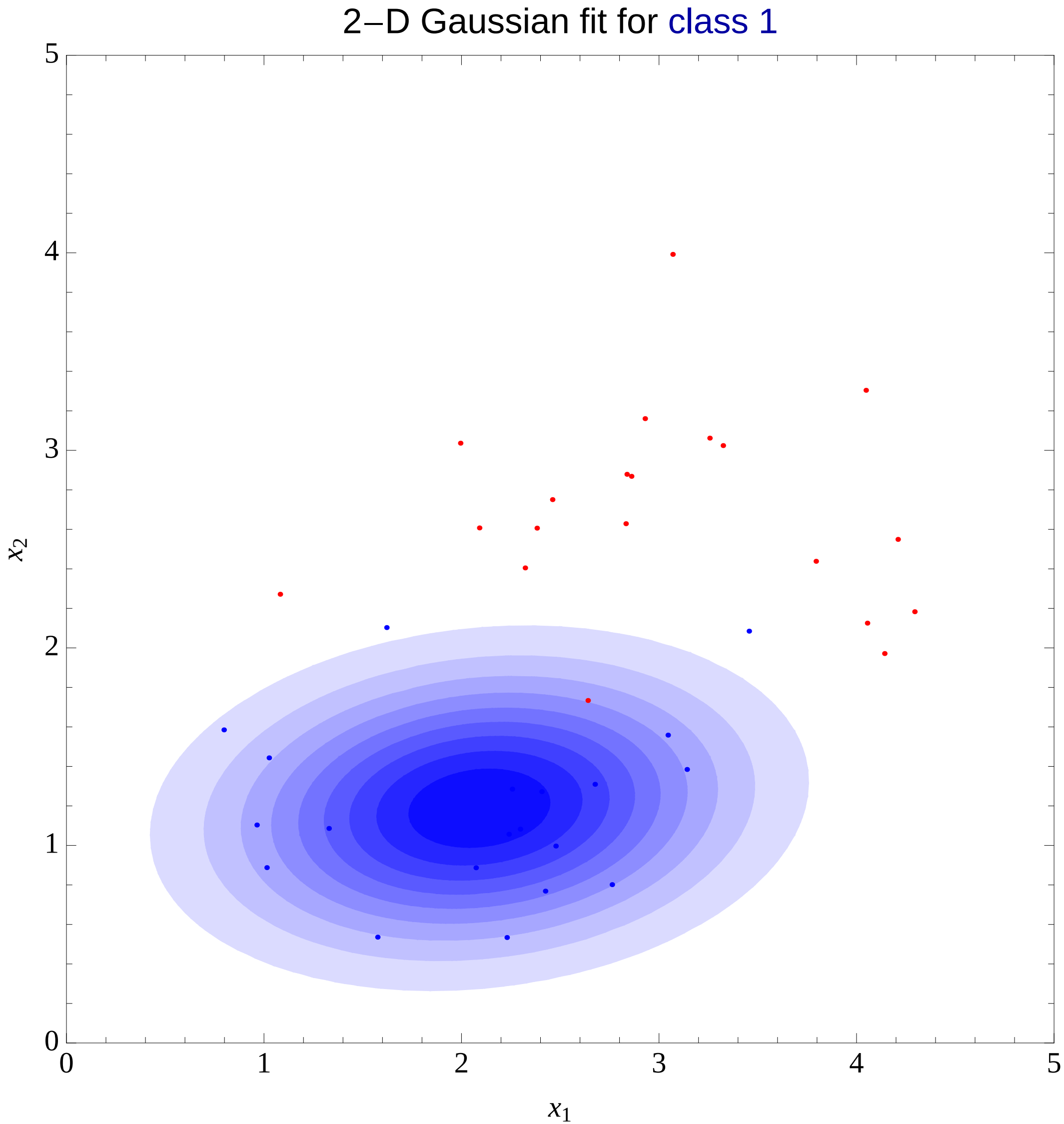
$$P(Y=1|\mathbf{x}) = \frac{p(\mathbf{x}|Y=1)P(Y=1)}{p(\mathbf{x})} \propto p(\mathbf{x}|Y=1)P(Y=1)$$

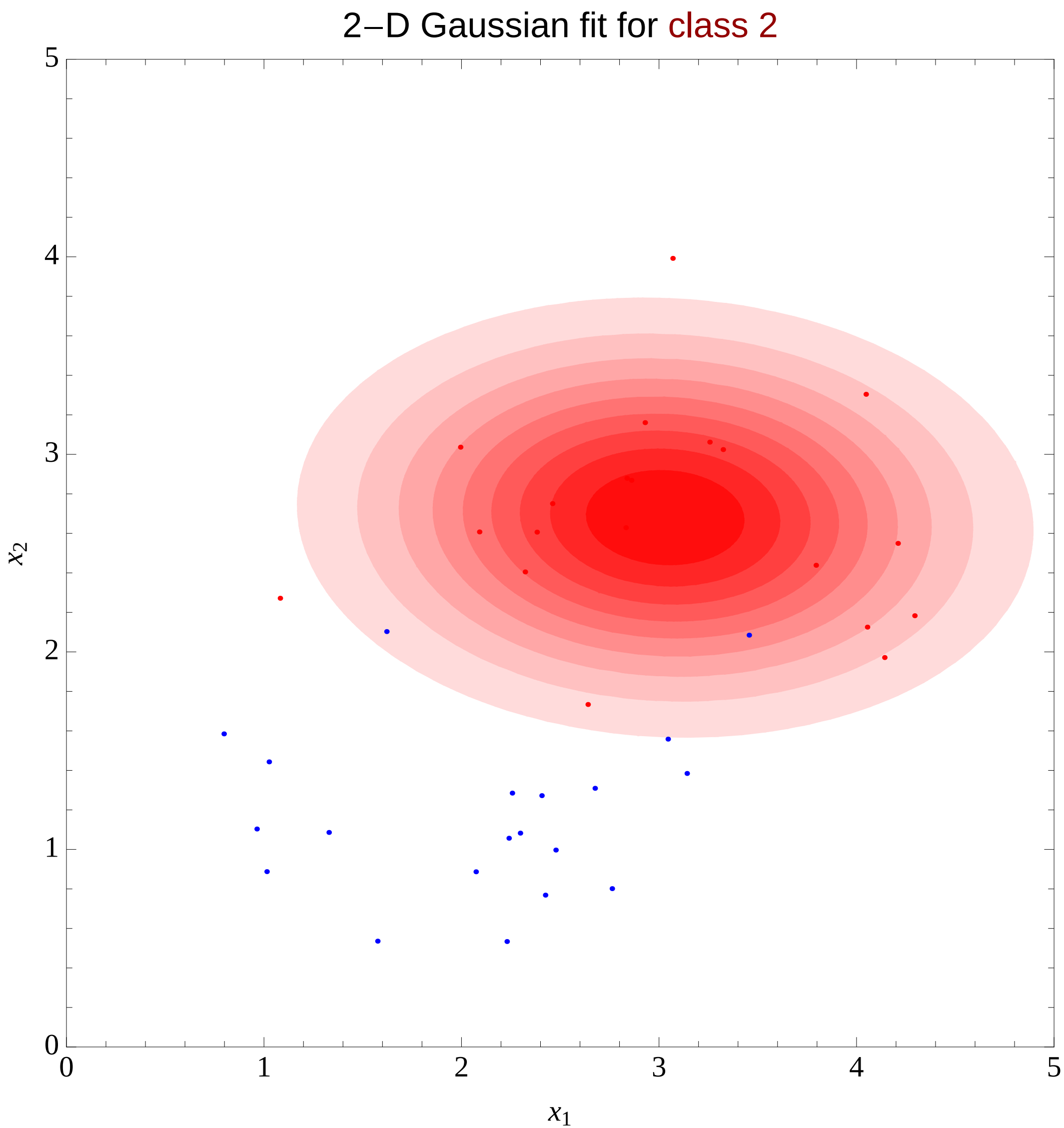
  - Decision:

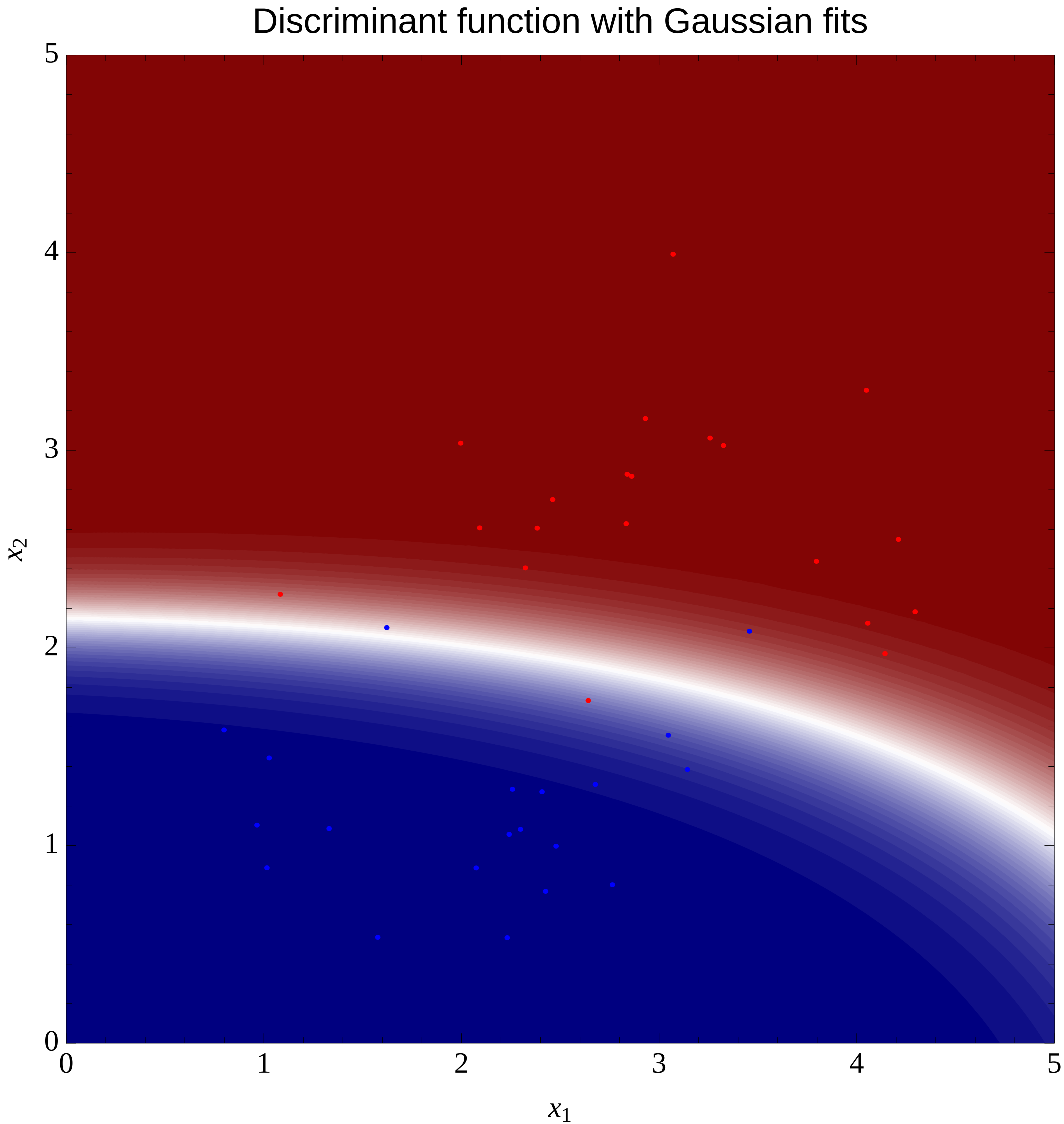
$$g(\mathbf{x}) = \begin{cases} 1 & \text{if } \frac{p(\mathbf{x}|Y=1)P(Y=1)}{p(\mathbf{x}|Y=-1)P(Y=-1)} > 1, \\ -1 & \text{otherwise.} \end{cases}$$

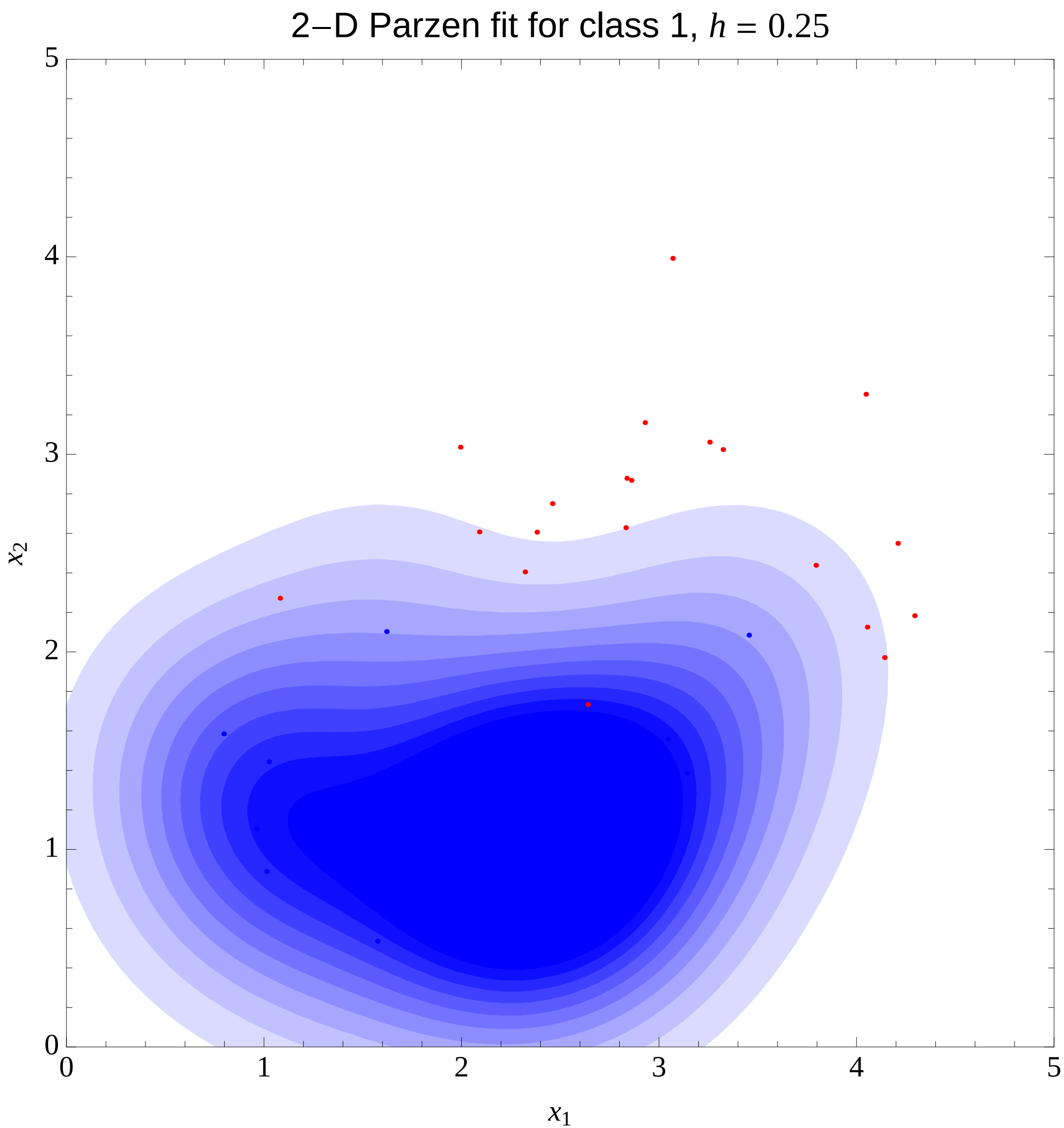
'Two moons' data

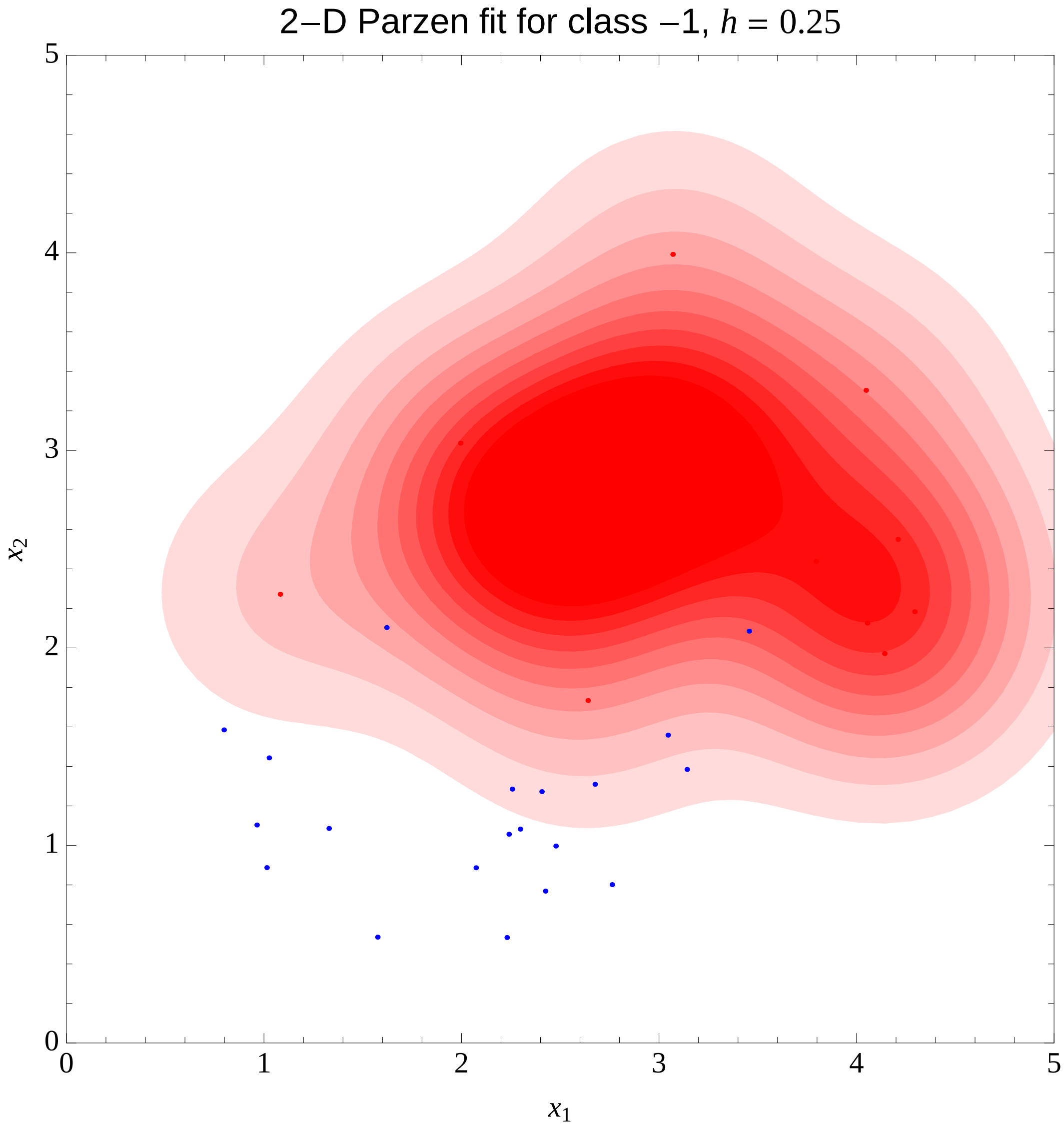


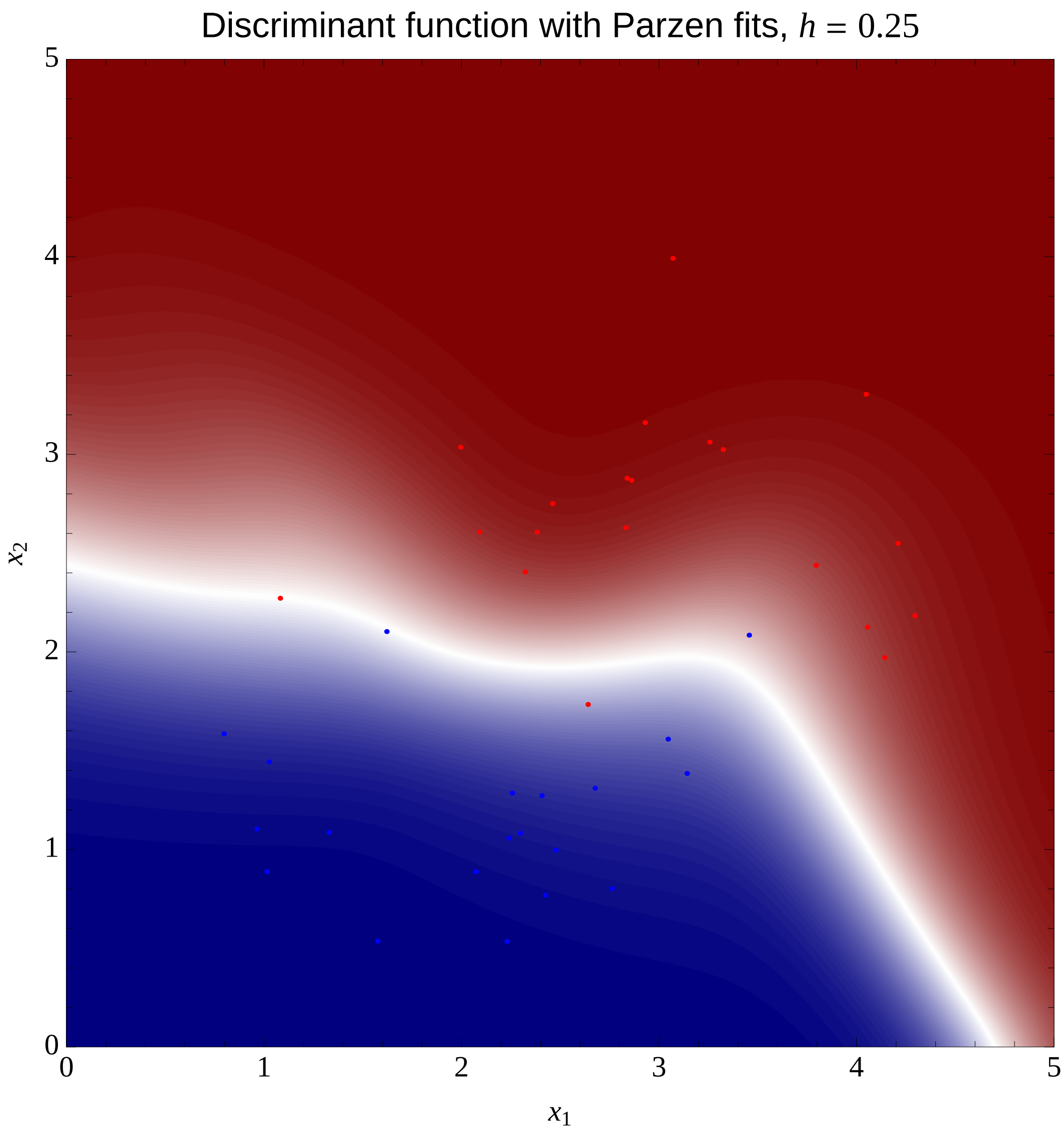

2−D Gaussian fit for class 1
5
4
3
2
1
0
0
1
2
3
4
5
$x_2$
$x_1$


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

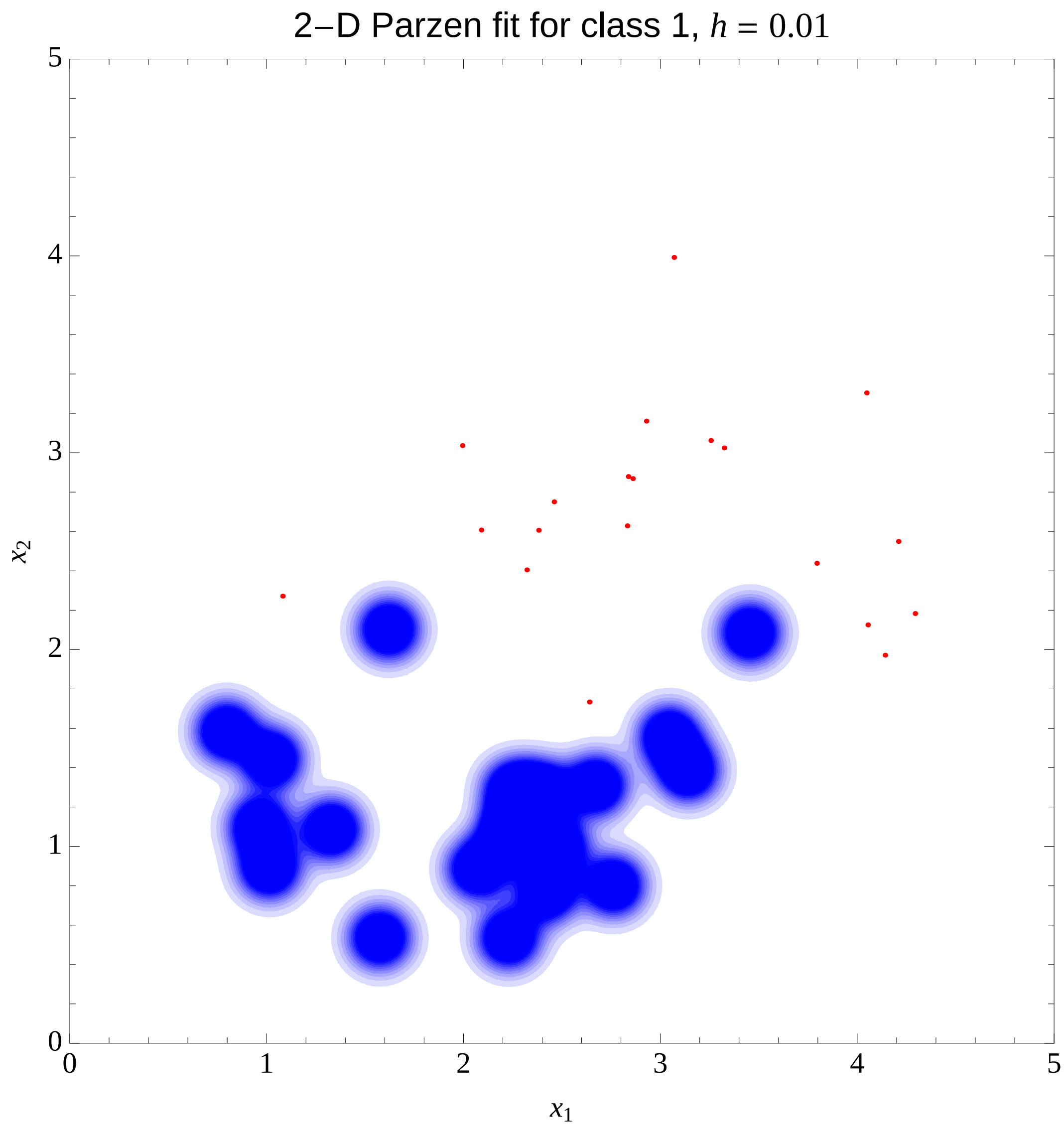

Discriminant function with Gaussian fits
5
4
3
2
1
0
0
1
2
3
4
5
$x_2$
$x_1$

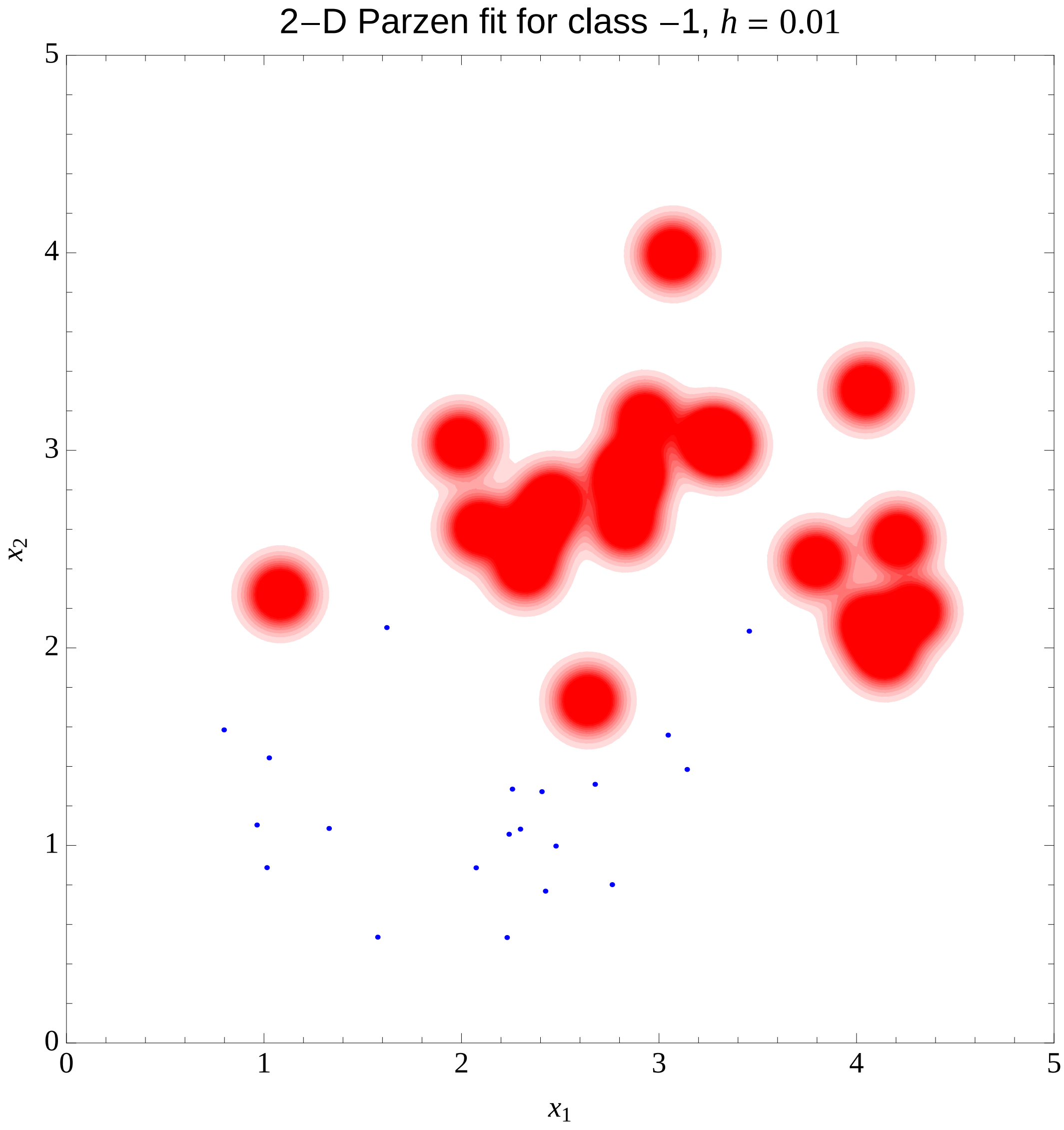

2−D Parzen fit for class 1, $h$ = 0.25
$x_1$
$x_2$
0
1
2
3
4
5
0
1
2
3
4
5

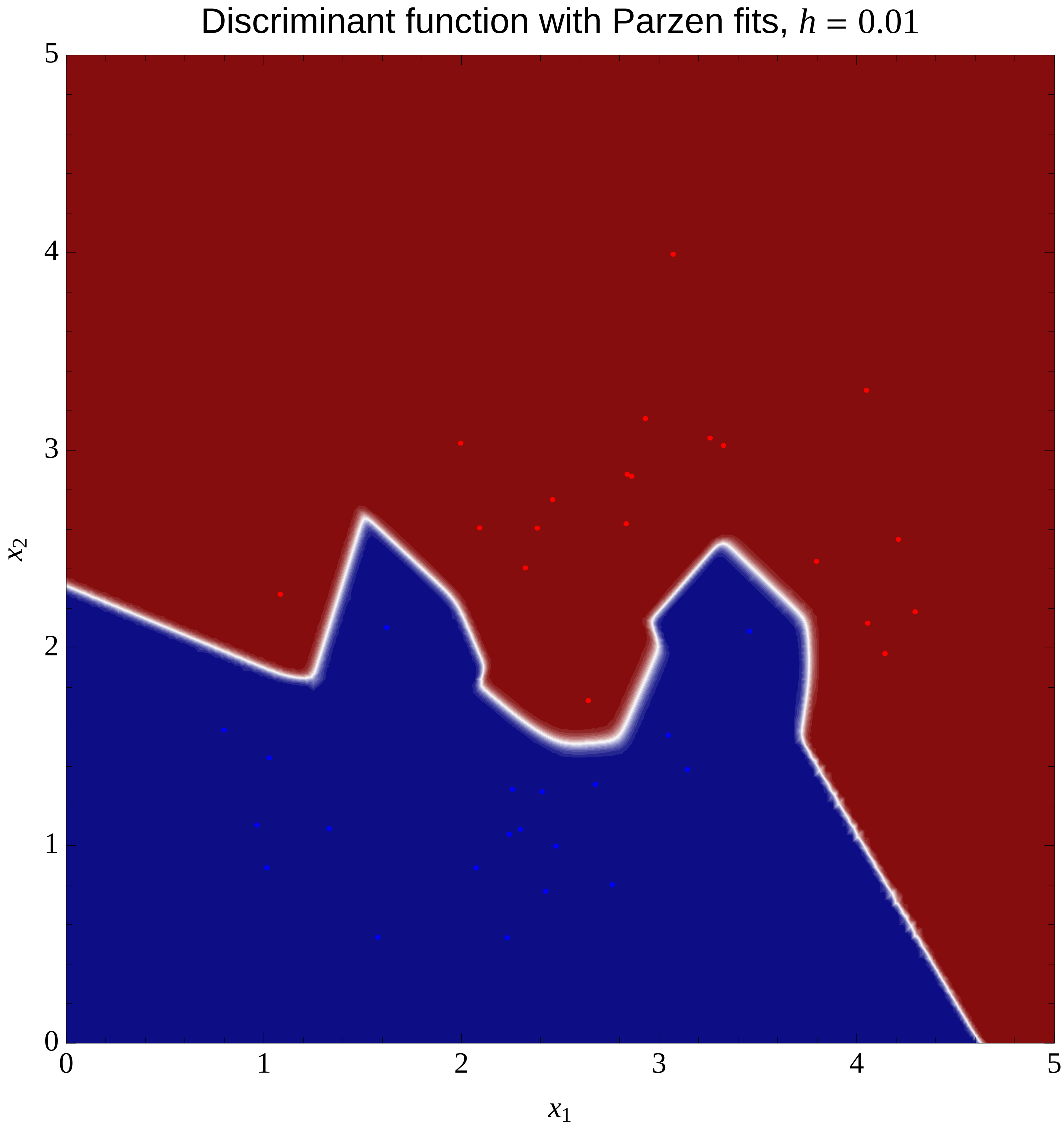

2−D Parzen fit for class −1, h = 0.25
0
1
2
3
4
5
x1
x2

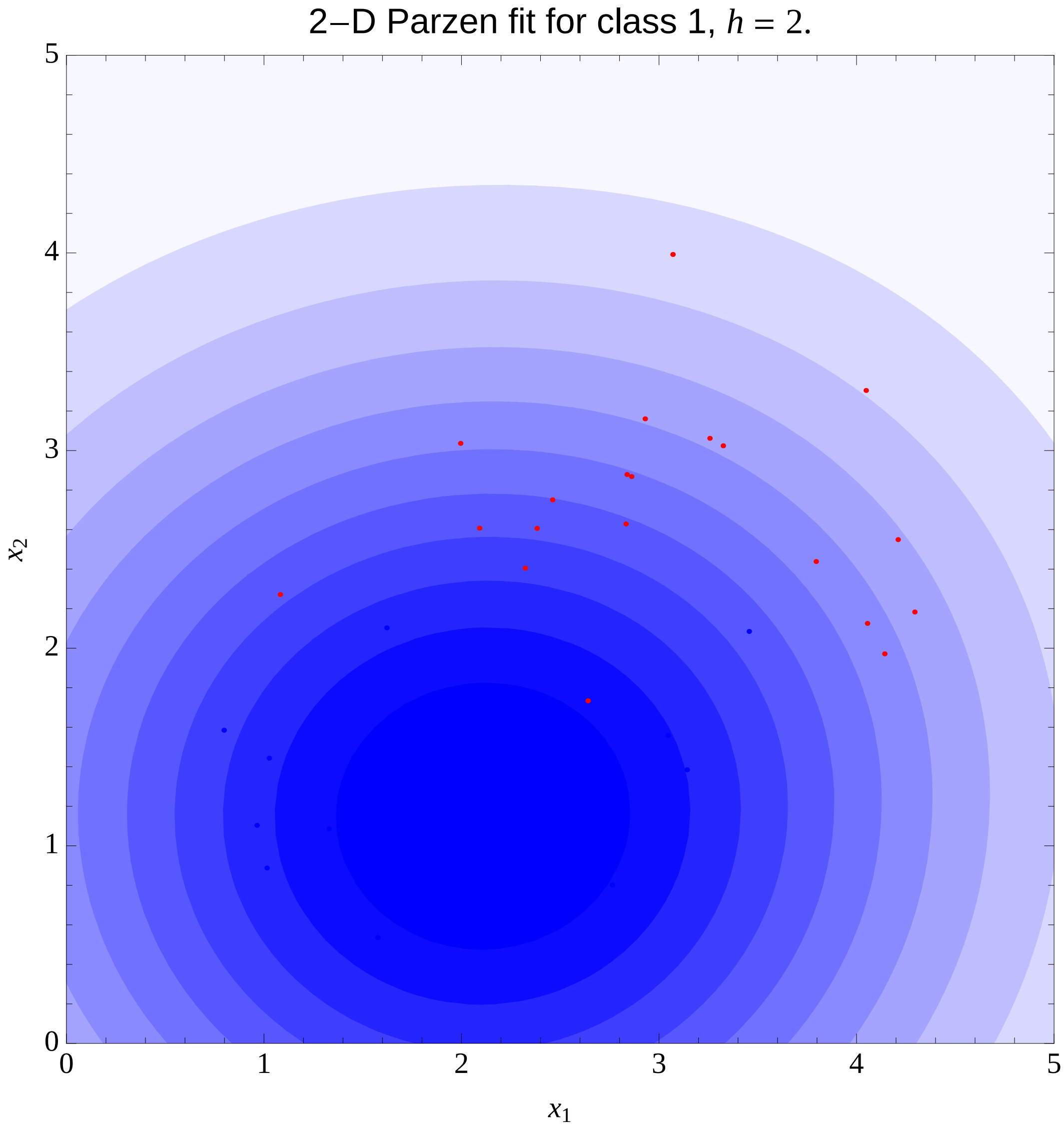


Discriminant function with Parzen fits, h = 0.25
0
1
2
3
4
5
x2
x1

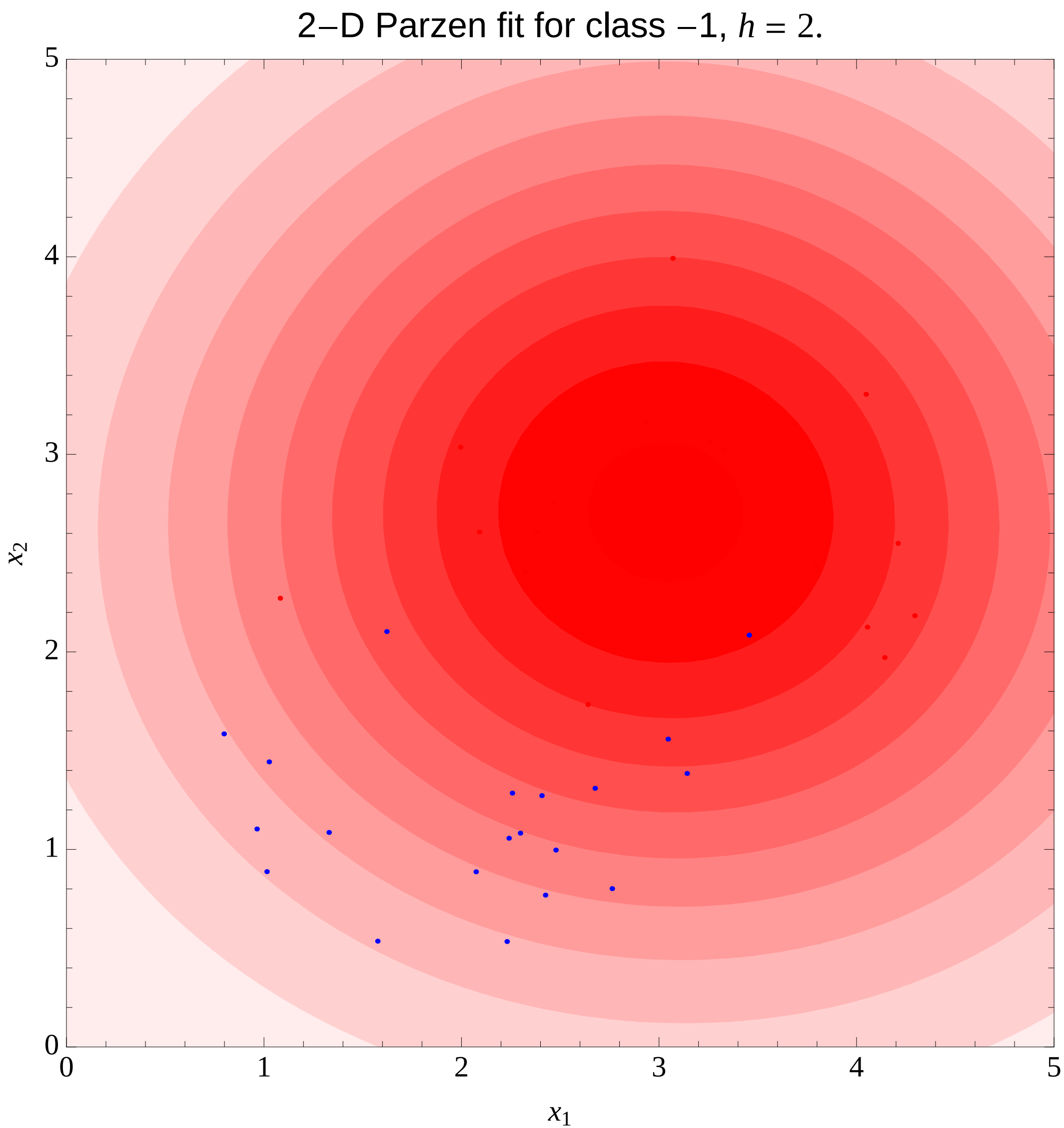

2−D Parzen fit for class 1, h = 0.01
0
1
2
3
4
5
0
1
2
3
4
5
x1
x2

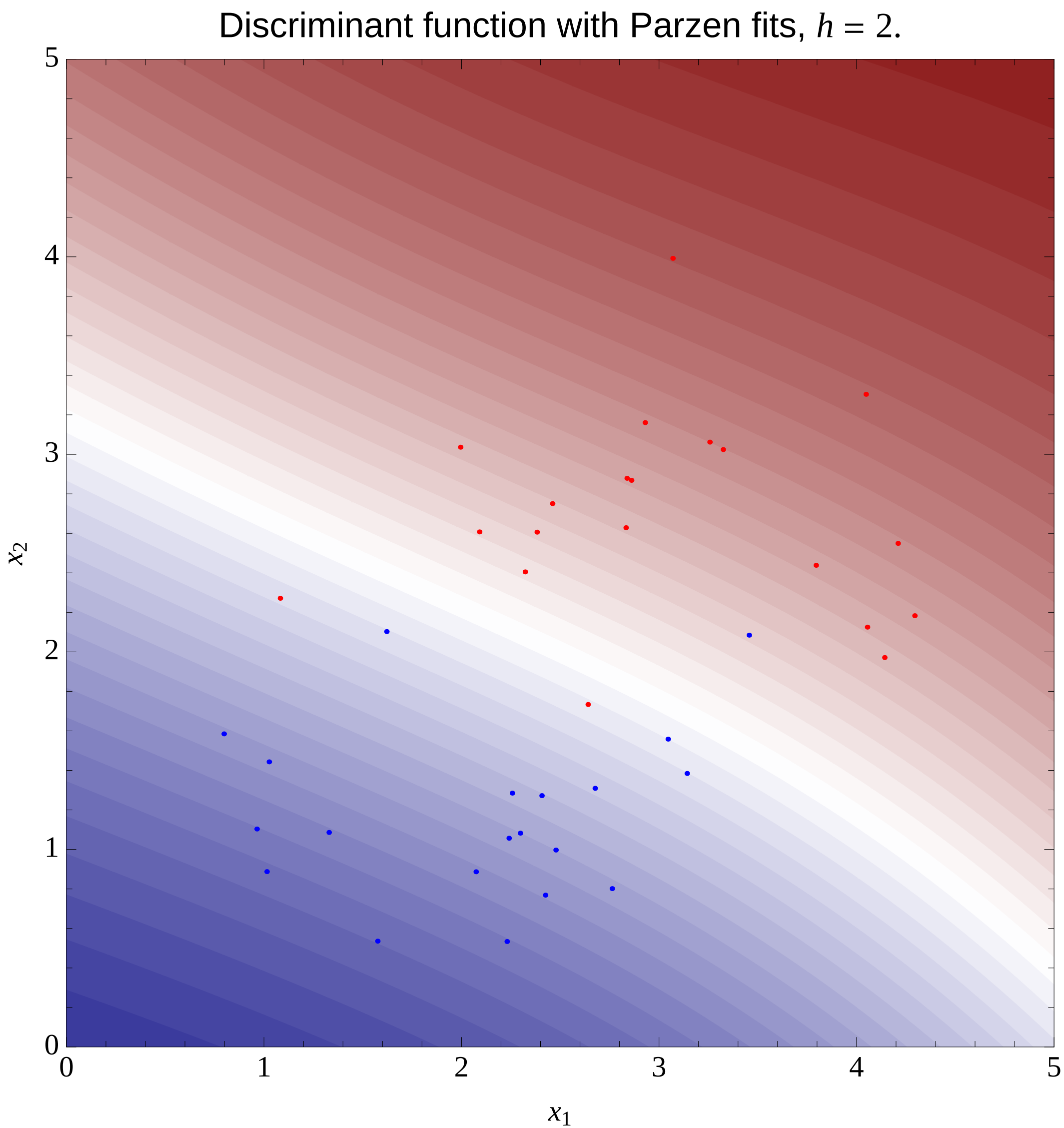

2−D Parzen fit for class −1, h = 0.01
0
1
2
3
4
5
0
1
2
3
4
5
x1
x2


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

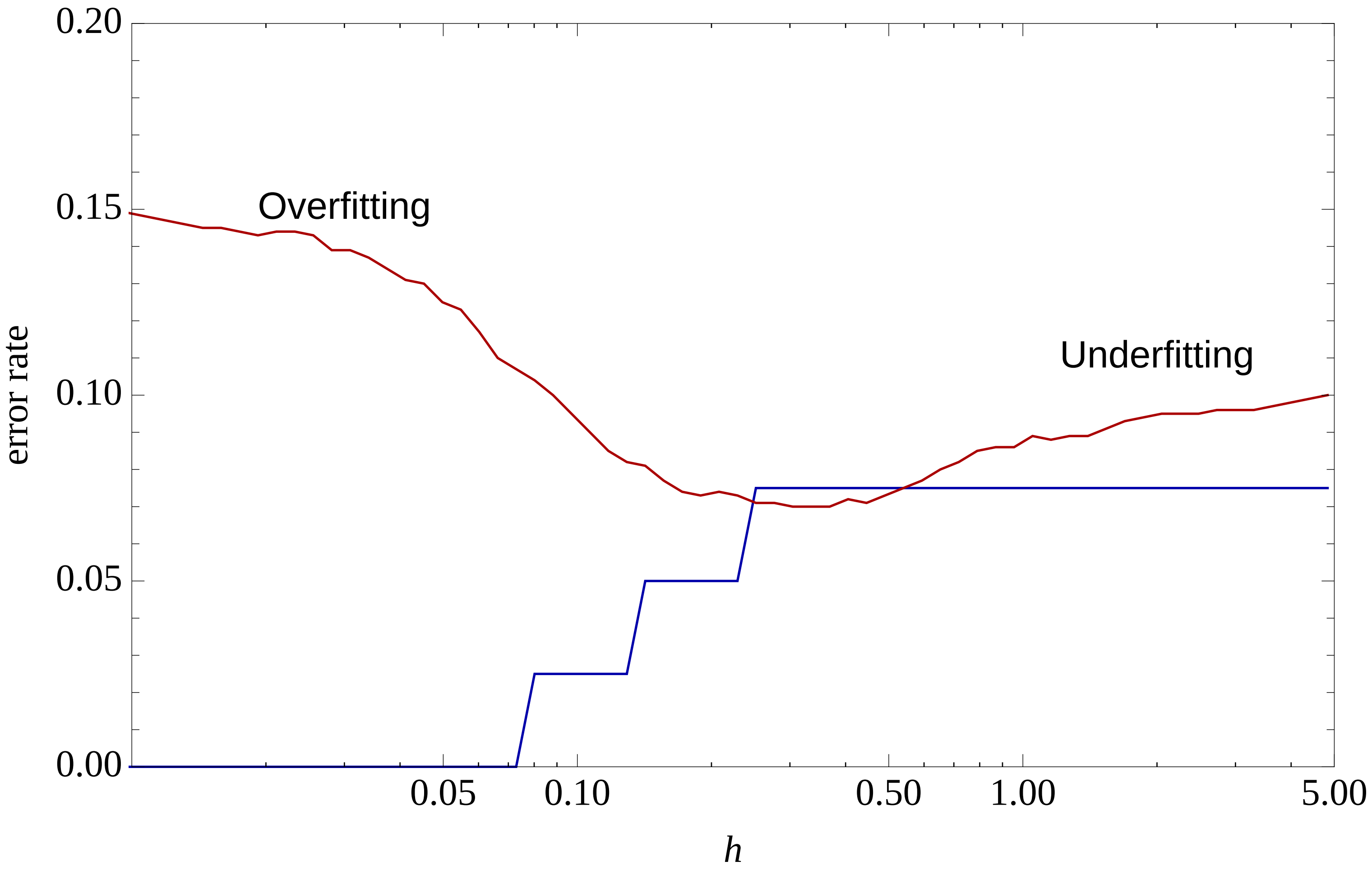


2–D Parzen fit for class 1, h = 2.
0
1
2
3
4
5
x1
x2


2−D Parzen fit for class −1, h = 2.
5
4
3
2
1
0
0
1
2
3
4
5
x2
x1


Discriminant function with Parzen fits, h = 2.
0
1
2
3
4
5
x1
x2

# Training and test error rates



# 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

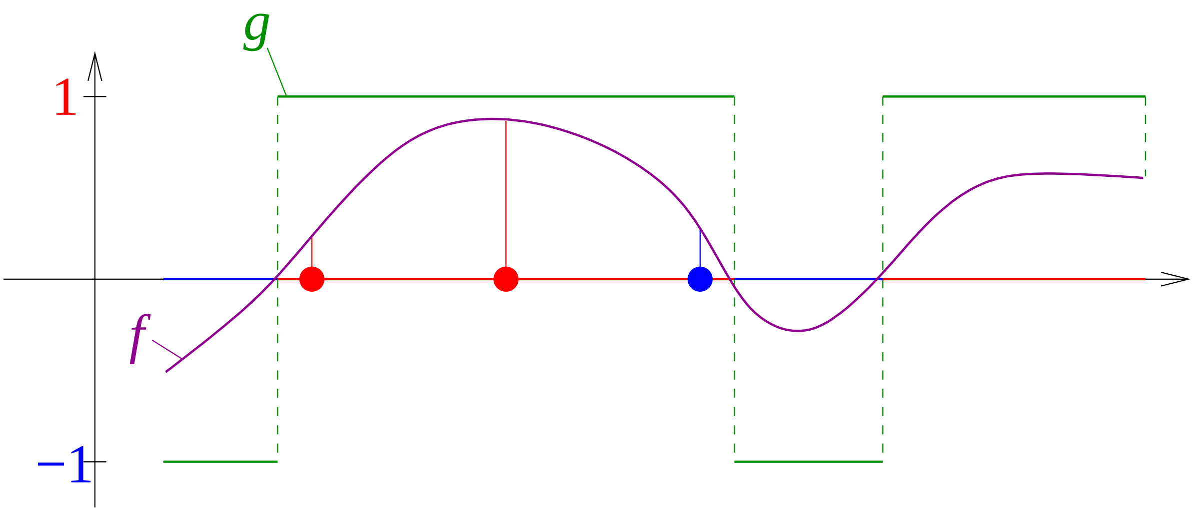
# The classification model



- 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: $f : \mathbb{R}^d \rightarrow [-1, 1]$

- $\longrightarrow$ 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 : $D_n = \big\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big\}$
  - function class : $\mathcal{F}$
  - learning algorithm : ALGO : $\big(\mathbb{R}^d \times \{-1,1\}\big)^n \to \mathcal{F}$

$$\text{ALGO}(D_n) \mapsto f$$

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

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

# The classification model



- goal: small generalization error

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

- Learning principle: minimize the training error

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

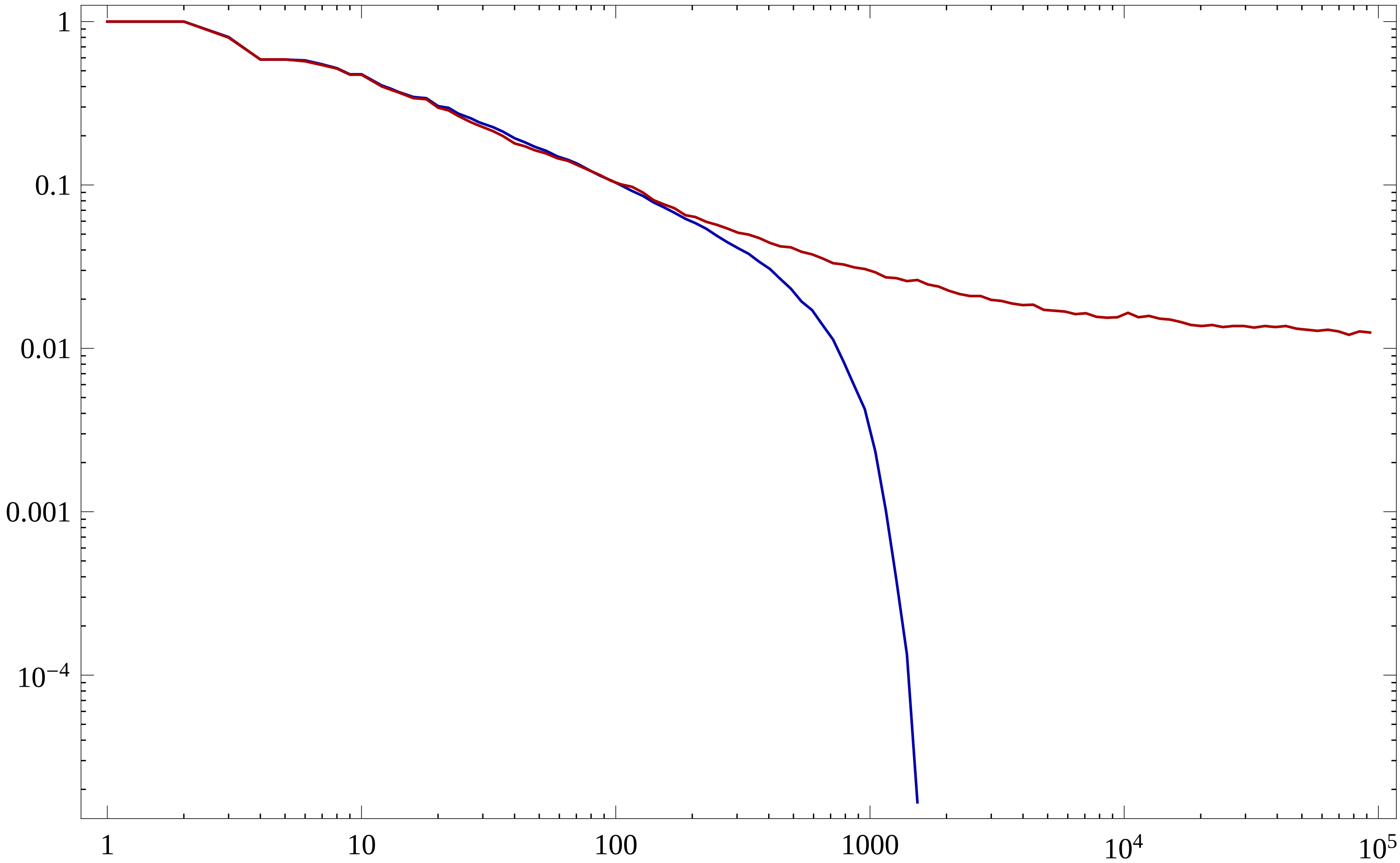
- Generalization error bounds

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

# The classification model

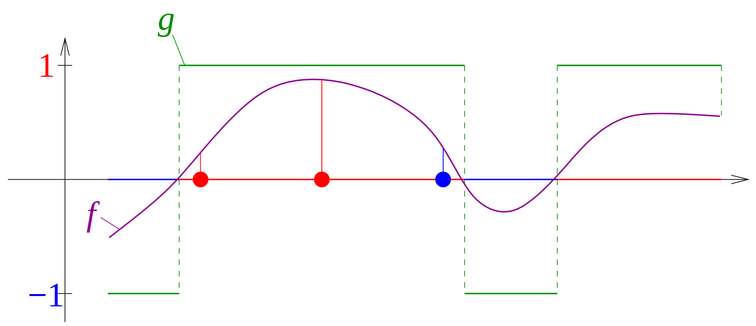


AdaBoost on MNIST: training and test error rates



# The classification model





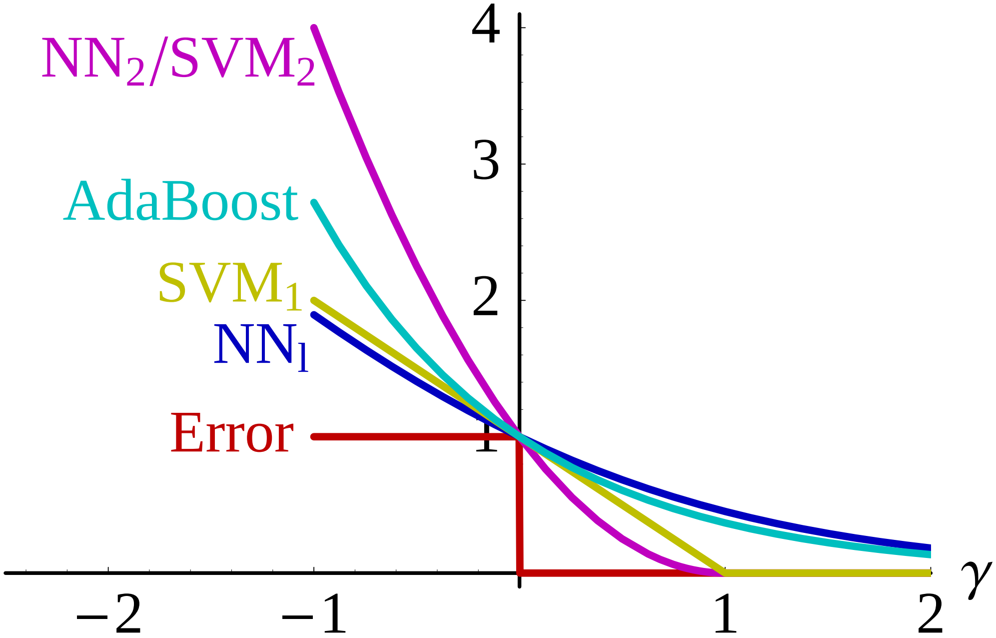

- Margin: $\gamma = y \cdot f(\mathbf{x})$
  - classification error $\equiv$ negative margin
  - the magnitude of a positive margin quantifies the confidence
  - learning principle: minimize a smooth loss function over the margin

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

# The classification model



- Margin loss functions



# History

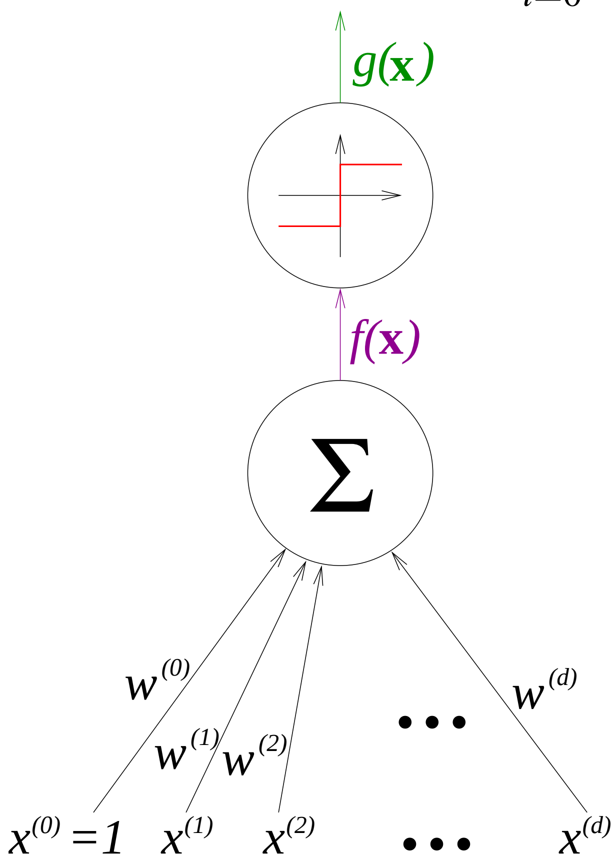


- Algorithms
  - 1958: Perceptron [Rosenblatt, ’58] – [Minsky–Papert ’69]
  - 1986: Multilayer perceptrons (neural networks) and the back-propagation algorithm [Rumelhart–Hinton–Williams, ’86]
  - 1995: Support vector machines [Boser–Guyon–Vapnik, ’92], [Cortes–Vapnik, ’95]
  - 1997: boosting, AdaBoost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

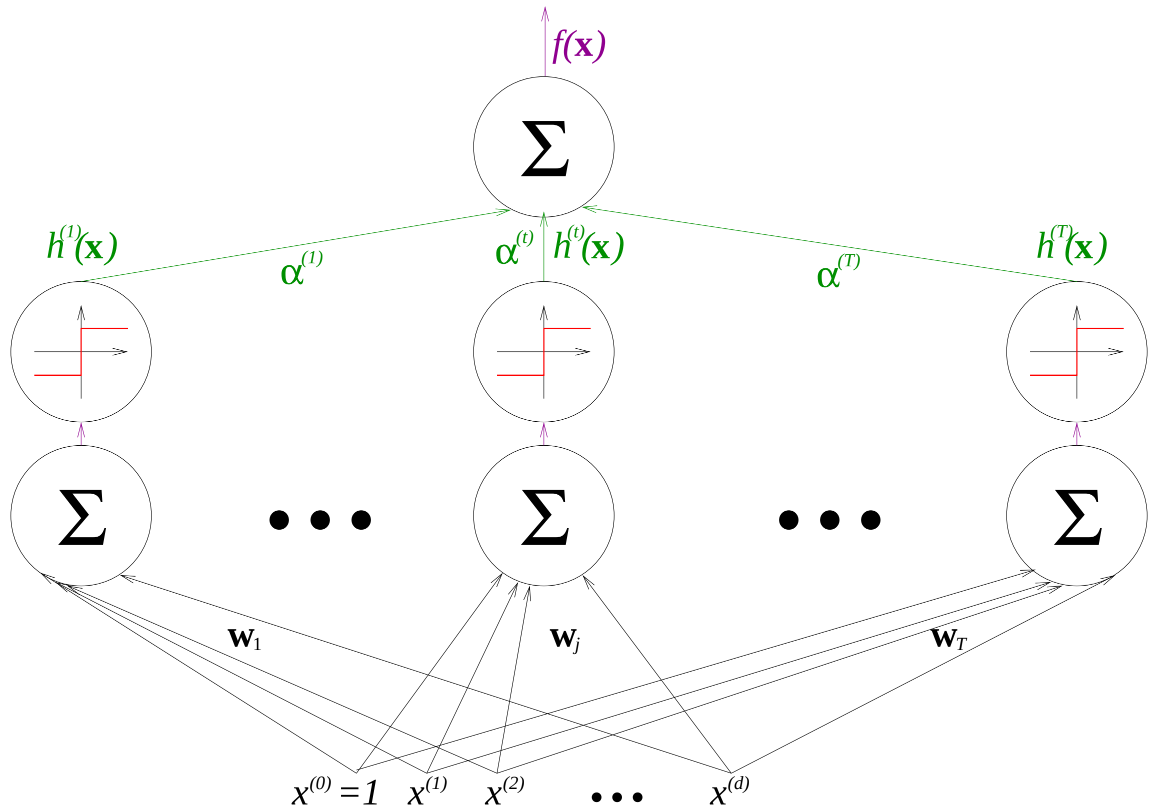
$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} h^{(j)}(\mathbf{x})$$

- $h^{(j)} : \mathbb{R}^d \to [-1, 1]$
  - 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 $\sigma$: 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
  - 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:

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

  - $I_{\text{SV}} \subset \{1, \ldots, n\}$ 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
  - $K(\mathbf{x}, \mathbf{x}') = \big(1 + \langle \mathbf{x}, \mathbf{x}' \rangle\big)^d \longrightarrow f(\mathbf{x})$ is a polynom of degree $d$
  - $K(\mathbf{x}, \mathbf{x}') = \exp\big(-1/h \|\mathbf{x} - \mathbf{x}'\|^2\big) \longrightarrow f(\mathbf{x})$ 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

# Machines à vecteurs de support



- Idée 1: séparation linéaire avec une marge maximale
  - marge fonctionnelle:

$$\gamma_i = f(\mathbf{x}_i) y_i = (\mathbf{w}^t \mathbf{x}_i + w_0) y_i$$

  - marge géométrique:

$$\gamma_i^{(g)} = \frac{1}{\|\mathbf{w}\|}(\mathbf{w}^t \mathbf{x}_i + w_0) y_i = \frac{1}{\|\mathbf{w}\|}\gamma_i$$

- Optimisations équivalentes
  - maximiser la marge géométrique
  - maximiser la marge fonctionnelle sous la contrainte $\|\mathbf{w}\| = 1$
  - minimiser $\|\mathbf{w}\|$ sous la contrainte $\gamma_i \geq 1$

# Machines à vecteurs de support

- Géométrie – deux classes

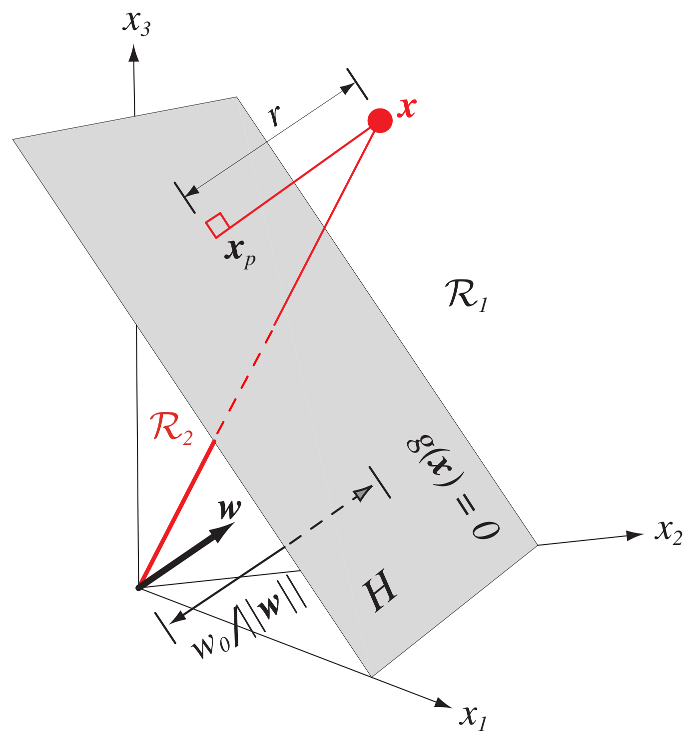
  - $r =$ distance algébrique de $\mathbf{x}$ et $H$:

$$
\begin{aligned}
\mathbf{x} &= \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \\
g(\mathbf{x}) &= \mathbf{w}^t\mathbf{x} + w_0 = r\|\mathbf{w}\| \\
r &= \frac{g(\mathbf{x})}{\|\mathbf{w}\|}
\end{aligned}
$$

# Machines à vecteurs de support



- Géométrie – deux classes



# Machines à vecteurs de support

- Problème d'optimisation:
  - soit $D_n = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big)$ linéairement séparable
  - minimiser $\|\mathbf{w}\|^2 = \mathbf{w}^t\mathbf{w}$
  - sous les contraintes $\gamma_i = (\mathbf{w}^t\mathbf{x}_i + w_0)y_i \geq 1, \;\; i = 1, \ldots, n$
- Résultat:
  - l'hyperplan ($\mathbf{w}^t\mathbf{x} + w_0 = 0$) avec une marge géométrique $\dfrac{1}{\|\mathbf{w}\|}$ maximale

- Problème primal – théorème de Kuhn-Tucker
  - minimiser par rapport à $\mathbf{w}$ et $w_0$ et maximiser par rapport à $\boldsymbol{\alpha}$:

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) &= \frac{1}{2}\mathbf{w}^t\mathbf{w} - \sum_{i=1}^{n} \alpha_i(\gamma_i - 1) \\ &= \frac{1}{2}\mathbf{w}^t\mathbf{w} - \sum_{i=1}^{n} \alpha_i\big((\mathbf{w}^t\mathbf{x}_i + w_0)y_i - 1\big) \end{aligned}$$

  - sous les contraintes $\alpha_i \geq 0, \quad i = 1, \ldots, n$

# Machines à vecteurs de support



- Optimisation:
  - les gradients:

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i \mathbf{x}_i y_i = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

  - resubstitution: maximiser par rapport à $\boldsymbol{\alpha}$ (problème dual):

$$\begin{aligned} W(\boldsymbol{\alpha}) &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j \end{aligned}$$

  - sous les contraintes $\alpha_i \geq 0, \; i = 1, \ldots, n$ et $\sum_{i=1}^{n} \alpha_i y_i = 0$

# Machines à vecteurs de support



- La solution
  - $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$
  - $w_0^* = -\frac{1}{2}\left(\max_{y_i=-1} \mathbf{w}^* \mathbf{x}_i + \min_{y_i=1} \mathbf{w}^* \mathbf{x}_i\right)$

# Machines à vecteurs de support



- La structure de la solution
  - $\mathbf{w}^*$ est une combinaison linéaire des points d'entraînement
  - théorème de Kuhn-Tucker:

$$\alpha_i^* \big( (\mathbf{w}^{*t} \mathbf{x}_i + w_0^*) y_i - 1 \big) = 0, \;\; i = 1, \ldots, n$$

  - si $\gamma_i > 1$ alors $\alpha_i^* = 0$
  - si $\alpha_i^* > 0$ alors $\gamma_i = 1$: vecteurs de support
  - $\mathbf{w}^* = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i$
  - $f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x} + w_0^* = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x} + w_0^*$

# Machines à vecteurs de support

- La structure de la solution
  - pour $j \in sv$

$$\gamma_j = y_j f^*(\mathbf{x}_j) = y_j \left( \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x}_j + w_0^* \right) = 1$$

  - alors

$$\begin{aligned} \mathbf{w}^{*t}\mathbf{w}^* &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i^* \alpha_j^* y_i y_j \mathbf{x}_i^t \mathbf{x}_j = \sum_{j \in sv} \alpha_j^* y_j \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x}_j = \sum_{j \in sv} \alpha_j^* (1 - y_j w_0^*) \\ &= \sum_{j \in sv} \alpha_j^* \end{aligned}$$

  - la marge maximale:

$$\gamma^* = \frac{1}{\|\mathbf{w}\|} = \left( \sum_{i \in sv} \alpha_i^* \right)^{-1/2}$$

# Machines à vecteurs de support



- Idée 2: le noyau

  - l'optimisation: $W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j$

  - la fonction optimale: $f^*(\mathbf{x}) = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x} + w_0^*$

- Remplacer $\mathbf{x}^t \mathbf{x}'$ par une fonction de noyaux $K(\mathbf{x}, \mathbf{x}')$

  - l'optimisation: $W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$

  - la fonction optimale: $f^*(\mathbf{x}) = \sum_{i \in sv} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^*$

  - matrice de Gram: $\mathbf{G}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$

  - équivalent à une transformation non-linéaire dans une espace de traits de dimension très élévée

# Machines à vecteurs de support



- Exemples de noyaux
  - linéaire: $K^{\ell}(\mathbf{x},\mathbf{x}') = \mathbf{x}^t\mathbf{x}'$
  - polynômial: $K_d^p(\mathbf{x},\mathbf{x}') = (\mathbf{x}^t\mathbf{x}' + 1)^d$
  - base radiale: $K^r(\mathbf{x},\mathbf{x}') = K(\|\mathbf{x}-\mathbf{x}'\|)$
    - gaussien: $K_\gamma^g(\mathbf{x},\mathbf{x}') = e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2}$
  - réseaux de neurones: $K^\sigma(\mathbf{x},\mathbf{x}') = \sigma(s\mathbf{x}^t\mathbf{x}' + c)$
    - sigmoïde: $K_{s,c}^t(\mathbf{x},\mathbf{x}') = \tanh(s\mathbf{x}^t\mathbf{x}' + c)$

# Machines à vecteurs de support



- Discrimination linéaire généralisée

  - linéaire:

$$f^*(\mathbf{x}) = \mathbf{w}^{*t}\mathbf{x} + w_0^* = \sum_{j=1}^{d} w_j^* x^{(j)} + w_0^* = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i^t \mathbf{x} + w_0^*$$

  - généralisée:

$$f^*(\mathbf{x}) = \sum_{j=1}^{D} w_j^* \phi^{(j)}(\mathbf{x}) + w_0^* = \sum_{i \in sv} \alpha_i^* y_i \sum_{j=1}^{D} \phi^{(j)}(\mathbf{x}_i)\phi^{(j)}(\mathbf{x}) + w_0^*$$

  - produit scalaire dans l'espace de traits:

$$\boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}') = \sum_{j=1}^{D} \phi^{(j)}(\mathbf{x})\phi^{(j)}(\mathbf{x}')$$

# Machines à vecteurs de support



- Discrimination linéaire généralisée

  - exemple:

$$\begin{aligned}\phi^{(j)}(\mathbf{x}) &= \sqrt{2}x^{(j)}, \quad j = 1,\ldots,d,\\ \phi^{(id+j)}(\mathbf{x}) &= x^{(i)}x^{(j)}, \quad i,j = 1,\ldots,d\\ \phi^{(d^2)}(\mathbf{x}) &= 1\end{aligned}$$

  - produit scalaire dans l'espace des traits:

$$\begin{aligned}\boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}') &= \sum_{j=1}^{d^2}\phi^{(j)}(\mathbf{x})\phi^{(j)}(\mathbf{x}')\\ &= \sum_{j=1}^{d}\sqrt{2}x^{(j)}\sqrt{2}x'^{(j)} + \sum_{i=1}^{d}\sum_{j=1}^{d}x^{(i)}x^{(j)}x'^{(i)}x'^{(j)} + 1\\ &= 2\mathbf{x}^t\mathbf{x}' + (\mathbf{x}^t\mathbf{x}')^2 + 1\\ &= (\mathbf{x}^t\mathbf{x}' + 1)^2 = K_2^p(\mathbf{x},\mathbf{x}')\end{aligned}$$

# Machines à vecteurs de support

- Caractérisation des noyaux
  - $K(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}') = \sum_{j=1}^{D} \phi^{(j)}(\mathbf{x})\phi^{(j)}(\mathbf{x}')$
  - commutativité: $K(\mathbf{x},\mathbf{x}') = K(\mathbf{x}',\mathbf{x})$
  - inégalité de Cauchy-Schwartz:

$$\begin{aligned} K(\mathbf{x},\mathbf{x}')^2 &= \left(\boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}')\right)^2 \\ &\leq \|\boldsymbol{\phi}(\mathbf{x})\|^2\|\boldsymbol{\phi}(\mathbf{x}')\|^2 \\ &= \boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}^t(\mathbf{x}')\boldsymbol{\phi}(\mathbf{x}') \\ &= K(\mathbf{x},\mathbf{x})K(\mathbf{x}',\mathbf{x}') \end{aligned}$$

# Machines à vecteurs de support



- Théorème de Mercer:
  - $\boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}') = K(\mathbf{x},\mathbf{x}')$
- Conditions
  - diagonaliser la matrice de Gram: $\mathbf{G}_{i,j} = K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{V}\Lambda\mathbf{V}$
  - valeurs propres: $\lambda_t$, vecteurs propres: $\mathbf{v}_t$
  - condition suffisante: $\mathbf{G}$ est positif semi-definit $(\forall t : \lambda_t > 0)$ pour tout $\mathcal{X}_n = (\mathbf{x}_1,\ldots,\mathbf{x}_n)$
  - $D$ peut être $\infty$!!!

# Machines à vecteurs de support



- Idée 3: les variables d'écart (slack variables)
  - cas non-séparable
- Permettre des erreurs:
  - minimiser $\|\mathbf{w}\|^2 = \mathbf{w}^t\mathbf{w}$
  - sous les contraintes $\gamma_i = (\mathbf{w}^t\mathbf{x}_i + w_0)y_i \geq 1 - \xi_i, \;\; i = 1, \ldots, n$
  - où $\xi_i \geq 0, \;\; i = 1, \ldots, n$
  - minimiser l'erreur $\equiv$ minimiser le nombre de points avec $\xi_i > 0$: NP-difficile

# Machines à vecteurs de support



- Problème soluble 1: minimiser $\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i^2$
  - on peut supprimer la contrainte de positivité des $\xi_i$
  - $C$ est un hyper-paramètre réglé par la validation croisée (par exemple)

# Machines à vecteurs de support

- Problème primal
  - minimiser par rapport à $\mathbf{w}$, $\boldsymbol{\xi}$ et $w_0$ et maximiser par rapport à $\boldsymbol{\alpha}$:

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left[(\mathbf{w}^t\mathbf{x}_i + w_0)y_i - 1 + \xi_i\right]$$

  - sous les contraintes $\alpha_i \geq 0, \; i = 1, \ldots, n$
  - les gradients:

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i\mathbf{x}_i y_i = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \boldsymbol{\xi}} = C\boldsymbol{\xi} - \boldsymbol{\alpha} = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = \sum_{i=1}^{n}\alpha_i y_i = 0$$

# Machines à vecteurs de support



- Problème dual

  - resubstitution: maximiser par rapport à $\boldsymbol{\alpha}$:

$$
\begin{aligned}
W(\boldsymbol{\alpha}) &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \frac{1}{2C}\boldsymbol{\alpha}^t\boldsymbol{\alpha} - \frac{1}{C}\boldsymbol{\alpha}^t\boldsymbol{\alpha} \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \frac{1}{2C}\boldsymbol{\alpha}^t\boldsymbol{\alpha} \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \left(K(\mathbf{x}_i,\mathbf{x}_j) + \frac{1}{C}\delta_{i,j}\right)
\end{aligned}
$$

  - sous les contraintes $\alpha_i \geq 0,\ \ i = 1,\ldots,n$ et $\sum_{i=1}^{n}\alpha_i y_i = 0$

# Machines à vecteurs de support

- La solution
  - $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i$
  - $f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^* = \sum_{i \in sv} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^*$
  - $w_0^*$ est choisi tel que $\gamma_i = f(\mathbf{x}_i) y_i \geq 1 - \xi_i = 1 - \frac{\alpha_i^*}{C}$
  - la marge obtenue: $\gamma = \left( \sum_{i \in sv} \alpha_i^* - \frac{1}{C} \alpha^{*t} \alpha^* \right)^{-1/2}$

# Machines à vecteurs de support

- Problème soluble 2: minimiser $\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$

- Problème primal

  - minimiser par rapport à $\mathbf{w}$, $\boldsymbol{\xi}$ et $w_0$ et maximiser par rapport à $\boldsymbol{\alpha}$:

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[(\mathbf{w}^t\mathbf{x}_i + w_0)y_i - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i\xi_i$$

  - sous les contraintes $\alpha_i \geq 0, r_i \geq 0 \;\; i = 1, \ldots, n$

# Machines à vecteurs de support



- Problème dual

  - les gradients:

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i \mathbf{x}_i y_i = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \boldsymbol{\xi}} = C - \boldsymbol{\alpha}_i - r_i = 0$$

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

# Machines à vecteurs de support



- Problème dual
  - resubstitution: maximiser par rapport à $\boldsymbol{\alpha}$:

$$W(\boldsymbol{\alpha}) \;=\; \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$$

  - sous les contraintes $\alpha_i \geq 0, r_i \geq 0, C - \boldsymbol{\alpha}_i - r_i = 0 \;\; i = 1,\ldots,n$ et $\sum_{i=1}^{n}\alpha_i y_i = 0$
  - $\equiv$ sous les contraintes $0 \leq \alpha_i \leq C$ (contraintes de boîte)

# Machines à vecteurs de support

- La solution
  - $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i = \sum_{i \in sv} \alpha_i^* y_i \mathbf{x}_i$
  - $f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^* = \sum_{i \in sv} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + w_0^*$
  - $w_0^*$ est choisi tel que $\gamma_i = f(\mathbf{x}_i) y_i = 1$ pour tous $i : 0 < \alpha^* < C$
  - la marge obtenue: $\gamma = \left( \sum_{i,j \in sv} \alpha_i^* \alpha_j^* y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)^{-1/2}$

# Final thoughts



- If you can build a generative model for the class densities, do it
- All algorithms are optimized for classification, not hypothesis testing
  - doesn't mean they don't work, but they are probably not optimal
- None of the margin-based algorithms like label noise
- Non-parametric methods also exist for other objectives (e.g., regression, density estimation)

# Software



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

Thank you!