Experimental Methods for Tuning Parameter Selection in Machine Learning and Statistics

Benjamin Tyner
Department of Statistics
Purdue University

William S. Cleveland
Department of Statistics
Department of Computer Science
Purdue University

Tuning Parameters

Certain parameters in learning algorithms and statistical models

Typically, as parameter increases or decreases, the fit gets closer to data

Parameter might be “in” or “out” to indicate whether a term of a model is included or not

One example

- y = polynomial(x, coefficients, degree) + noise
- tuning parameter = polynomial degree

Tuning Parameter Selection: The Current Practice

A Machine Optimization

A “model selection criterion”

- cross-validation sum of squares
- Mallows C_p
- AIC and BIC
- minimize criterion

Return just the parameters that optimize the criterion

Widely practiced in statistics and machine learning

Different Approach to Tuning Parameter Selection

Use Experimental Methods:

(1) Experimental Design (2) Response Surface Analysis

Evaluate model selection criterion (response) over space of tuning parameters (factors)

Add a measure of model complexity and an estimate of the noise variance

Three responses

- model selection criterion
- model complexity
- the estimate of the noise variance

Treat these meta-data as we would data from a designed experiment

The design space

- evaluate three responses for set of points in the space of tuning parameters

Analyze the meta-data using response surface methods

Why Experimental Methods?

Much more effective tuning parameter selection for the data at hand

- all models are wrong
- balance model prediction bias and variance
- a complex matter

Build up a knowledge base across experiments about the effects and properties of the tuning parameters

Expand the scope of the model selection

- greater numbers of tuning parameters
- success where optimizer would get lost

Leads naturally to an effective separation of estimating the noise variance and selecting the tuning parameters

Design and Response Surfaces

Response Surface Methods

Extensive data visualization

Transformations of the factors to simplify the surfaces

Fitting equations to surfaces

Characterization of statistical variability of surfaces

Methods of Experimental Design

Guide choice of design space

Often full factorial

When each run costly, fractionated designs

The evolutionary operation methods of Box

Illustrate with Locally-Weighted Regression (Loess)

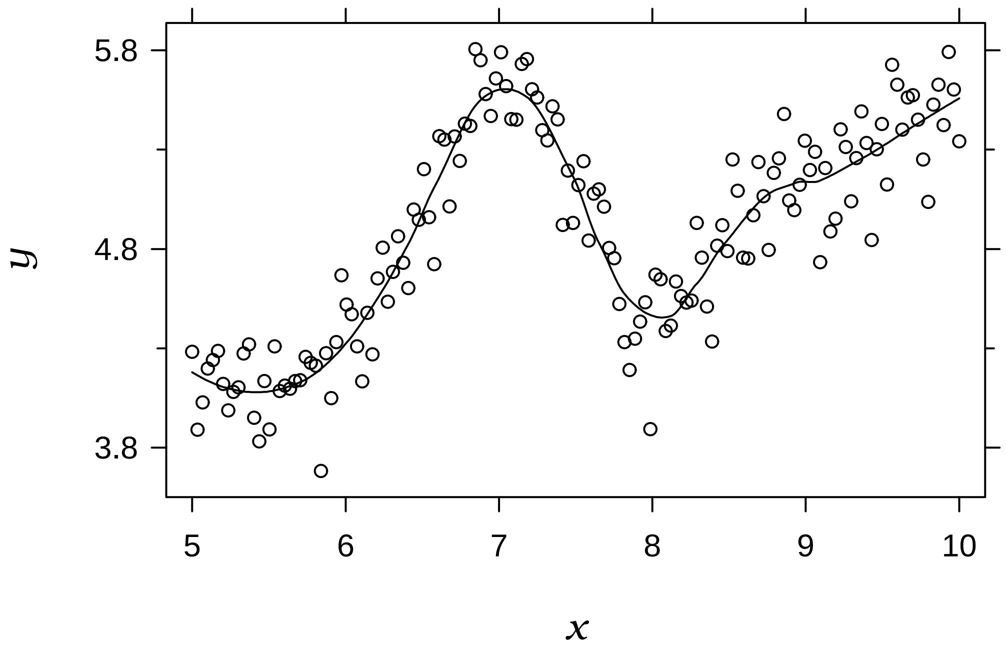
Fit response y_i to p explanatory variables $x_{i1}, \ldots, x_{ip}$, i = 1 to n

Loess smooths y_i as a function of the $x_{i1}, \ldots, x_{ip}$ by local fitting of polynomial family

Result is $\hat{g}(x_{i1}, \ldots, x_{ip})$, which describes mean of response given explanatory variables

Assumptions

- $y_i = g(x_i) + \epsilon_i$
- ϵ_i are i.i.d. with mean 0 and variance σ^2
- g is a smooth function of the explanatory variables



Assumption

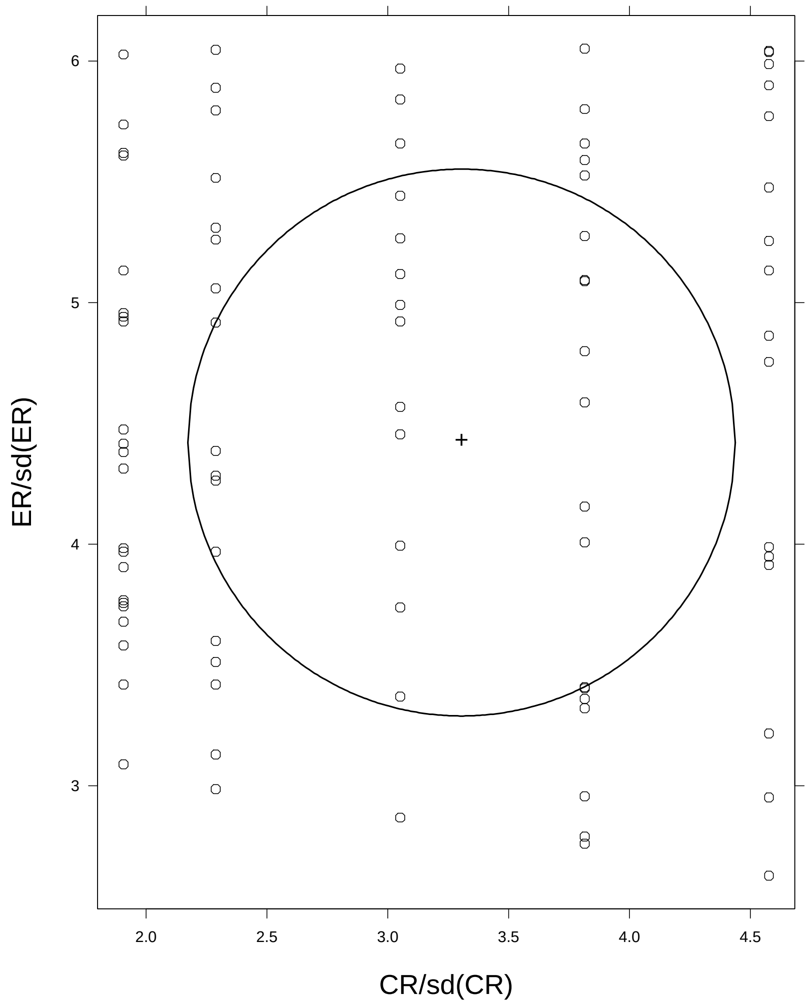
- for method of fitting to come, ϵ_i not too far from normal

Neighborhood and Weight Function

Example: two explanatory variables

Steps to get a value of loess fit at “+”

- 1. Divide each explanatory variable by a measure of scale
- 2. Choose polynomial degree
- 3. Choose neighborhood by number of nearest neighbors
- 4. Compute weight function at each point
- 5. Fit polynomial by weighted least squares
- 6. Evaluate fitted polynomial at “+”



Camel

We will use the camel function of Larry Brown

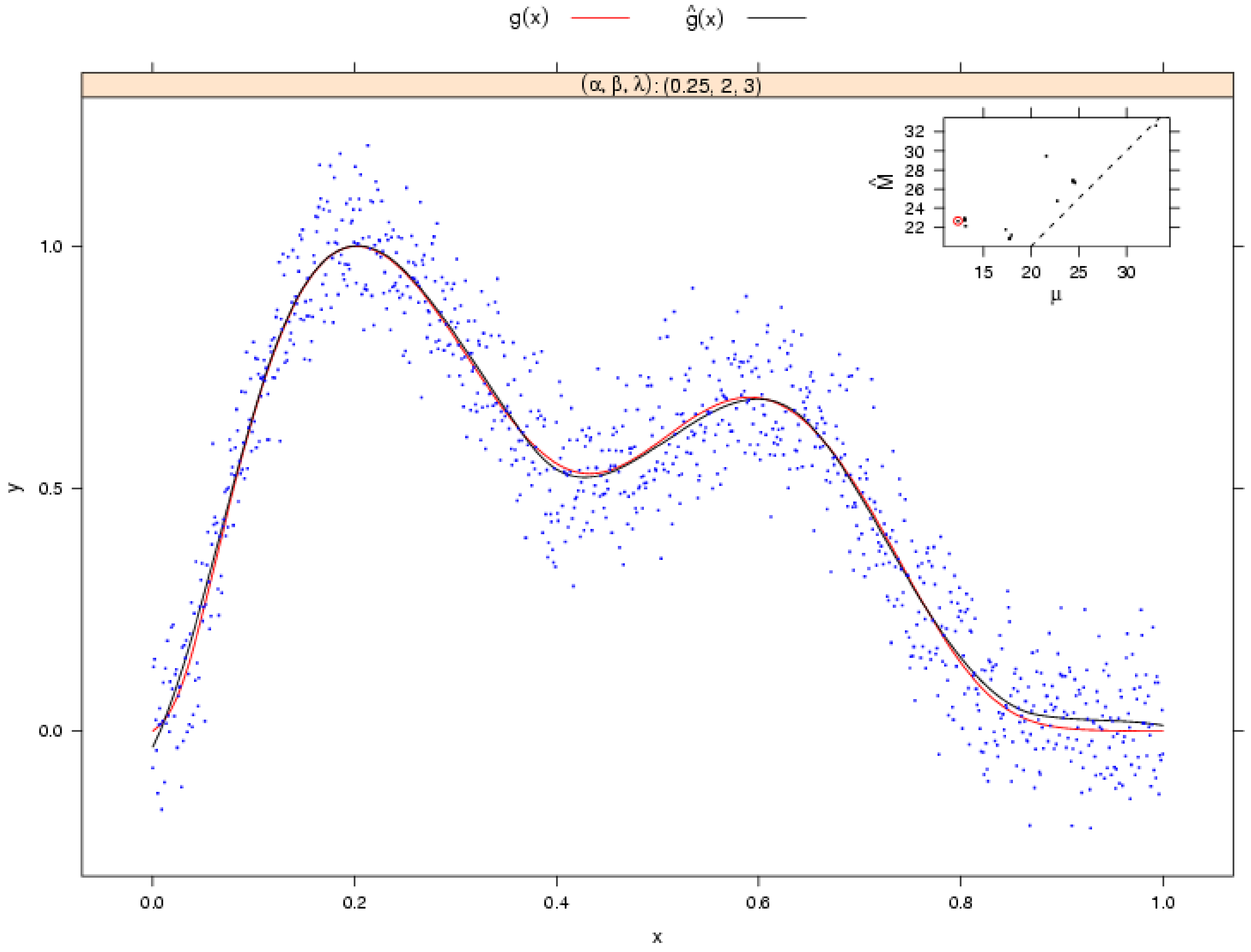
One independent variable

1028 observations

Noise is normal with mean 0 and standard deviation 0.1

In the next slide the loess fit has 256 points per neighborhood and local cubic fitting. The small insert plot will be explained latter.

Camel Loess Fit (Black) and Actual (Red)



Computational Method

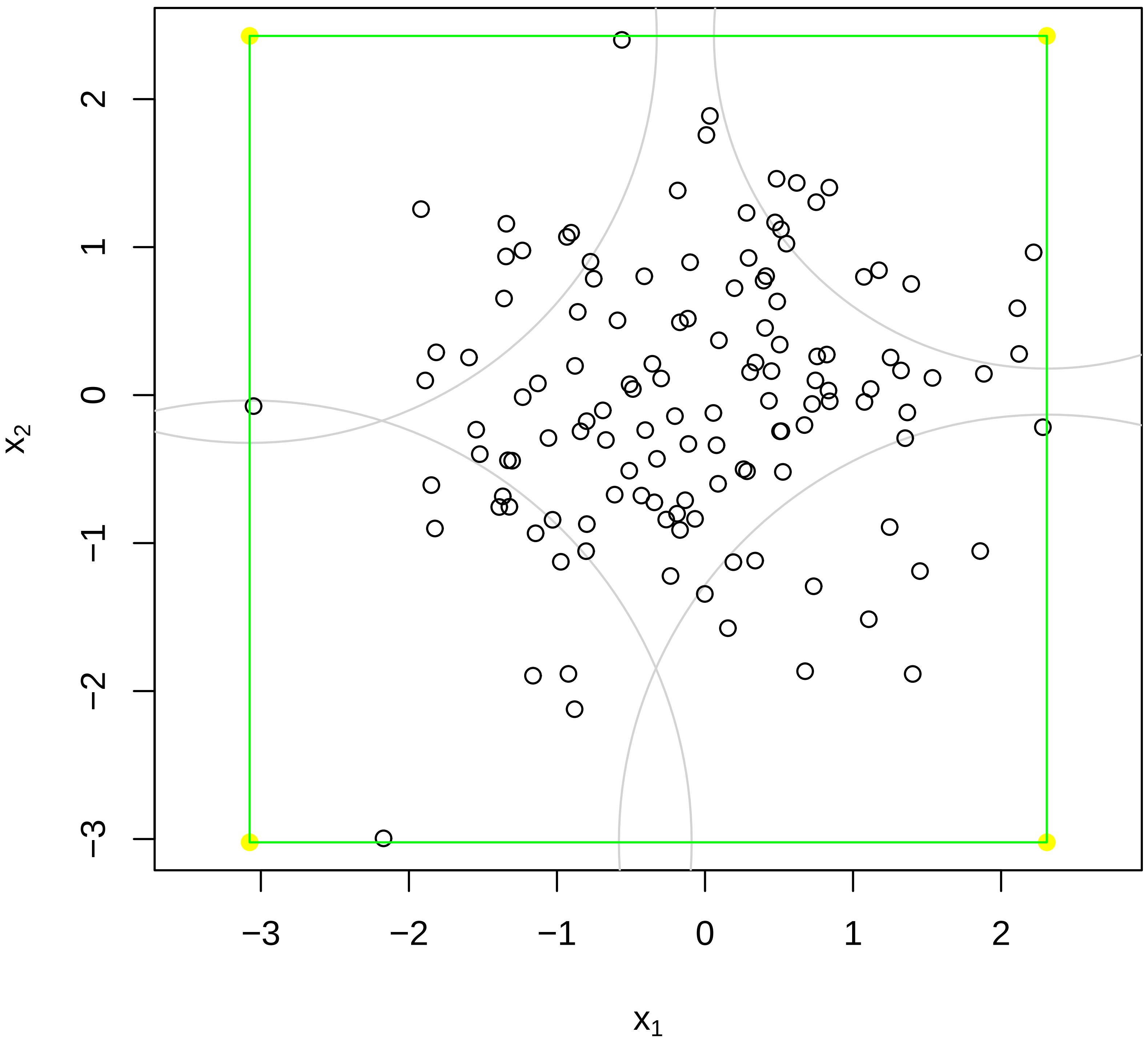
A loess computation at the at all $x_{i1}, \ldots, x_{ip}$ is order n^2, too costly for large data sets

Build a k-d tree in the space of the explanatory variables

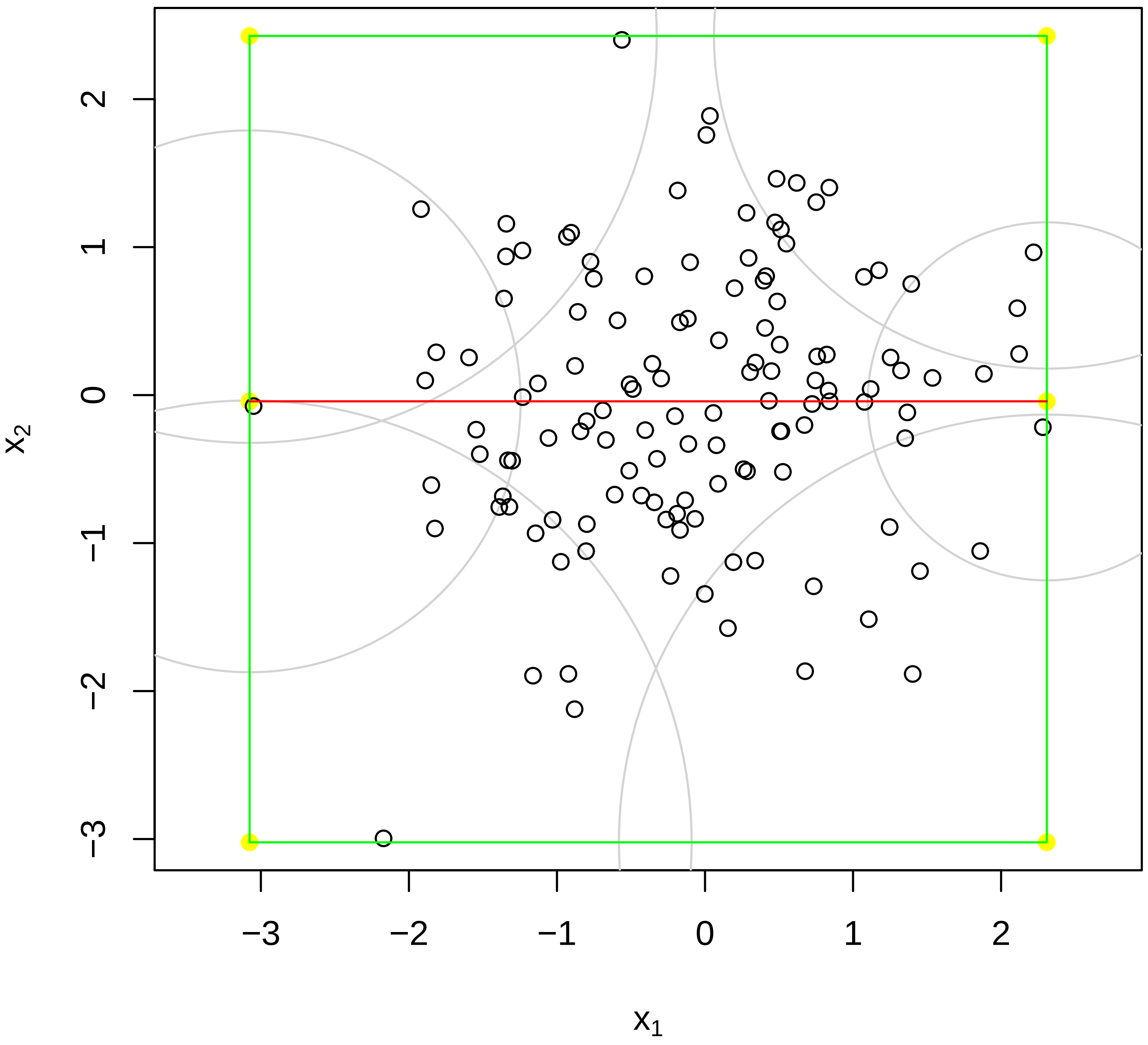
Do full loess computation at the vertices of the tree and interpolate elsewhere

Number of cells, or leaf nodes, of the tree is a power of 2

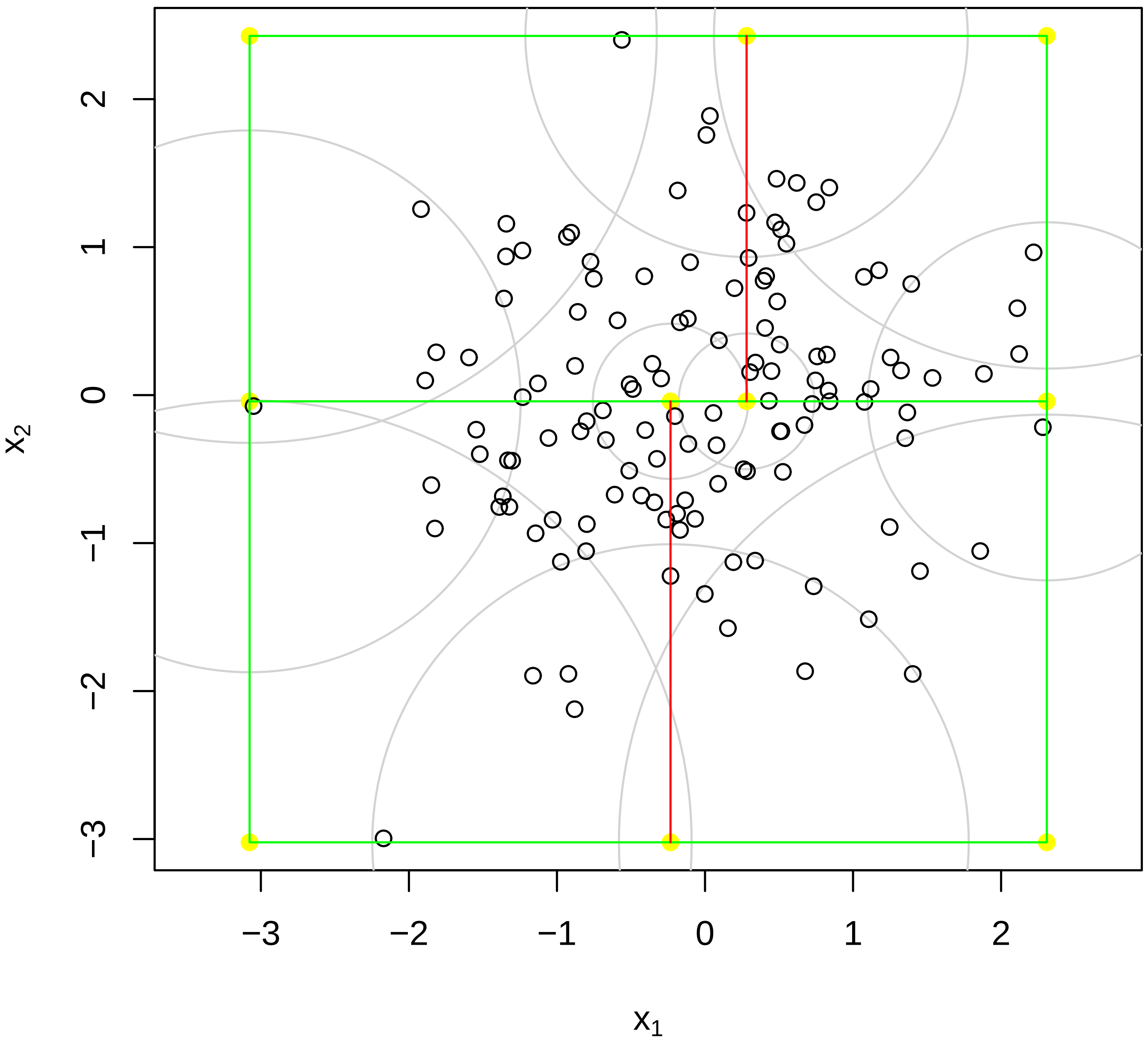
1 leaf cell   4 vertices

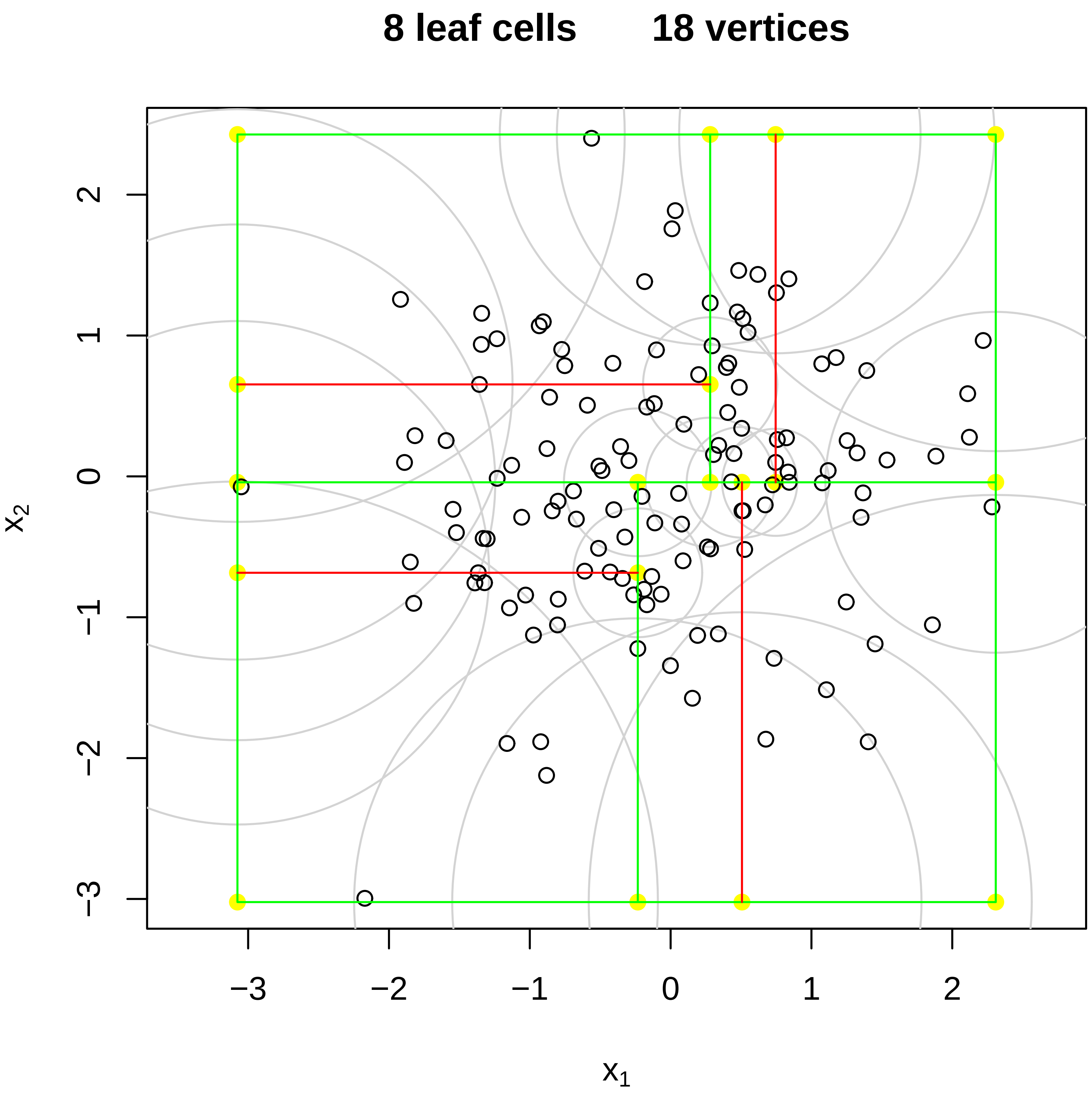


2 leaf cells 6 vertices

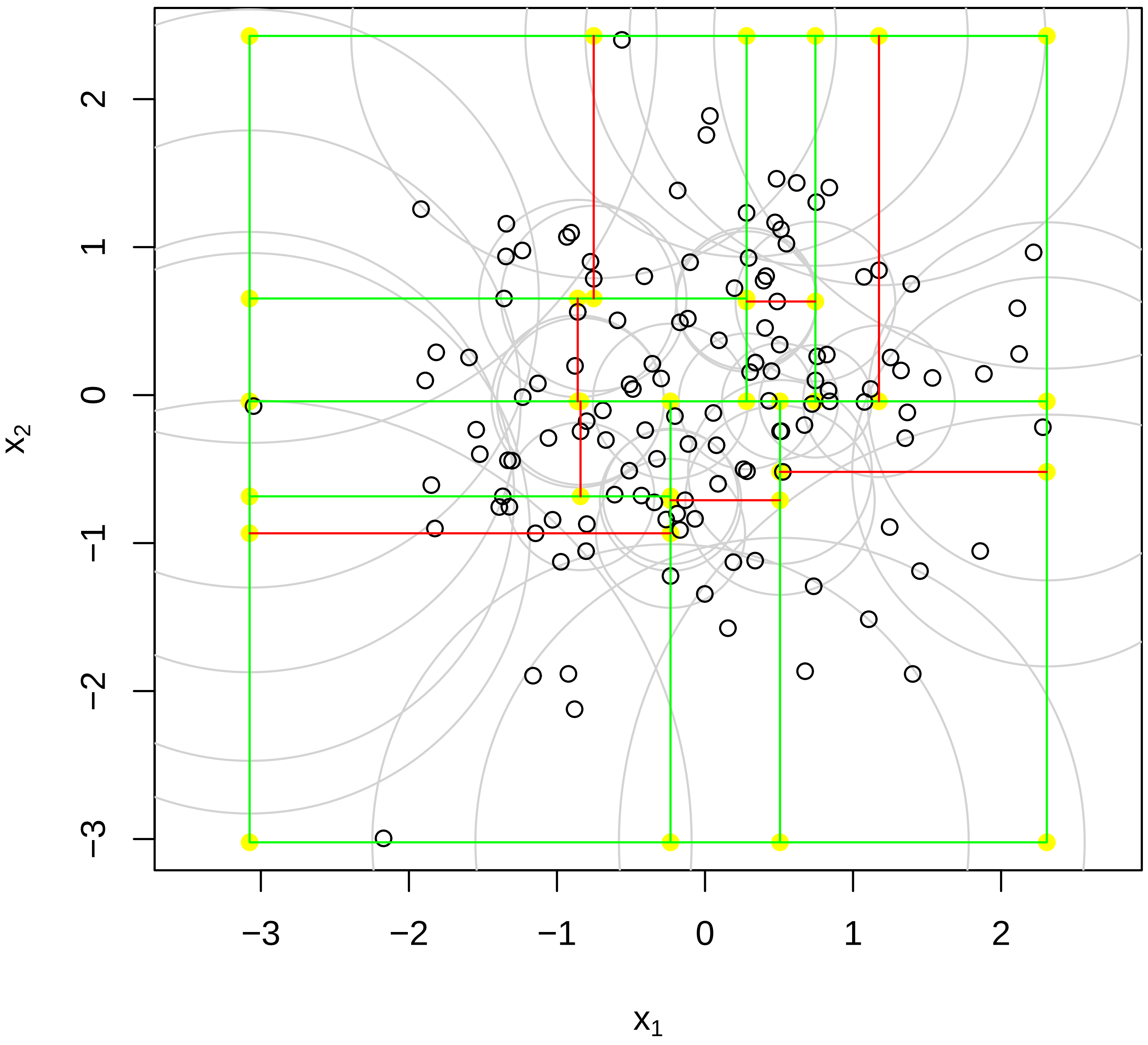


4 leaf cells 10 vertices




8 leaf cells
18 vertices
x_1
x_2
−3
−2
−1
0
1
2
−3
−2
−1
0
1
2

16 leaf cells   34 vertices



Loess Tuning Parameters (An Aggressive Agenda)

Polynomial degree

- λ
- 0, 1, 2, 3
- alternative is τ = number of monomial fitting variables

Bandwidth

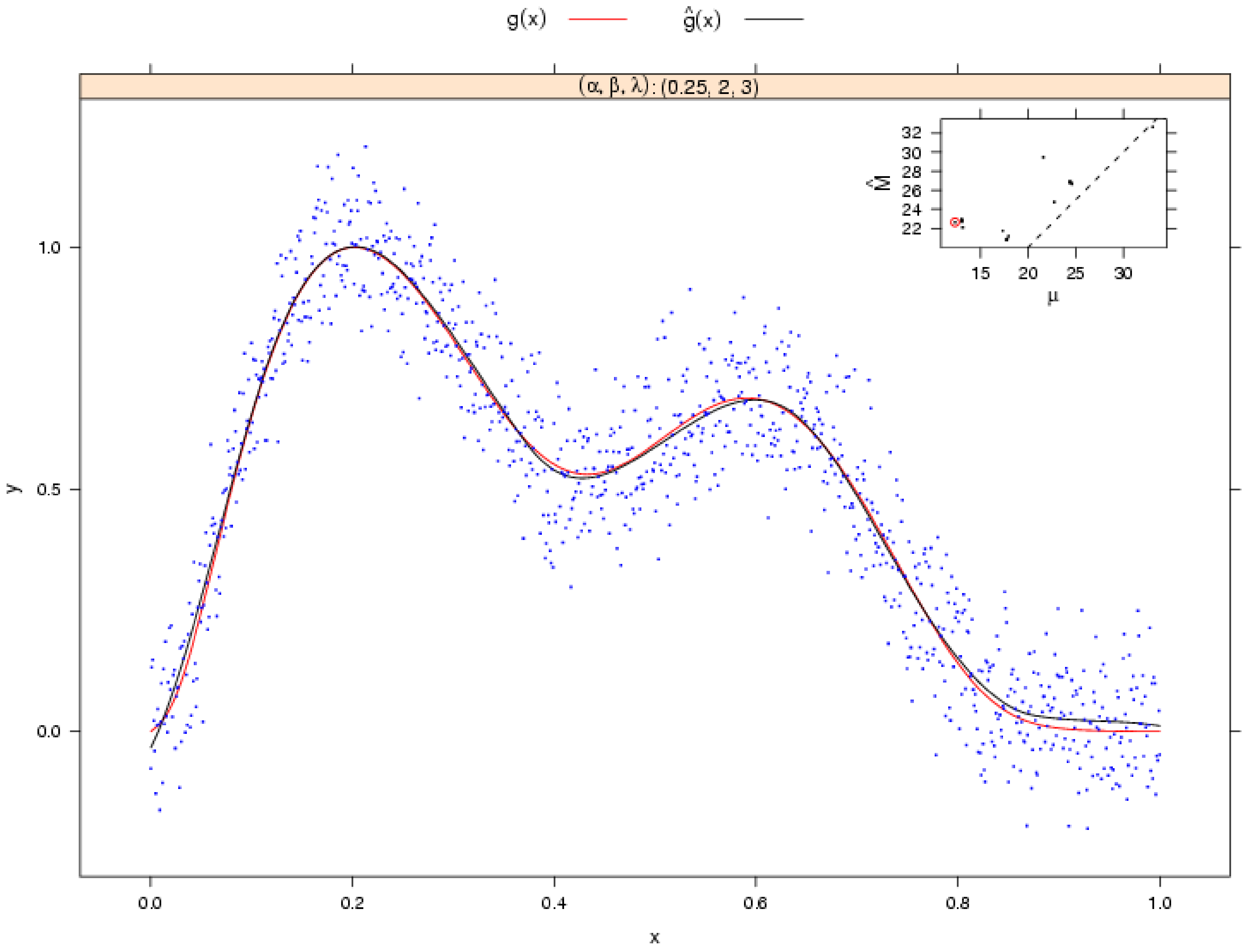
- α
- fraction of points in each neighborhood

Number of leaf cells of k-d tree

- n_ℓ = 2^v

Other tuning parameters in the future

Camel Fit: $\lambda = 3$, $\alpha = 0.25$, $n_\ell = 8$



Loess: Residuals, Fitted Values, and Hat Matrix

i = 1 to n observations

y_i is the response at $x_{i1}, \ldots, x_{ip}$

$\hat{y}_i$ is the loess fit at $x_{i1}, \ldots, x_{ip}$

$\hat{\epsilon}_i = y_i - \hat{y}_i$ is the residual

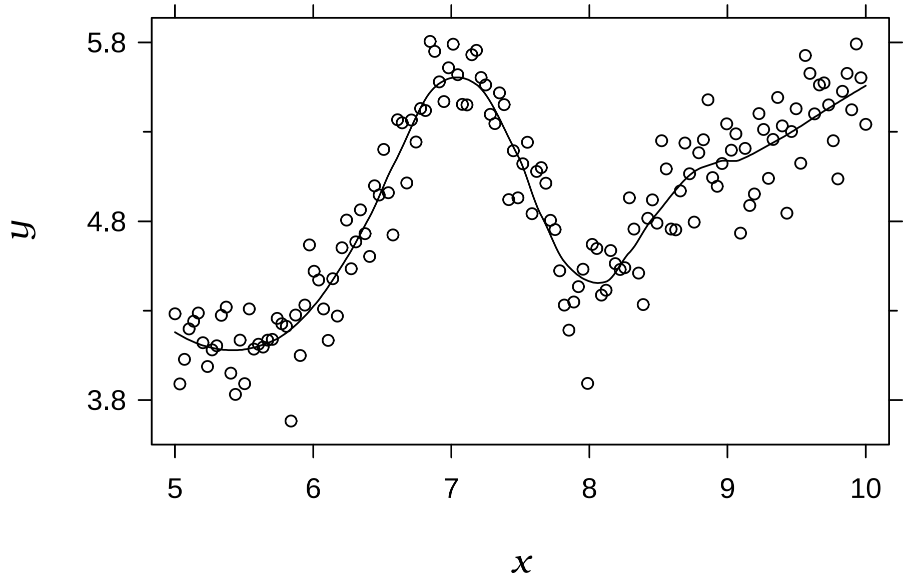
y, $\hat{y}$, and $\hat{\epsilon}$ are the vectors of values

$\hat{y} = Ly$

L is loess hat matrix

L depends on $x_{i1}, \ldots, x_{ip}, \alpha$, polynomial degree, and number of leaf cells

Least squares hat matrix is a projection operator but not L



Two Response Variables

Model Complexity

$\mu = \text{tr}LL'$

$\sum_{i=1}^{n} \text{Var}(\hat{y}_i) = \sigma^2\mu$

Linear least squares: μ = number of parameters

Loess: μ = equivalent number of parameters

Estimate of Noise Variance

Let I be $n \times n$ identity matrix

Estimate of σ^2

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} \hat{\epsilon}_i^2}{\text{tr}(I - L)(I - L)'}$$

$\hat{y}_i$: fitted values

$\hat{\epsilon}_i$: residuals

L: loess hat matrix

Third Response Variable: Mallows $C_p = M$

Estimate of mean-square error of prediction divided by σ^2

$$\begin{aligned}\hat{M} &= \frac{\sum_{i=1}^{n} \hat{\epsilon}_i^2}{\tilde{\sigma}^2} - \text{tr}(I - L)'(I - L) + \mu \\ &= \text{bias} + \text{variance}\end{aligned}$$

$\tilde{\sigma}^2$ from studying stabilization of $\hat{\sigma}^2$ as a function of μ

μ: complexity

$\hat{\epsilon}_i$: residuals

L: $n \times n$ hat matrix

Transformation: α and λ

Find transformations of tuning parameters: simplify dependence of μ on them

Guidance from $\mu \approx$ 1.2 (number of monomial fitting variables)/α

μ and α

- $\log_2(\mu)$ and $\log_2(1/\alpha)$

Two treatments of λ

- discrete variable
- $\tau = \log_2$(number of monomial fitting variables)

α: fraction of n points in neighborhood

λ: degree of polynomial

μ: complexity

Transformation: n_ℓ

n_ℓ, a power of 2

- $\log_2(n_\ell)$

An intrinsic, strong interaction between $\log_2(1/\alpha)$ and $\log_2(n_\ell)$

- need more leaf cells as the neighborhood size decreases

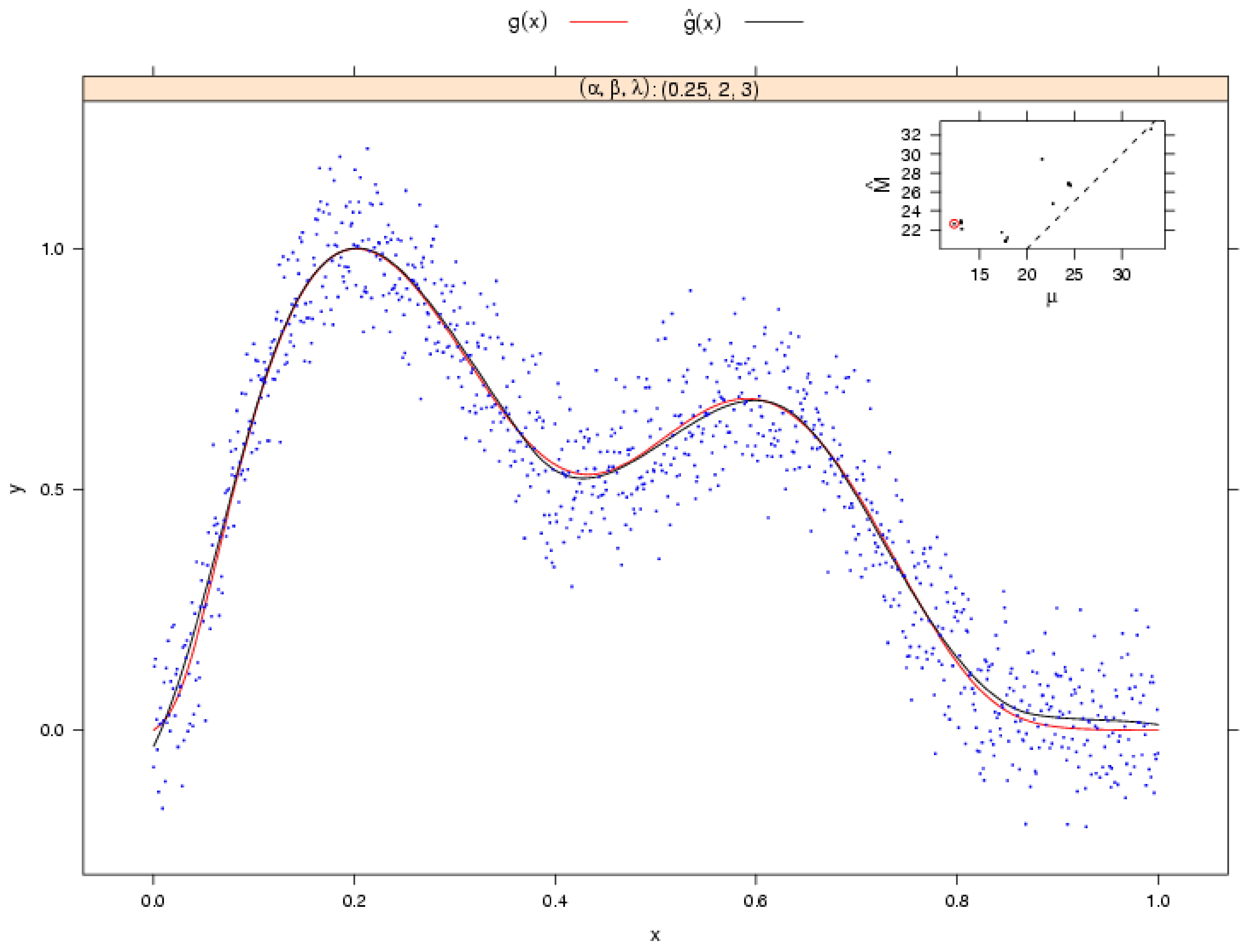
Measure of number of leaf cells per neighborhood:

$$\beta = \alpha n_\ell = \frac{n_\ell / n}{1/\alpha n}$$

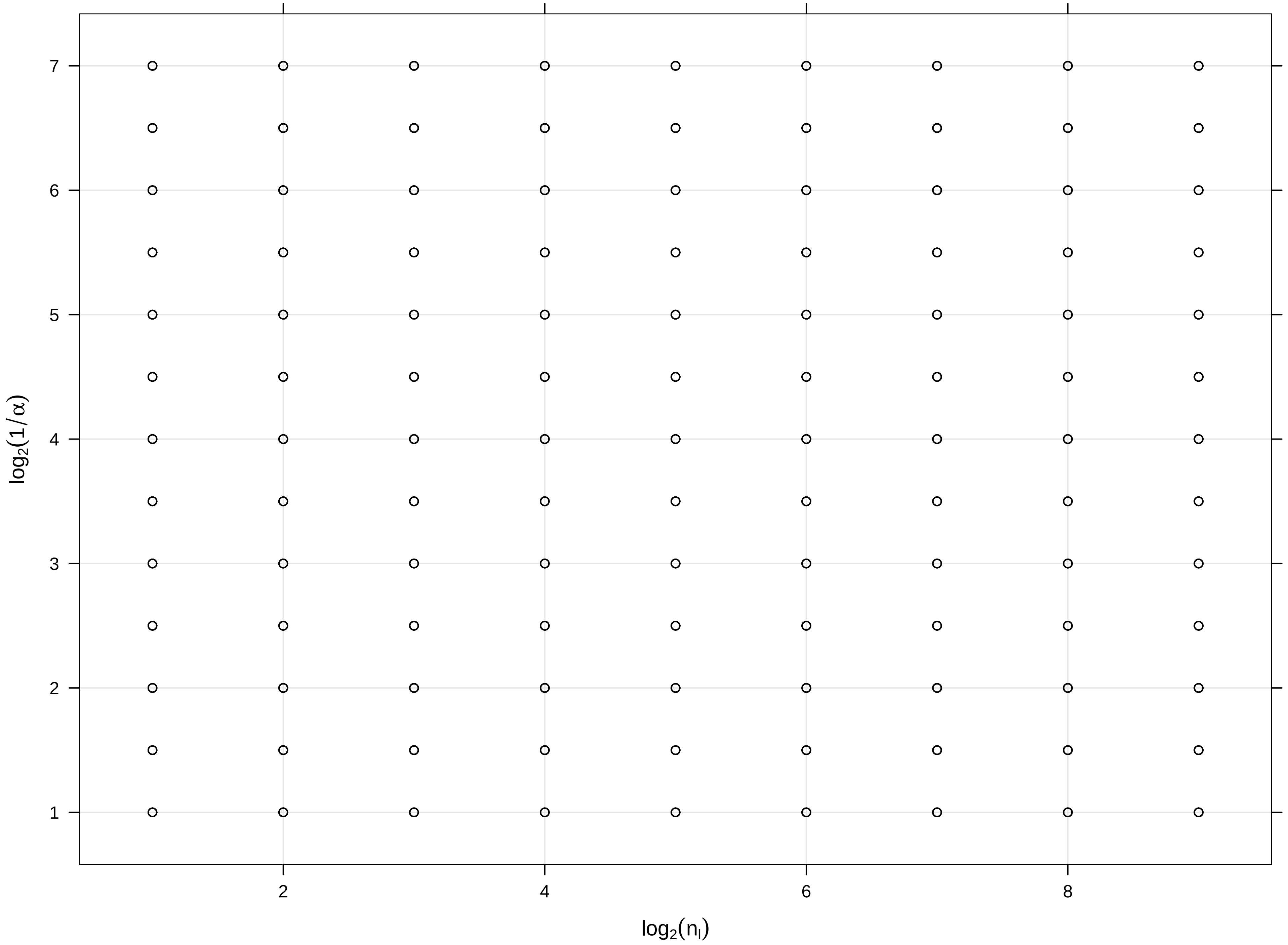
$\log_2(\beta)$: likely a reduced interaction with $\log_2(1/\alpha)$

α: fraction of n points in neighborhood n_l: number of leaf nodes

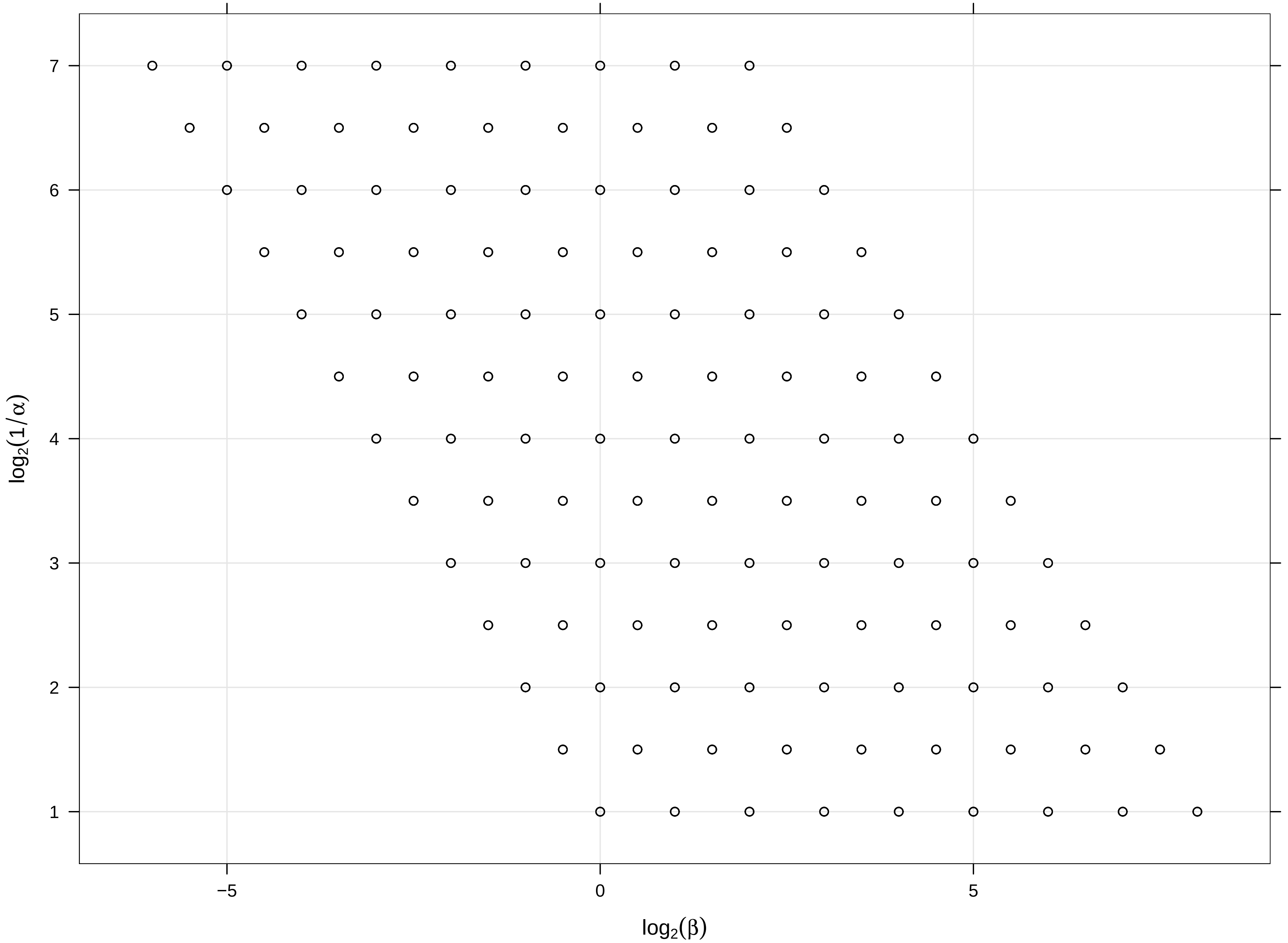
Camel Fit: λ = 3, α = 0.25, n_ℓ = 8, β = 2



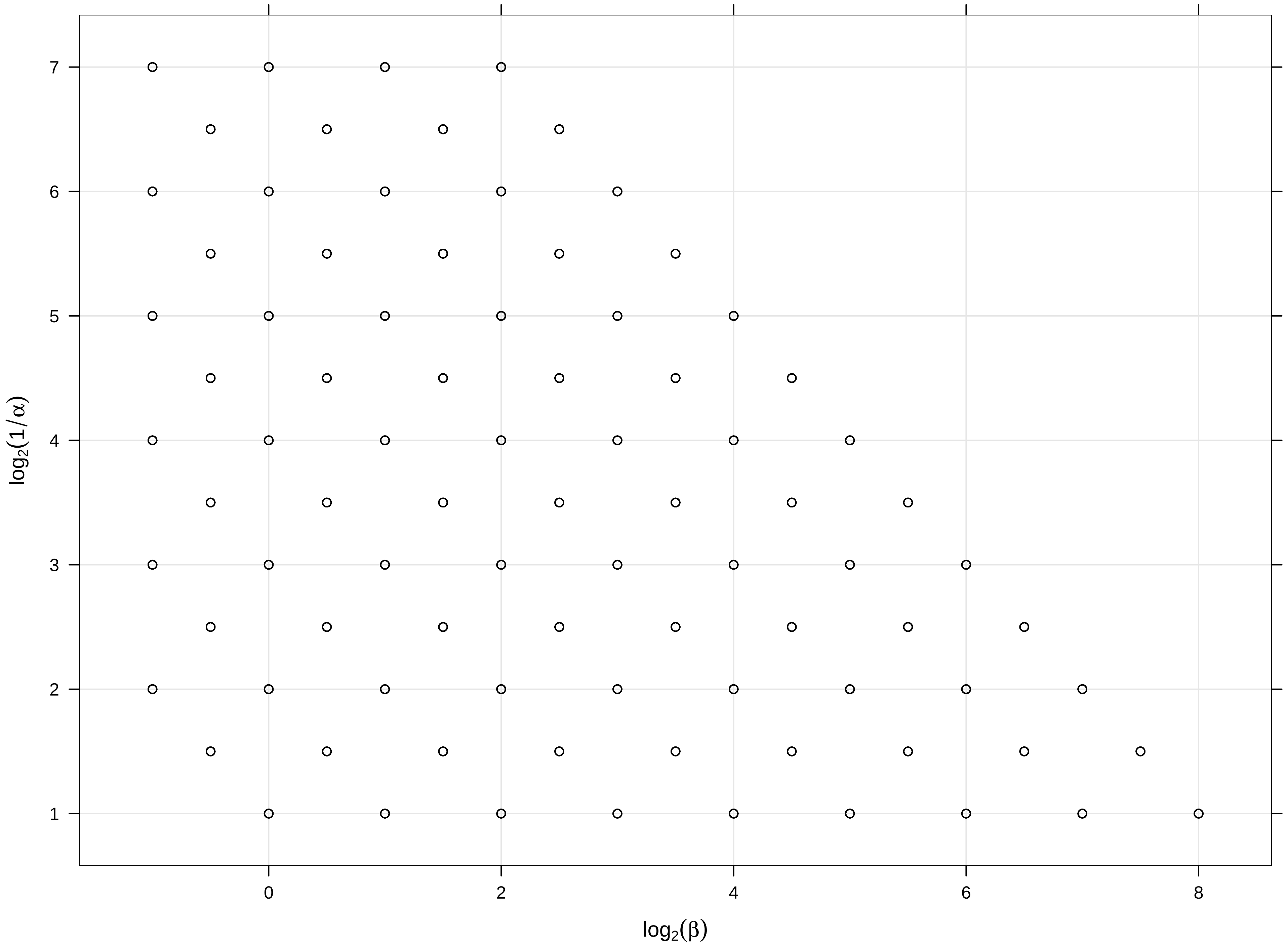
The n_ℓ & α Starting Design Space



The β & α Starting Design Space



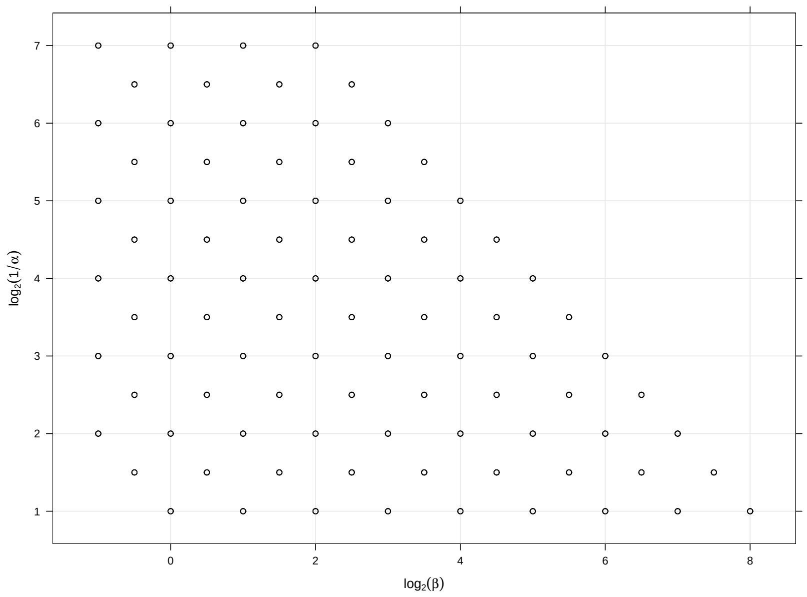
The Truncated β & α Design Space



The Design Space: β, α, and λ

Cross each pair of values of β & α with all four values of λ = 0, 1, 2, 3

87 values of β & α, so design space has $4 \times 87 = 348$ points

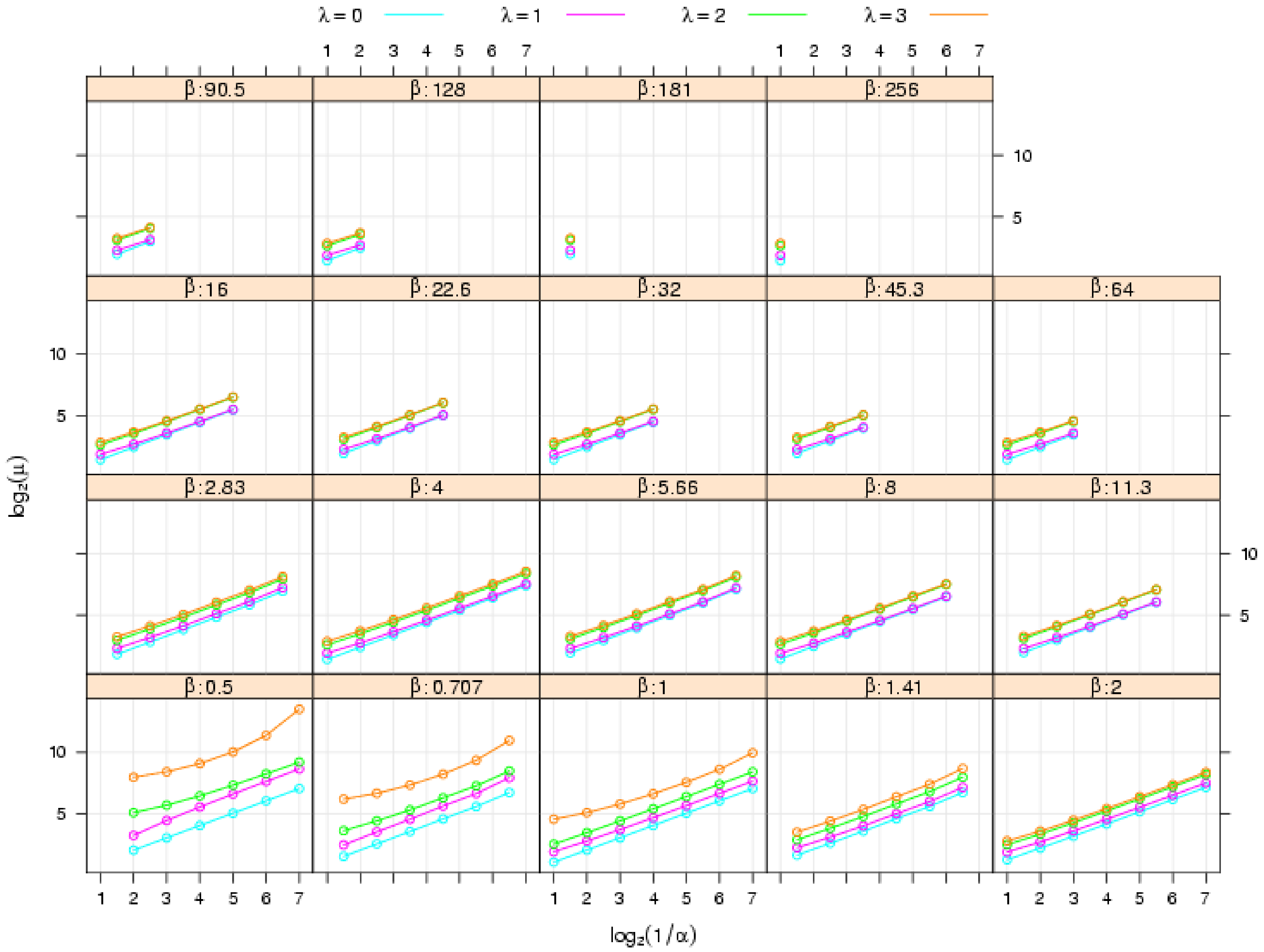


α: fraction of n points in neighborhood

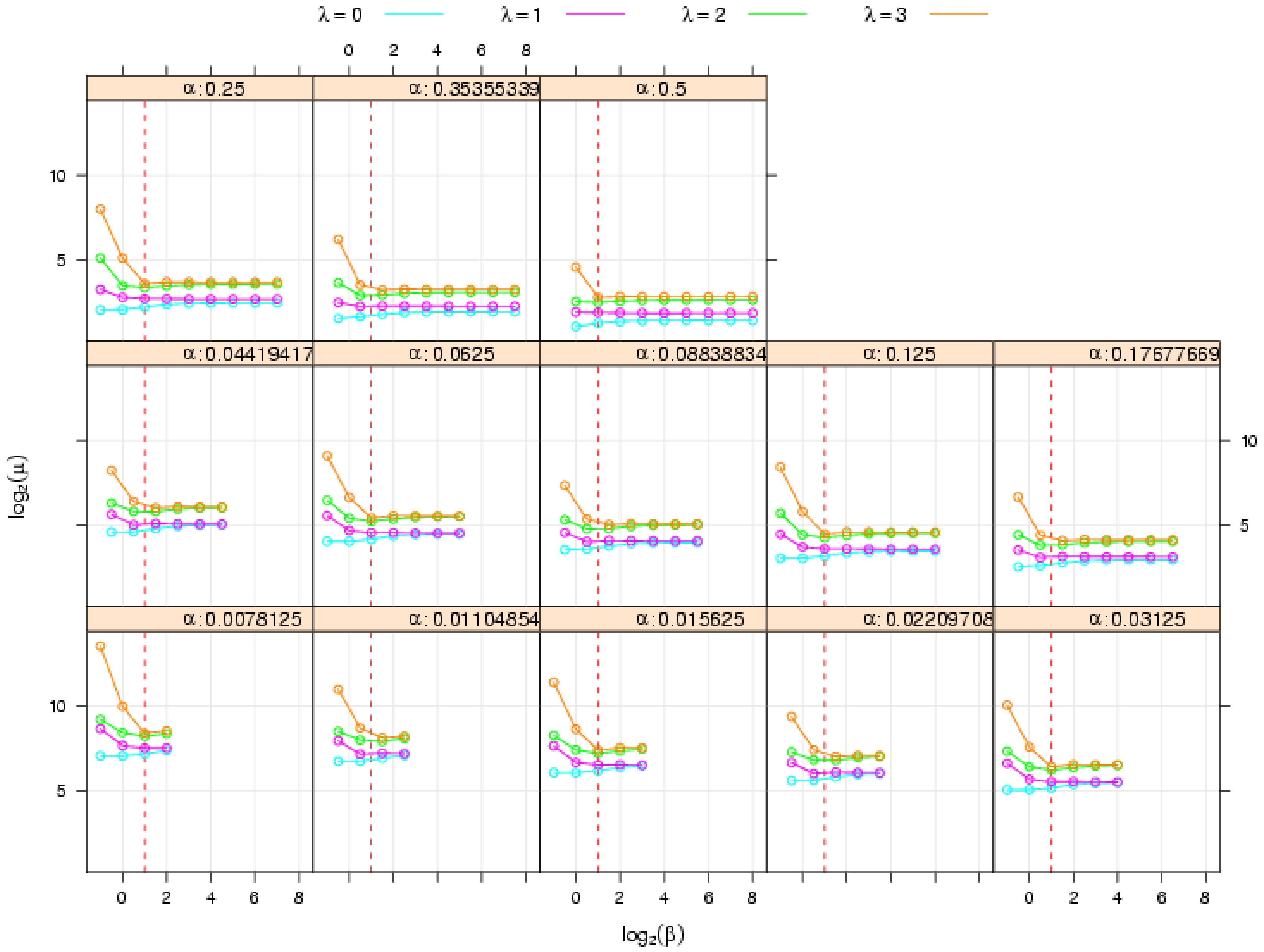
λ: degree of polynomial

β: number of leaf nodes per neighborhood

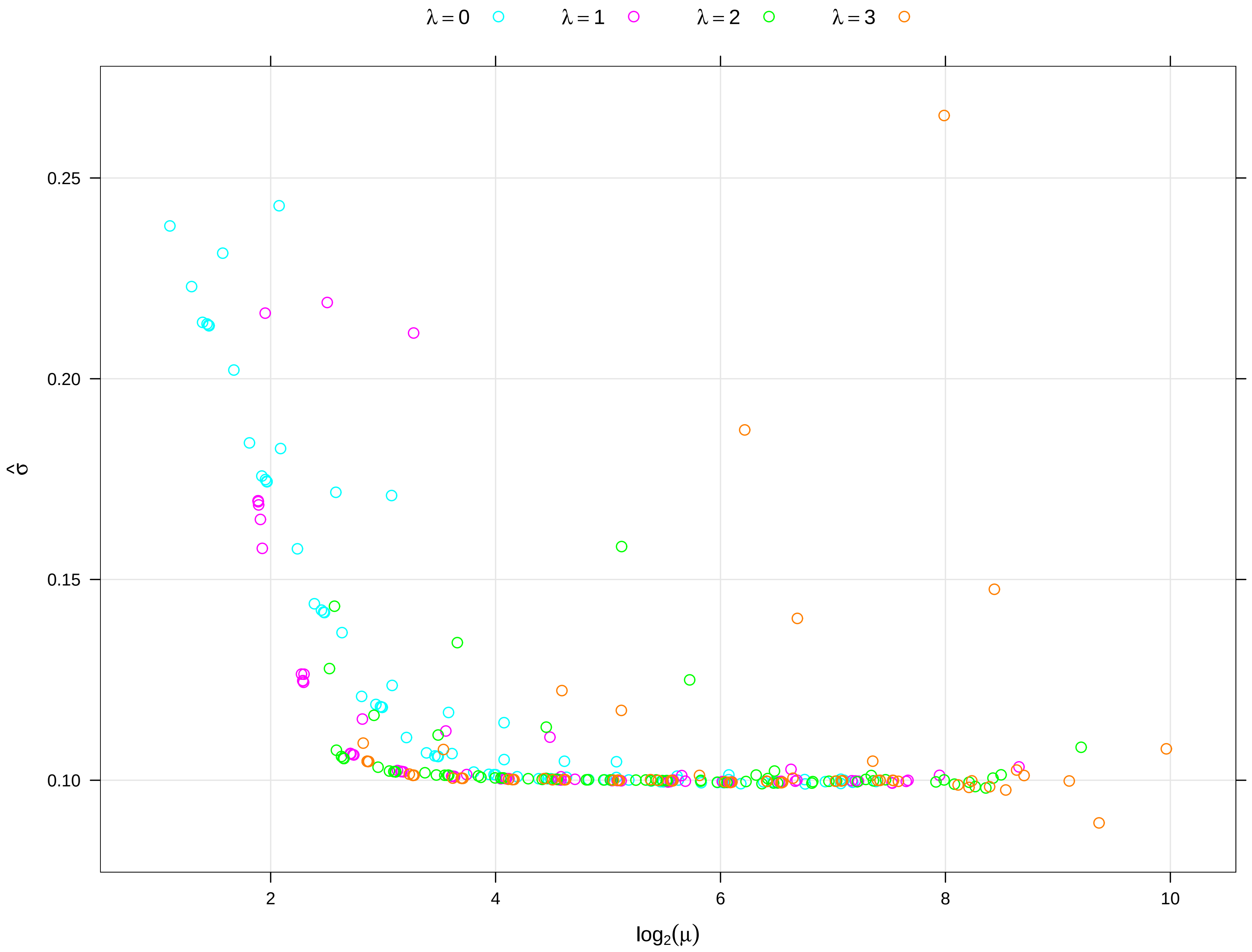
$\log_2(\mu)$ is approximately linear in $\log_2(1/\alpha)$ given β and λ



$\log_2(\mu)$ stabilizes as a function of $\log_2(\beta)$ in the range of about 2 to 3 leaf cells per neighborhood



$\hat{\sigma}$ can be estimated from this plot by the stabilized minimum value, close to the true value 0.1 in this case. This estimate is needed for $\hat{M}$.



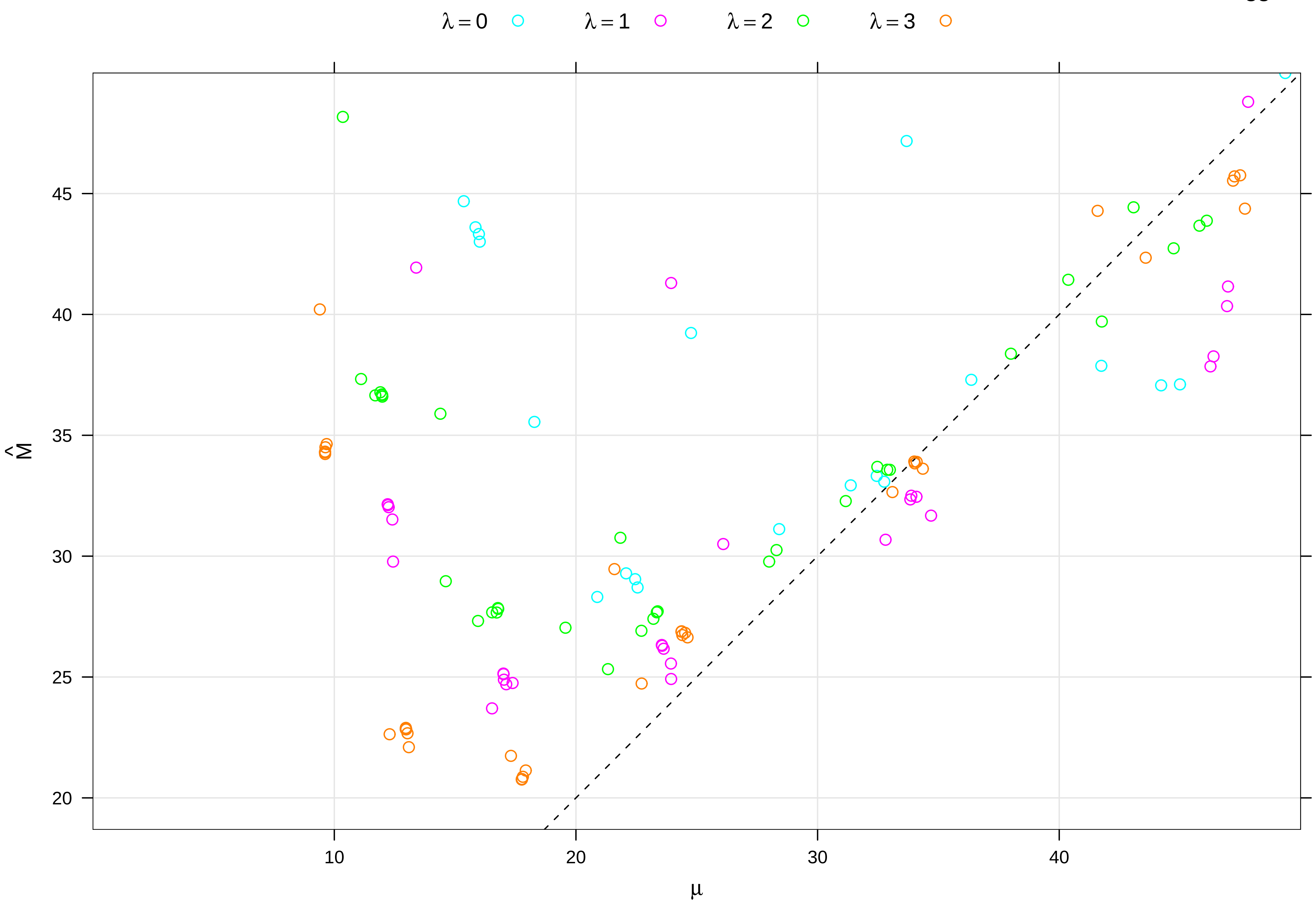
The M Plot

The next slide is the plot of $\hat{M}$ vs. μ

The admissible boundary is the lower envelope of the points

The model that minimizes $\hat{M}$ uses locally cubic fitting

The minimizer is not necessarily the best

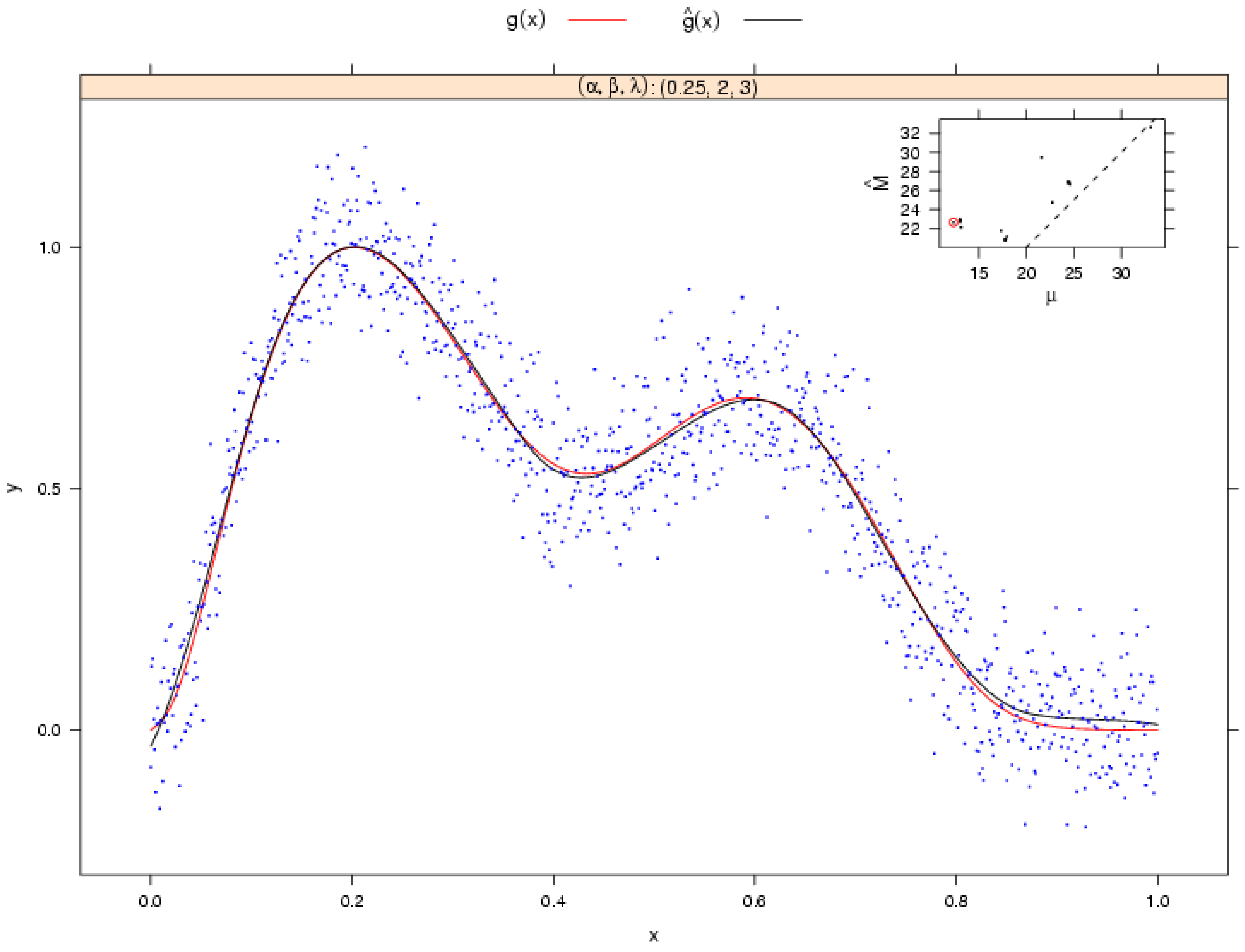


λ = 0 ○
λ = 1 ○
λ = 2 ○
λ = 3 ○
$\hat{M}$
μ
10
20
30
40
20
25
30
35
40
45

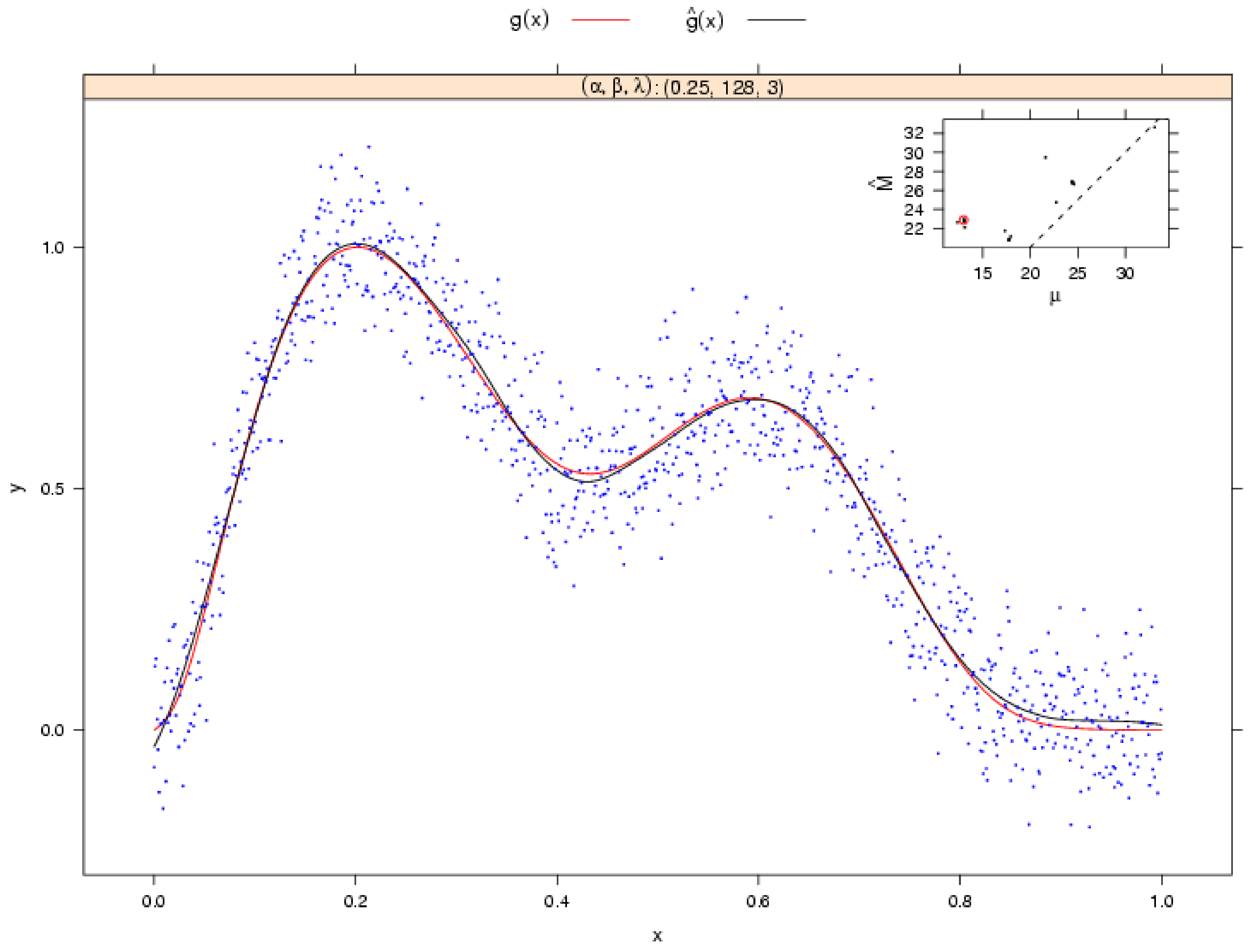
Fits Along Admissible Boundary of M Plot

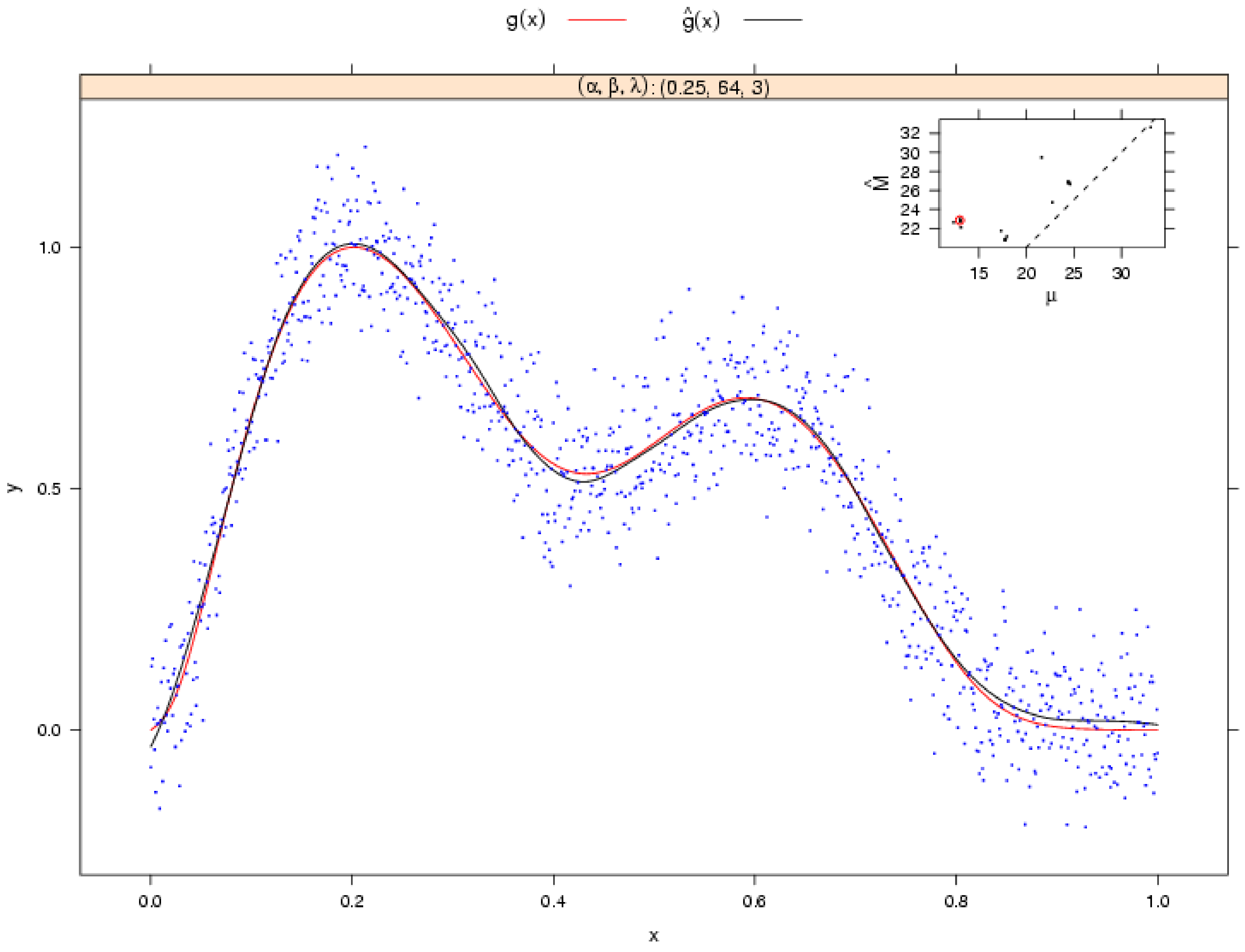
In the next slides we move from lower to higher μ through the admissible boundary for degree 3.

The insert shows the points of the admissible boundary and the red indicates the point for the current fit

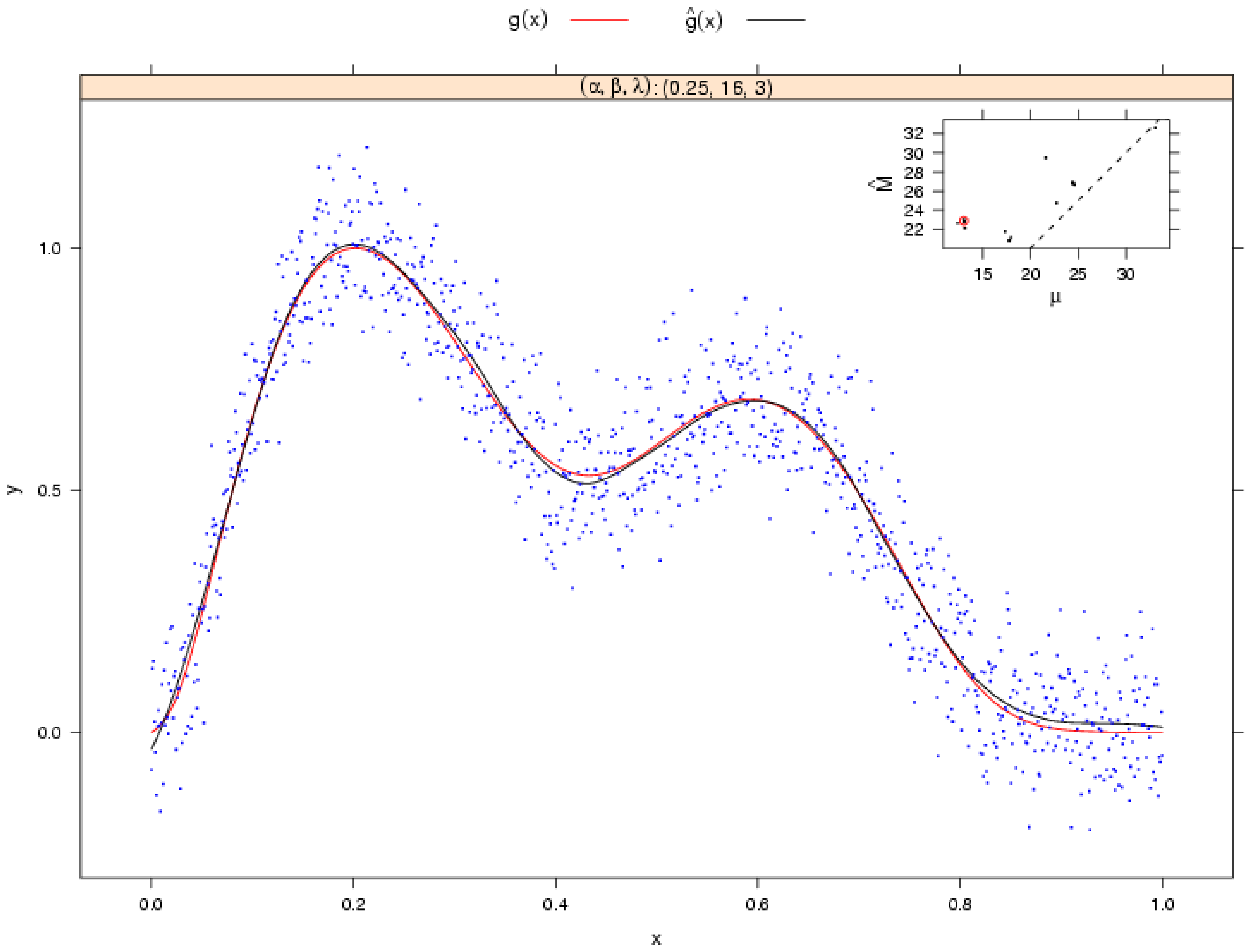


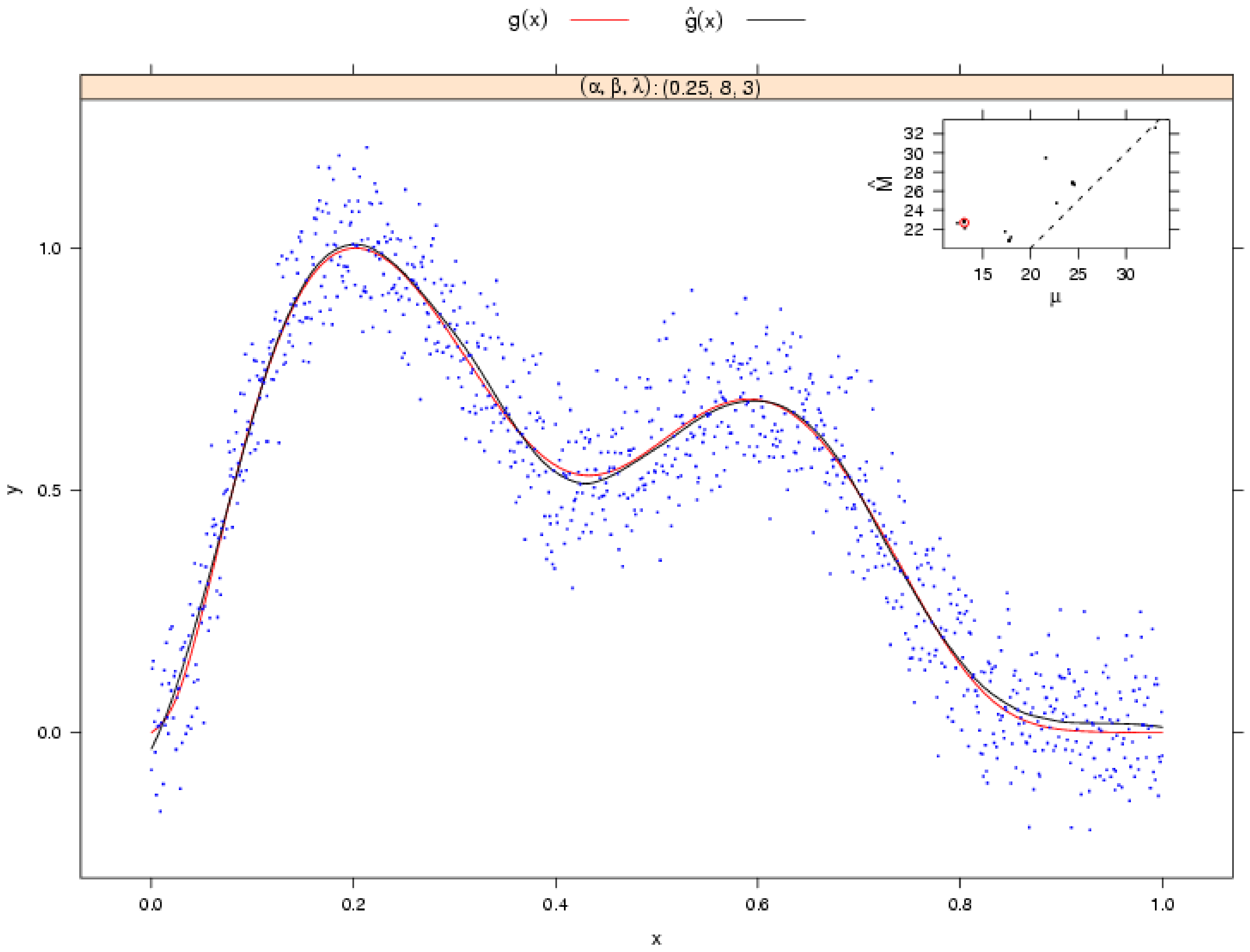
$g(x)$
$\hat{g}(x)$
(α, β, λ): (0.25, 2, 3)
y
0.0
0.5
1.0
x
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{M}$
22
24
26
28
30
32
μ
15
20
25
30


g(x)
$\hat{g}(x)$
(α, β, λ): (0.25, 128, 3)
$\hat{M}$
32
30
28
26
24
22
15
20
25
30
μ
y
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x

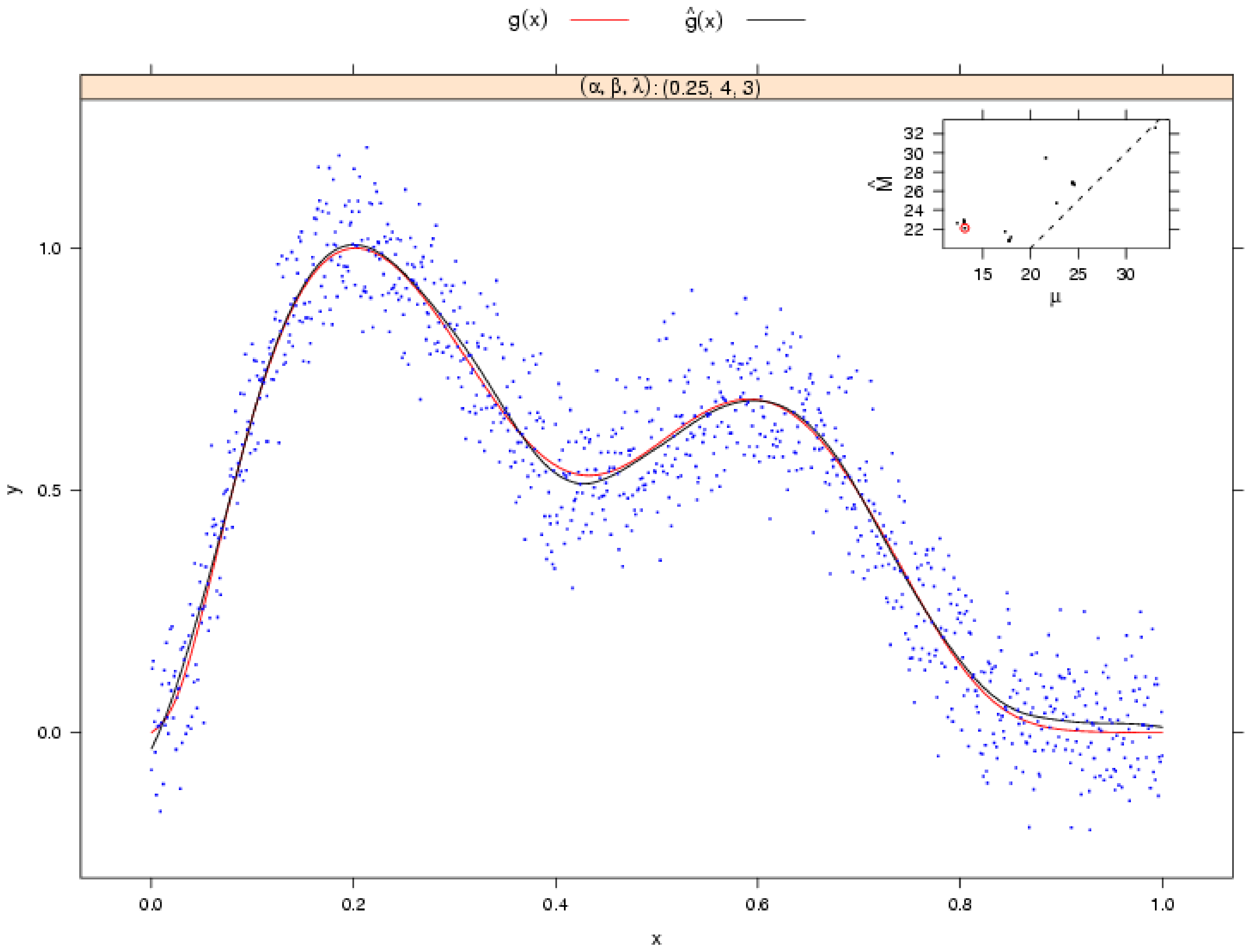


g(x)
ĝ(x)
(α, β, λ): (0.25, 64, 3)
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
M̂
μ
22
24
26
28
30
32
15
20
25
30

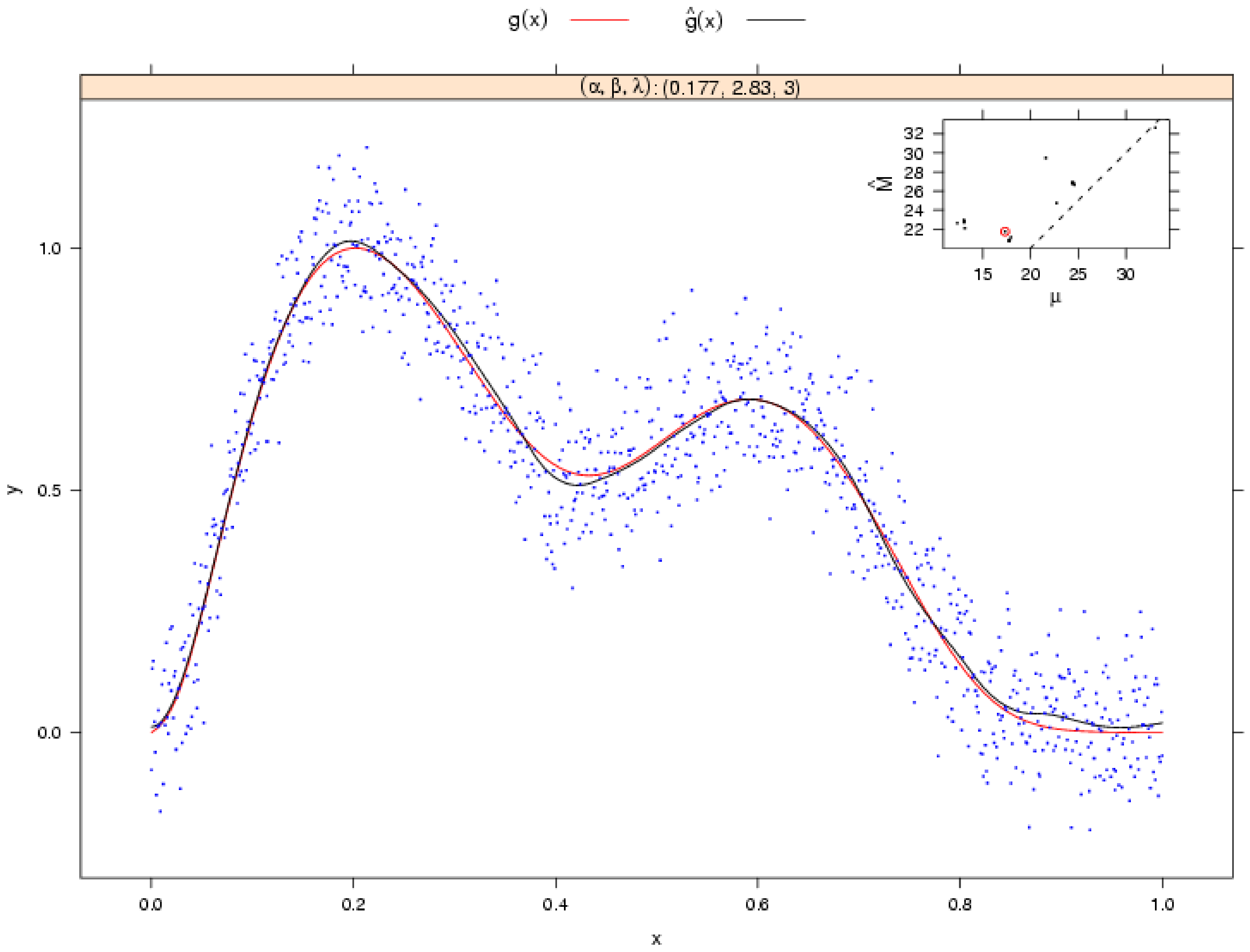

$g(x)$
$\hat{g}(x)$
$(\alpha, \beta, \lambda): (0.25, 16, 3)$
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{M}$
μ
22
24
26
28
30
32
15
20
25
30



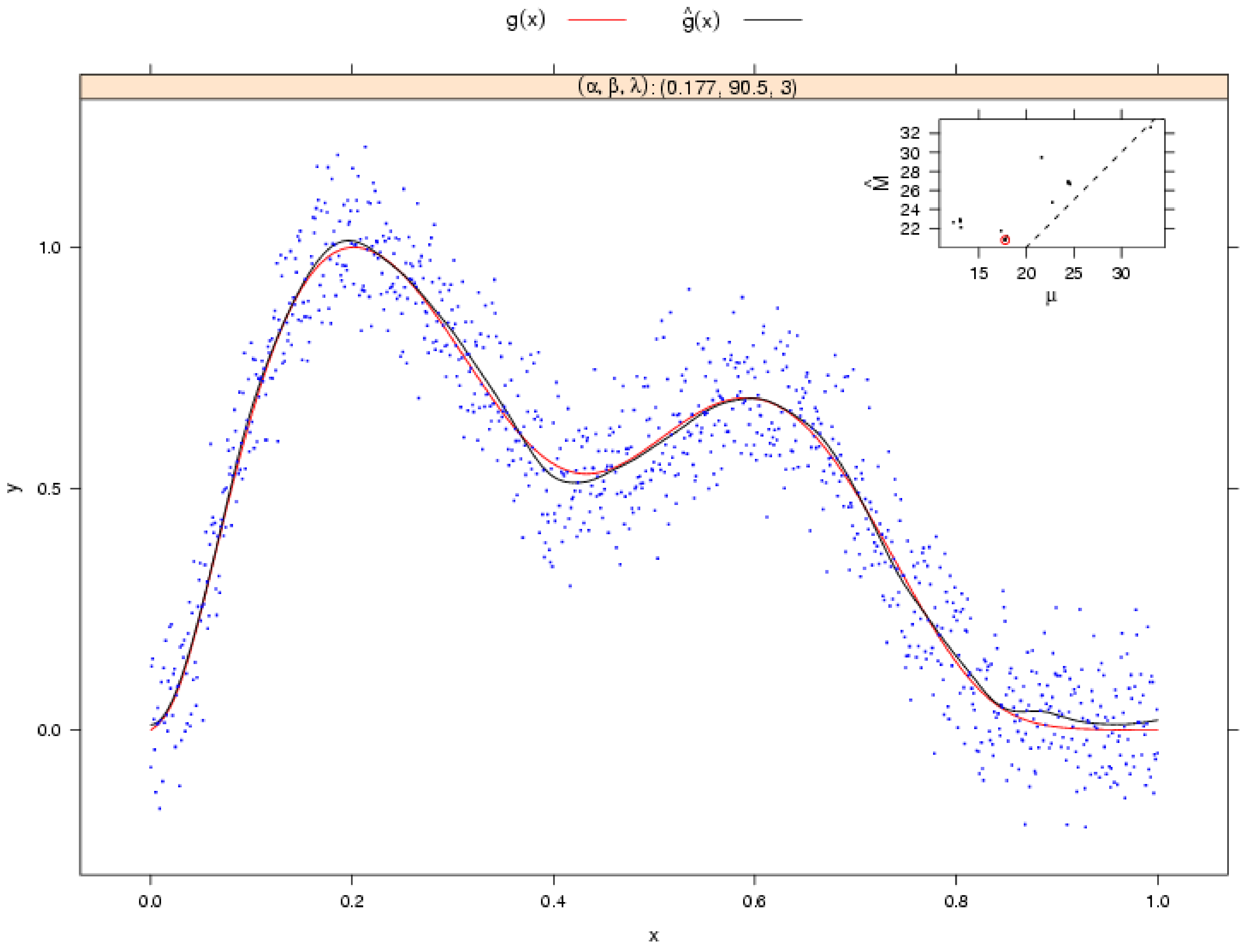
g(x)
ĝ(x)
(α, β, λ): (0.25, 8, 3)
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{M}$
μ
22
24
26
28
30
32
15
20
25
30



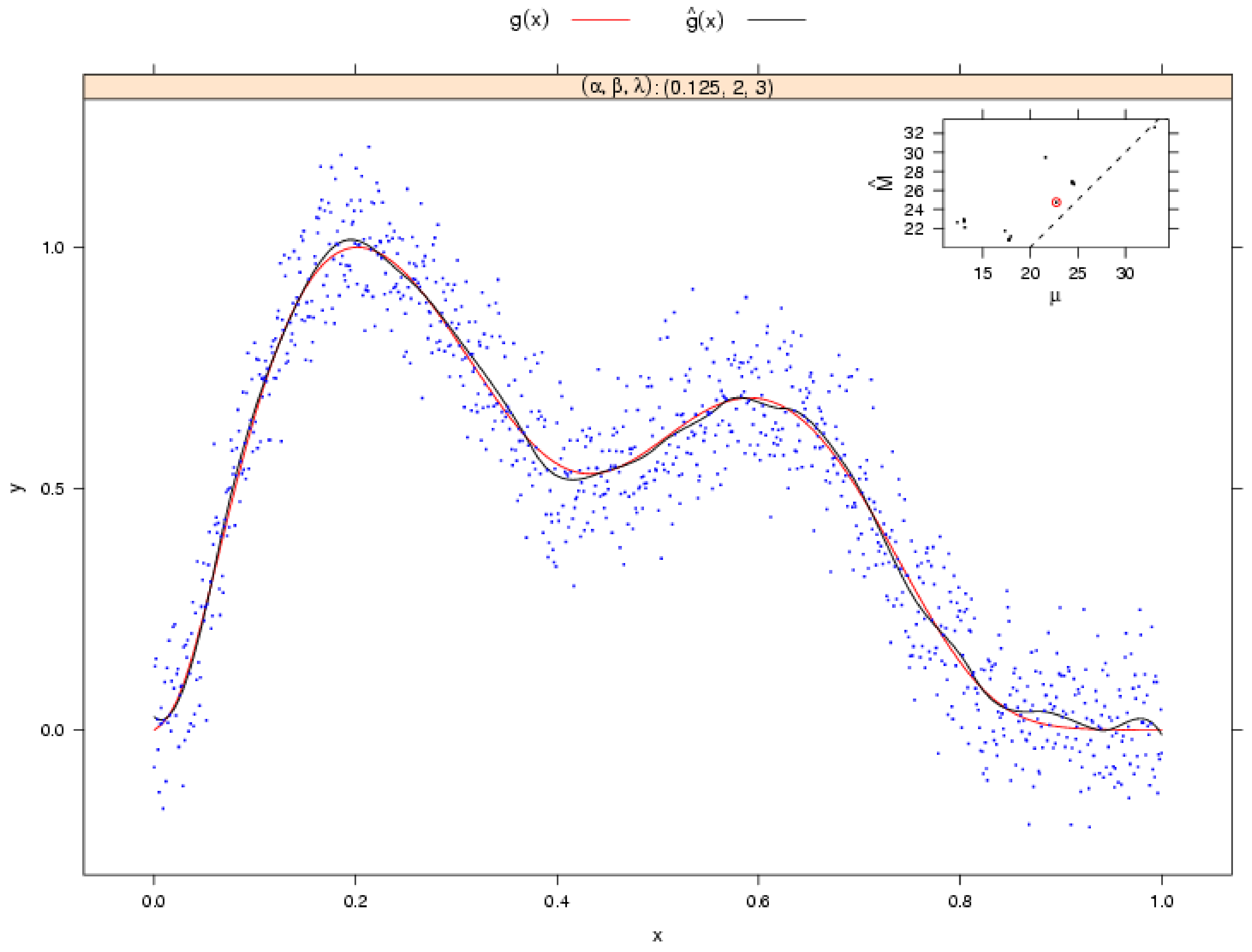
$g(x)$
$\hat{g}(x)$
(α, β, λ): (0.25, 4, 3)
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{M}$
μ
22
24
26
28
30
32
15
20
25
30



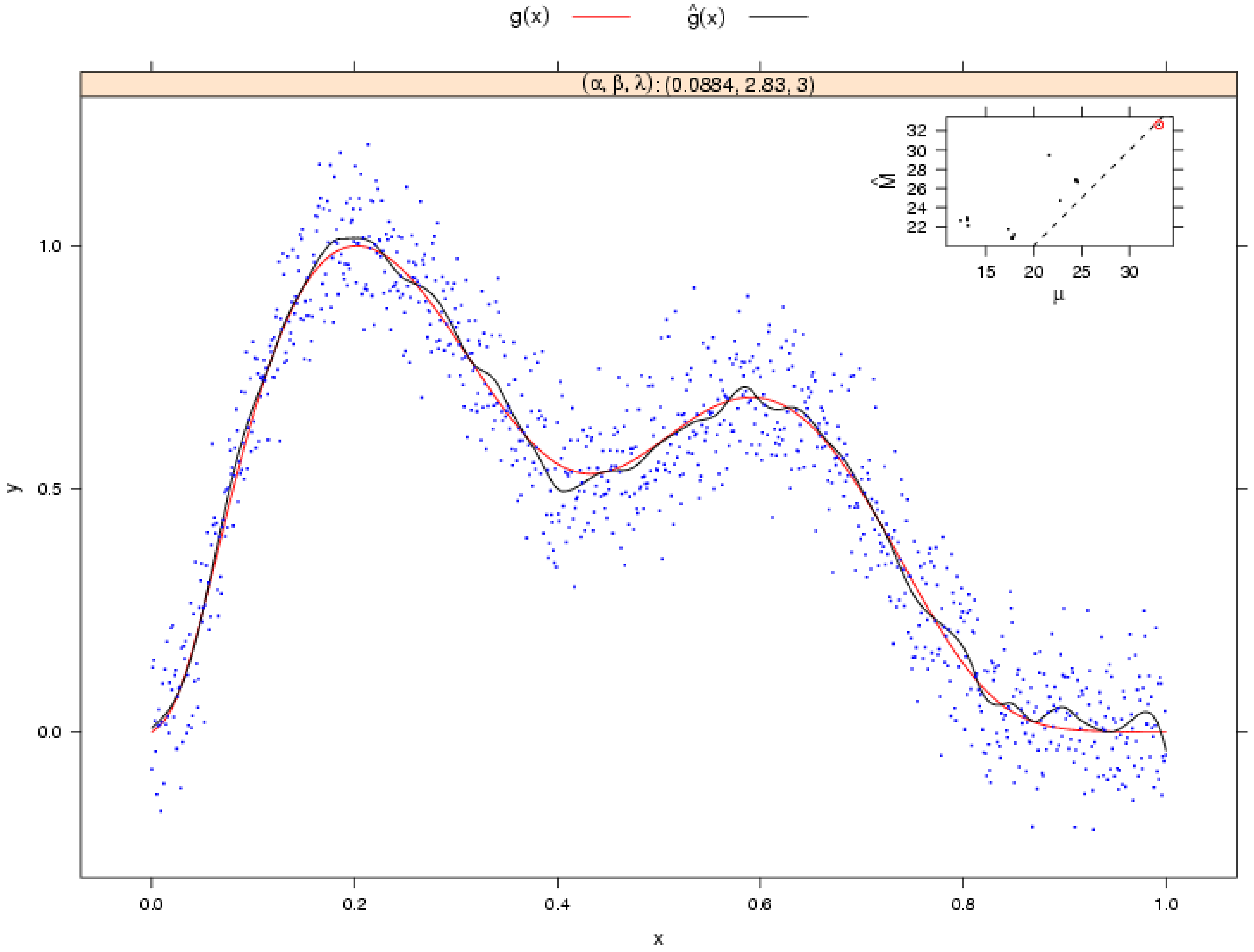
$g(x)$
$\hat{g}(x)$
(α, β, λ): (0.177, 2.83, 3)
32
30
28
26
24
22
$\hat{M}$
15
20
25
30
μ
1.0
0.5
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x



g(x)
ĝ(x)
(α, β, λ): (0.177, 90.5, 3)
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
M̂
μ
32
30
28
26
24
22
15
20
25
30



$g(x)$
$\hat{g}(x)$
(α, β, λ): (0.125, 2, 3)
y
x
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
$\hat{M}$
μ
32
30
28
26
24
22
15
20
25
30


$g(x)$
$\hat{g}(x)$
(α, β, λ): (0.0884, 2.83, 3)
$\hat{M}$
32
30
28
26
24
22
15
20
25
30
μ
y
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x

Why Experimental Methods?

Much more effective tuning parameter selection for the data at hand

- all models are wrong
- balance model prediction bias and variance
- a complex matter

Build up a knowledge base across experiments about the effects and properties of the tuning parameters

Expand the scope of the model selection

- greater numbers of tuning parameters
- success where optimizer would get lost

Leads naturally to an effective separation of estimating the noise variance and selecting the tuning parameters