Eight more Classic Machine Learning algorithms

Andrew W. Moore

Associate Professor

School of Computer Science

Carnegie Mellon University

www.cs.cmu.edu/~awm

awm@cs.cmu.edu

412-268-7599

8: Polynomial Regression

So far we've mainly been dealing with linear regression

X_1	X_2	Y
3	2	7
1	1	3
:	:	:

$$\mathbf{X}=\begin{bmatrix}3 & 2\\ 1 & 1\\ : & :\end{bmatrix} \qquad \mathbf{y}=\begin{bmatrix}7\\ 3\\ :\end{bmatrix}$$

$\mathbf{x}_1=(3,2)$.. $\quad y_1=7$..

$$\mathbf{Z}=\begin{bmatrix}1 & 3 & 2\\ 1 & 1 & 1\\ : & & :\end{bmatrix} \qquad \mathbf{y}=\begin{bmatrix}7\\ 3\\ :\end{bmatrix}$$

$\mathbf{z}_1=(1,3,2)$.. $\quad y_1=7$..

$\mathbf{z}_k=(1,x_{k1},x_{k2})$

$$\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$$

$$y^{est} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Quadratic Regression

It's trivial to do linear fits of fixed nonlinear basis functions

X_1	X_2	Y
3	2	7
1	1	3
:	:	:

X=

3	2
1	1
:	:

y=

7
3
:

$y_1=7$..

Z=

1	3	2	9	6	4
1	1	1	1	1	1
:					:

y=

7
3
:

$\mathbf{z}=(1\ ,\ x_1,\ \ x_2\ ,\ \ x_1^2, x_1x_2, x_2^2,)$

$$\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$$

$$y^{est} = \beta_0+ \beta_1 x_1+ \beta_2 x_2+ \beta_3 x_1^2 + \beta_4 x_1x_2 + \beta_5 x_2^2$$

Quadratic Regression

It's triv

X_1	X_2
3	2
1	1
:	:

Z=

1
1
:

z=(1 ,

Each component of a z vector is called a term.

Each column of the Z matrix is called a term column

How many terms in a quadratic regression with m inputs?

- 1 constant term
- m linear terms
- (m+1)-choose-2 = m(m+1)/2 quadratic terms

(m+2)-choose-2 terms in total = $O(m^2)$

Note that solving $\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$ is thus $O(m^6)$

Q^{th}-degree polynomial Regression

X_1	X_2	Y
3	2	7
1	1	3
:	:	:

X=

3	2
1	1
:	:

y=

7
3
:

Z=

1	3	2	9	6	…
1	1	1	1	1	…
:					…

y=

7
3
:

z=(all products of powers of inputs in which sum of powers is q or less,)

$\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$

$y^{est} = \beta_0 + \beta_1 x_1 + \ldots$

m inputs, degree Q: how many terms?

= the number of unique terms of the form

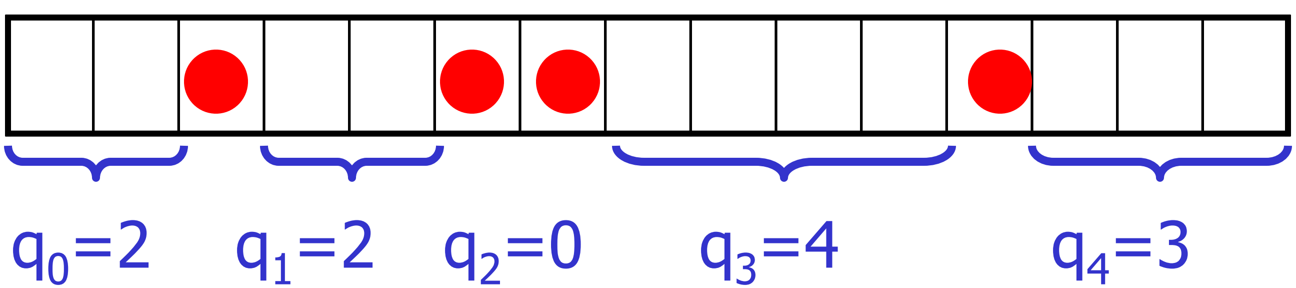
$$x_1^{q_1} x_2^{q_2} \ldots x_m^{q_m} \text{ where } \sum_{i=1}^{m} q_i \le Q$$

= the number of unique terms of the form

$$1^{q_0} x_1^{q_1} x_2^{q_2} \ldots x_m^{q_m} \text{ where } \sum_{i=0}^{m} q_i = Q$$

= the number of lists of non-negative integers $[q_0, q_1, q_2, .. q_m]$ in which $\Sigma q_i = Q$

= the number of ways of placing Q red disks on a row of squares of length Q+m = (Q+m)-choose-Q



Q=11, m=4

7: Radial Basis Functions (RBFs)

X_1	X_2	Y
3	2	7
1	1	3
:	:	:

X=

3	2
1	1
:	:

y=

7
3
:

Z=

…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

y=

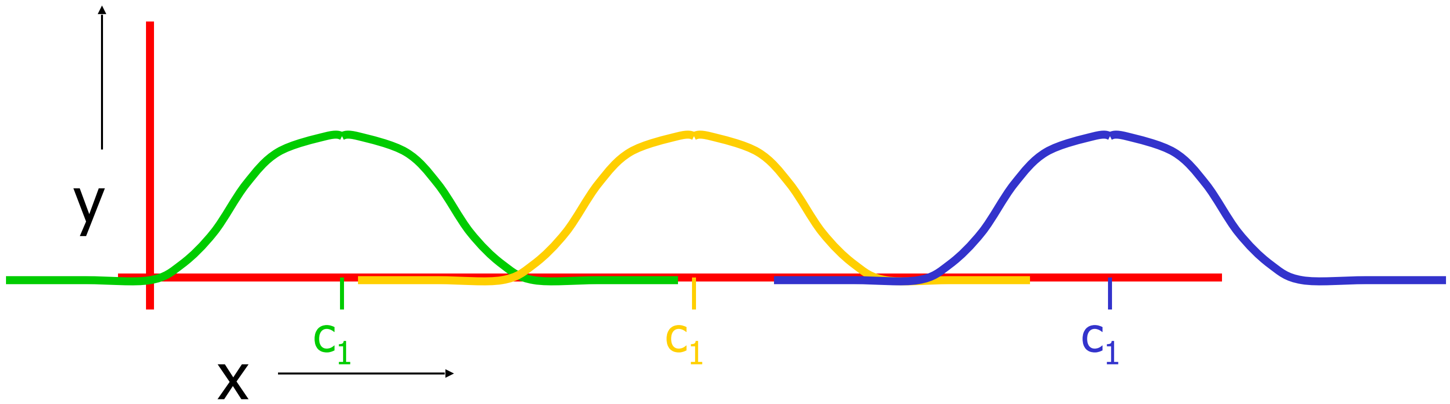
7
3
:

z=(list of radial basis function evaluations)

$\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$

$y^{est} = \beta_0 + \beta_1 x_1 + \ldots$

1-d RBFs

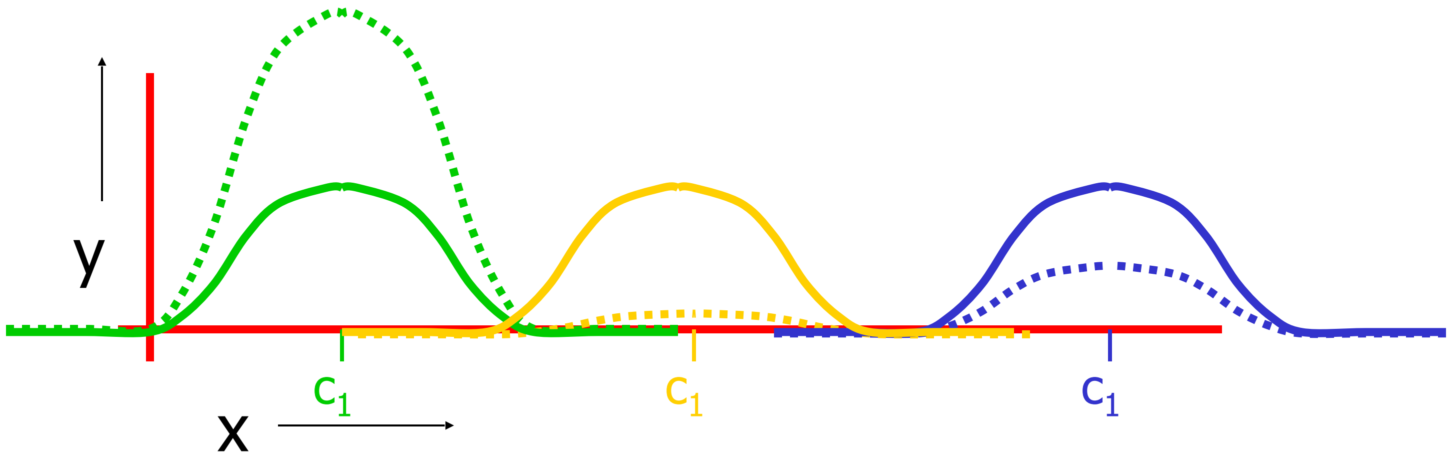


$$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$$

where

$$\phi_i(x) = \text{KernelFunction}(\ |\ x - c_i\ |\ /\ KW)$$

Example

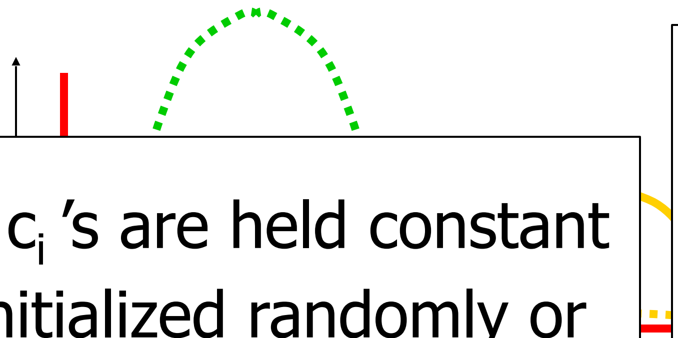


$$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$$

where

$$\phi_i(x) = \text{KernelFunction}(\mid x - c_i \mid / \text{KW})$$

RBFs with Linear Regression



All c_i 's are held constant (initialized randomly or on a grid in m-dimensional input space)

KW also held constant (initialized to be large enough that there's decent overlap between basis functions*

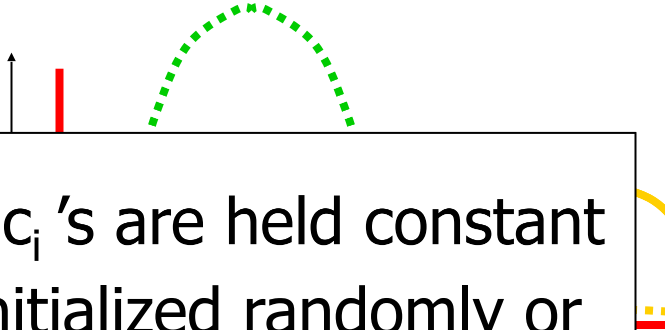
*Usually much better than the crappy overlap on my diagram

$y^{est} = 2\phi_1(x) + 0.05\phi_2($ … $.5\phi_3(x)$

where

$\phi_i(x) = \text{KernelFunction}(\ |\ x - c_i\ |\ /\ KW)$

RBFs with Linear Regression



All c_i's are held constant (initialized randomly or on a grid in m-dimensional input space)

KW also held constant (initialized to be large enough that there's decent overlap between basis functions*

*Usually much better than the crappy overlap on my diagram

$y^{est} = 2\phi_1(x) + 0.05\phi_2(x$ … $5\phi_3(x)$

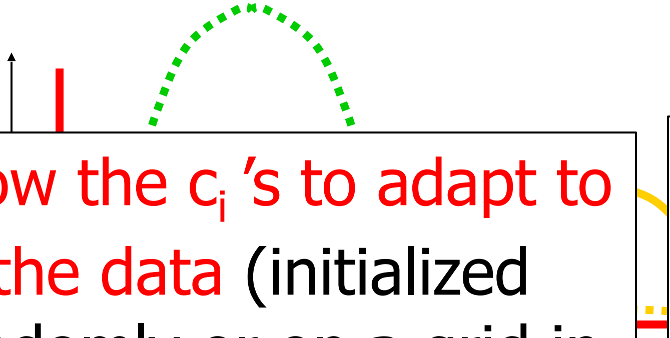
where

$\phi_i(x) = \text{KernelFunction}(\mid x - c_i \mid / KW)$

then given Q basis functions, define the matrix Z such that $Z_{kj} = \text{KernelFunction}(\mid x_k - c_i \mid / KW)$ where x_k is the kth vector of inputs

And as before, $\beta=(\mathbf{Z}^T\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{y})$

RBFs with NonLinear Regression



Allow the c_i's to adapt to the data (initialized randomly or on a grid in m-dimensional input space)

KW allowed to adapt to the data. (Some folks even let each basis function have its own KW_j, permitting fine detail in dense regions of input space)

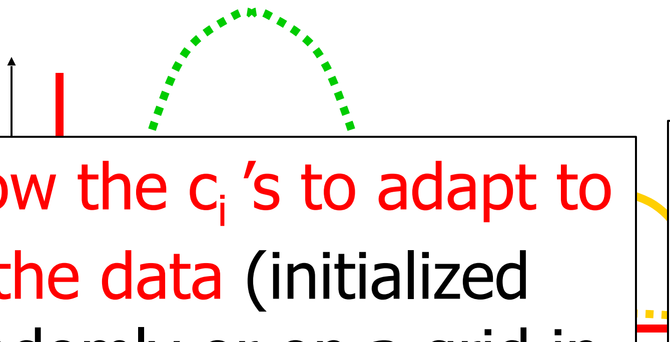
$y^{est} = 2\phi_1(x) + 0.05\phi_2(x$ … $5\phi_3(x)$

where

$\phi_i(x) = \text{KernelFunction}(\ | x - c_i | \ / \ KW)$

But how do we now find all the β_j's, c_i's and KW ?

RBFs with NonLinear Regression



Allow the c_i's to adapt to the data (initialized randomly or on a grid in m-dimensional input space)

KW allowed to adapt to the data. (Some folks even let each basis function have its own KW_j, permitting fine detail in dense regions of input space)

$y^{est} = 2\phi_1(x) + 0.05\phi_2(x$ … $5\phi_3(x)$

where

$\phi_i(x) = \text{KernelFunction}(\ |x - c_i|\ /\ KW)$

But how do we now find all the β_j's, c_i's and KW ?

Answer: Gradient Descent

RBFs with NonLinear Regression

Allow the c_i's to adapt to the data (initialized randomly or on a grid in m-dimensional input space)

KW allowed to adapt to the data. (Some folks even let each basis function have its own KW_j, permitting fine detail in dense regions of input space)

$y^{est} = 2\phi_1(x) + 0.05\phi_2(x$ … $5\phi_3(x)$

where

$\phi_i(x) = \text{KernelFunction}(\ | x - c_i | \ / \ KW)$

But how do we now find all the β_j's, c_i's and KW ?

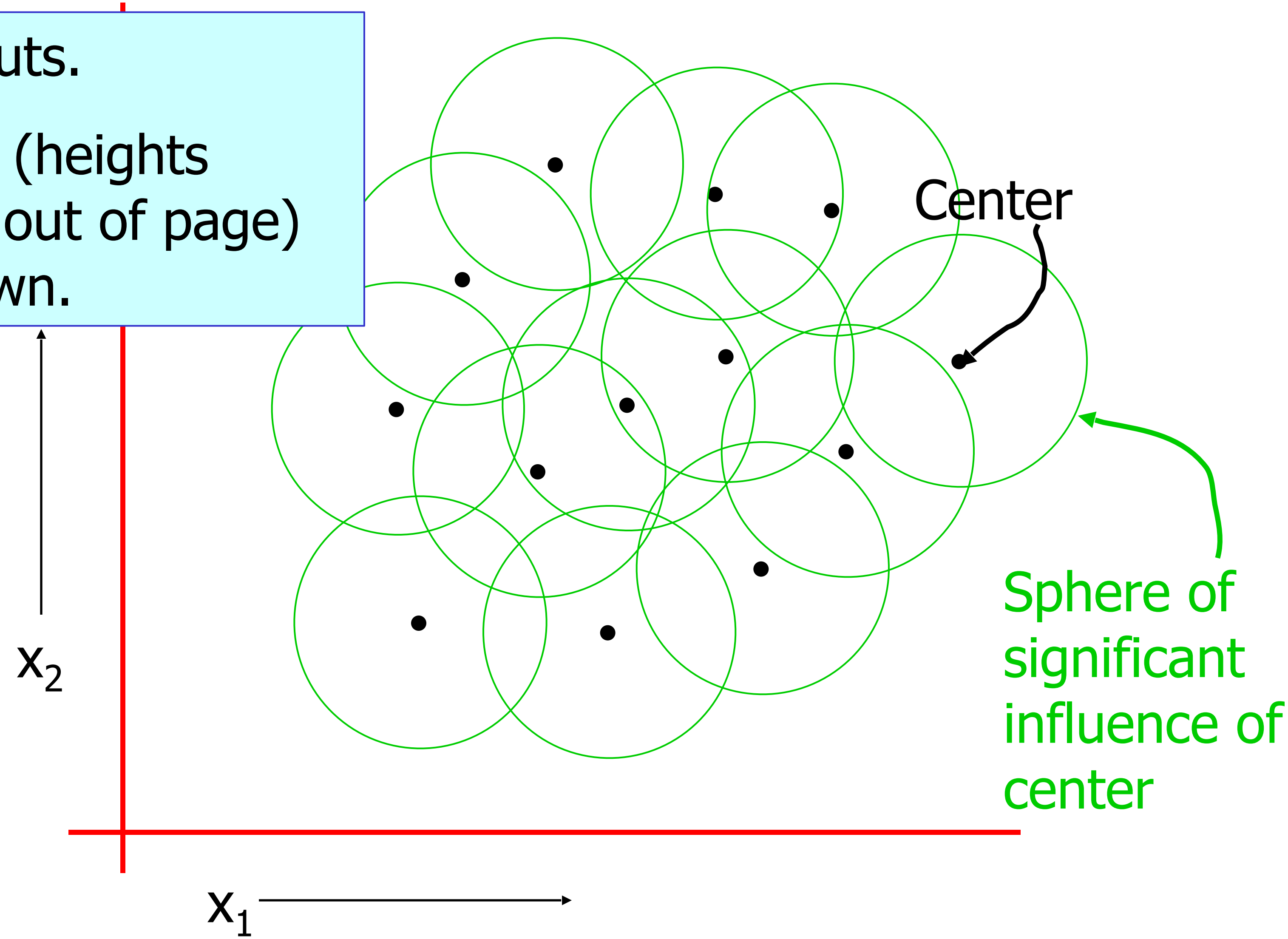
Answer: Gradient Descent

(But I'd like to see, or hope someone's already done, a hybrid, where the c_i's and KW are updated with gradient descent while the β_j's use matrix inversion)

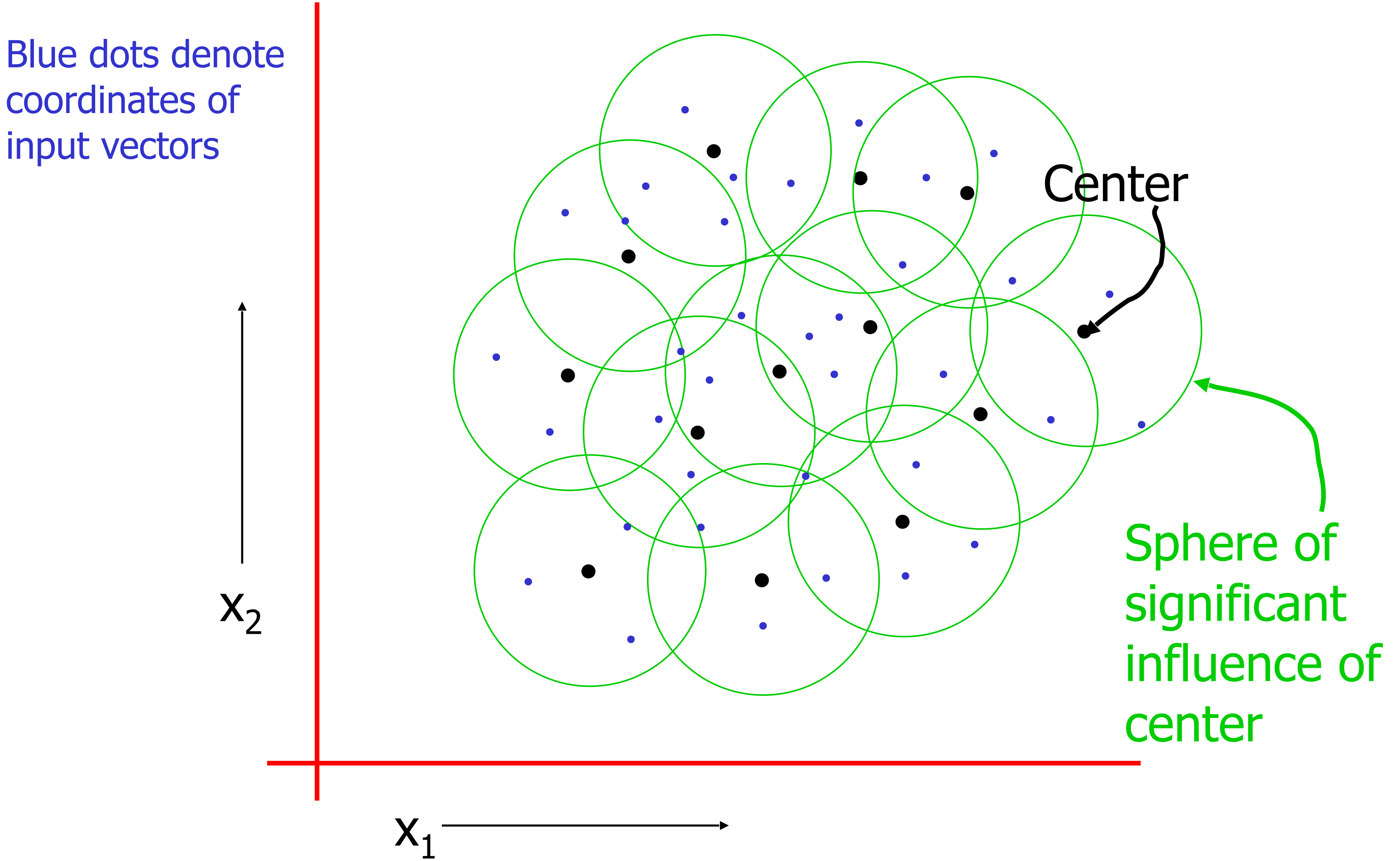
Radial Basis Functions in 2-d

Two inputs.

Outputs (heights sticking out of page) not shown.

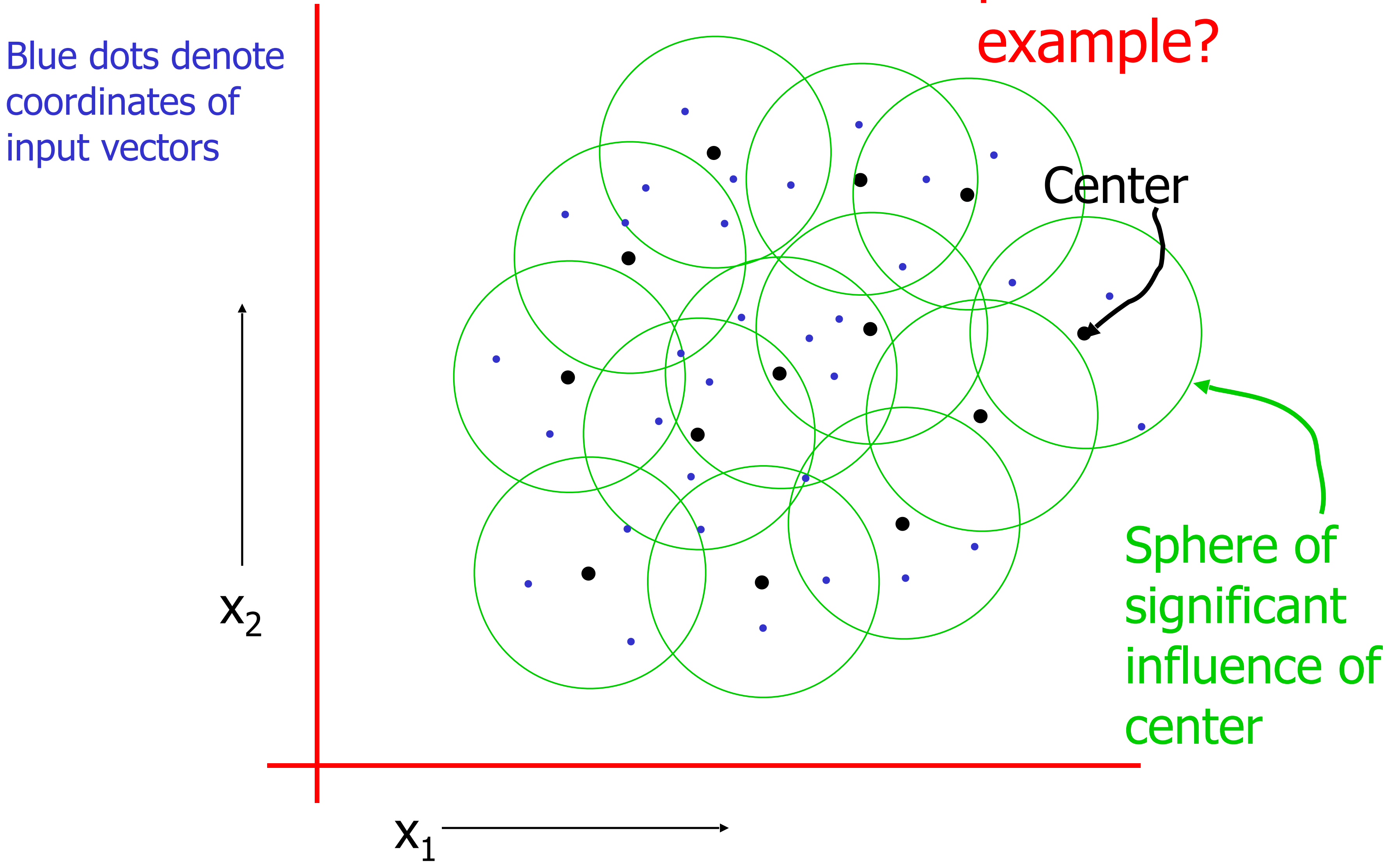


Happy RBFs in 2-d



Crabby RBFs in 2-d

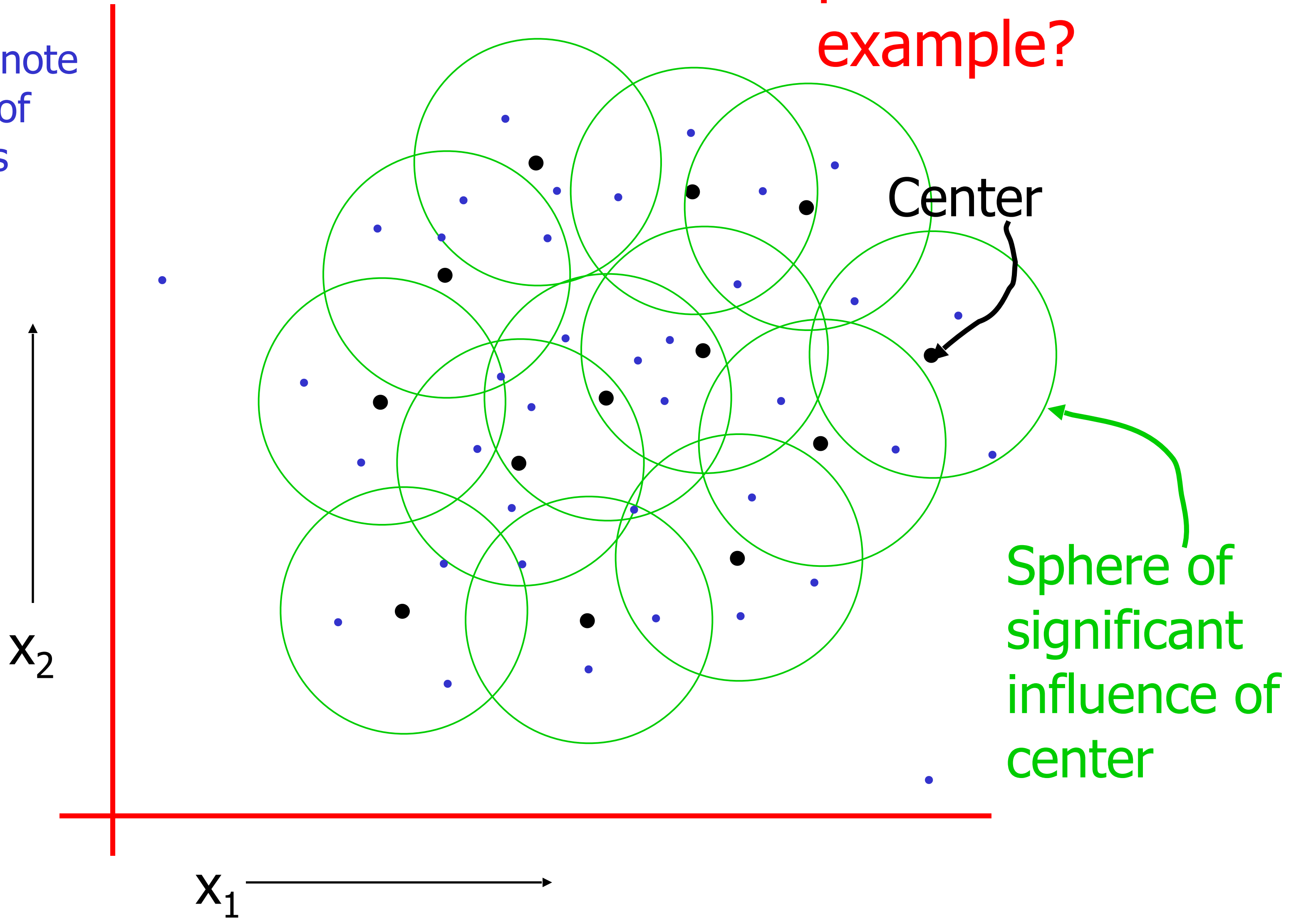
What's the problem in this example?



More crabby RBFs

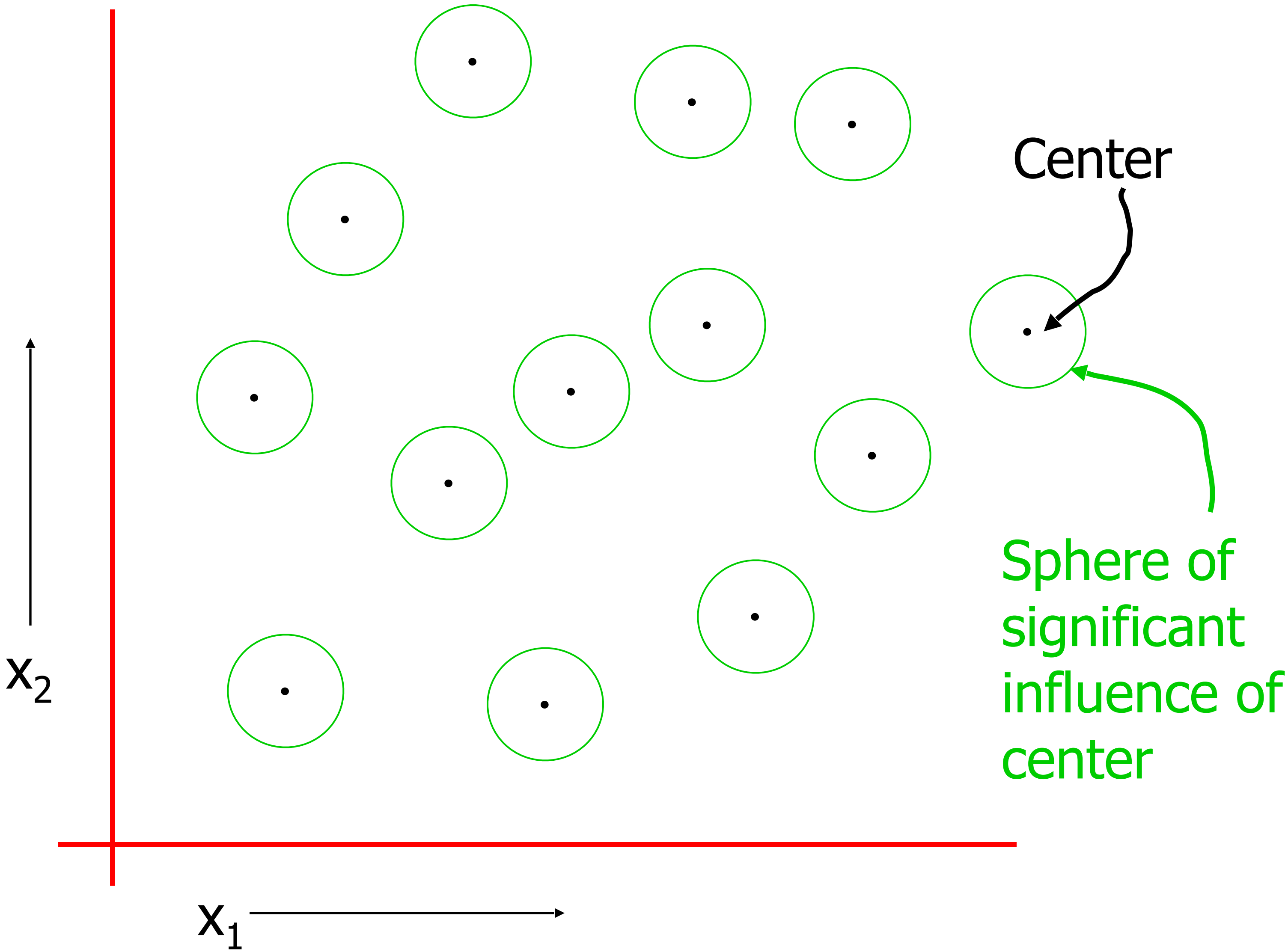
And what's the problem in this example?

Blue dots denote coordinates of input vectors



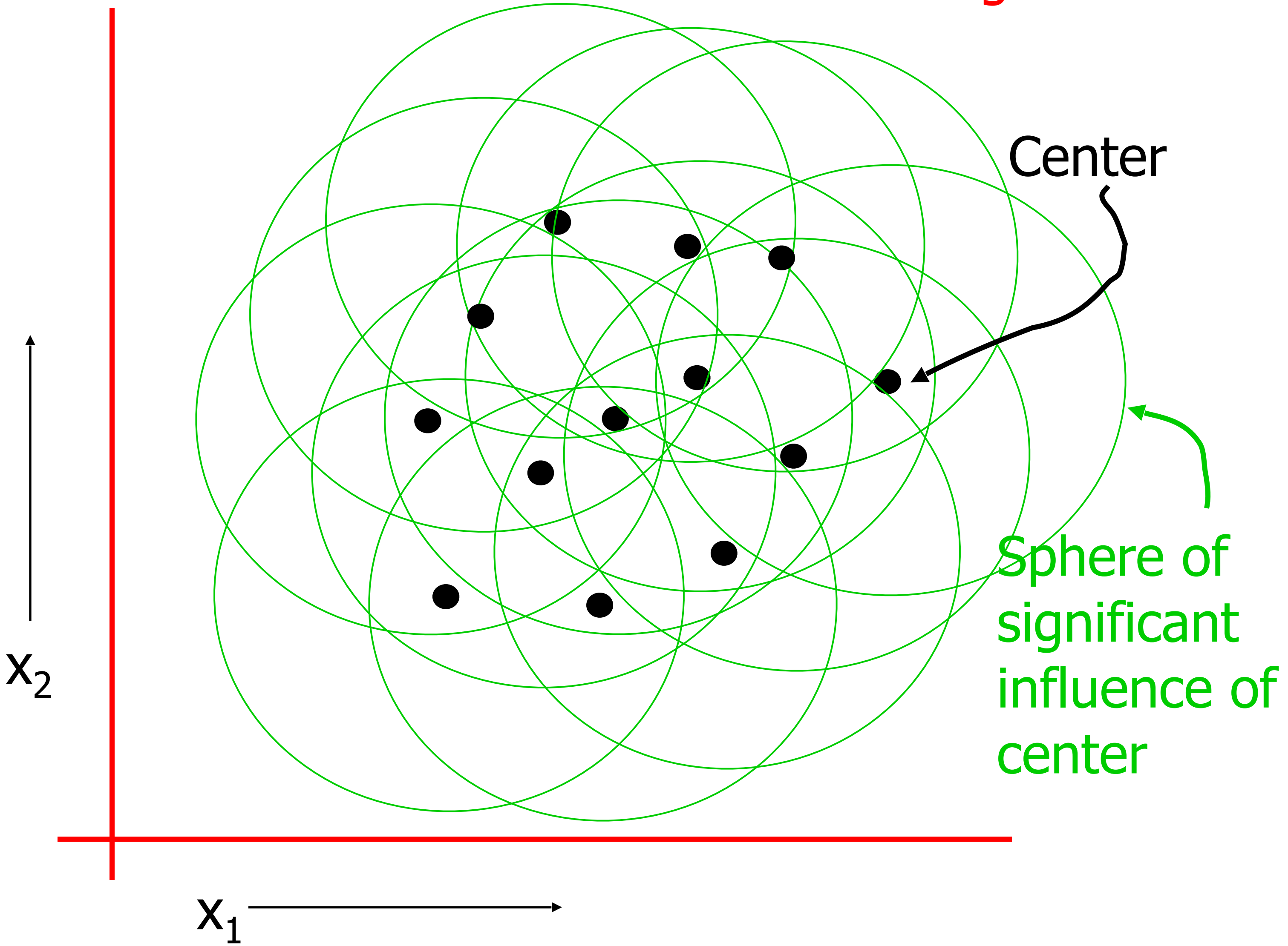
Hopeless!

Even before seeing the data, you should understand that this is a disaster!

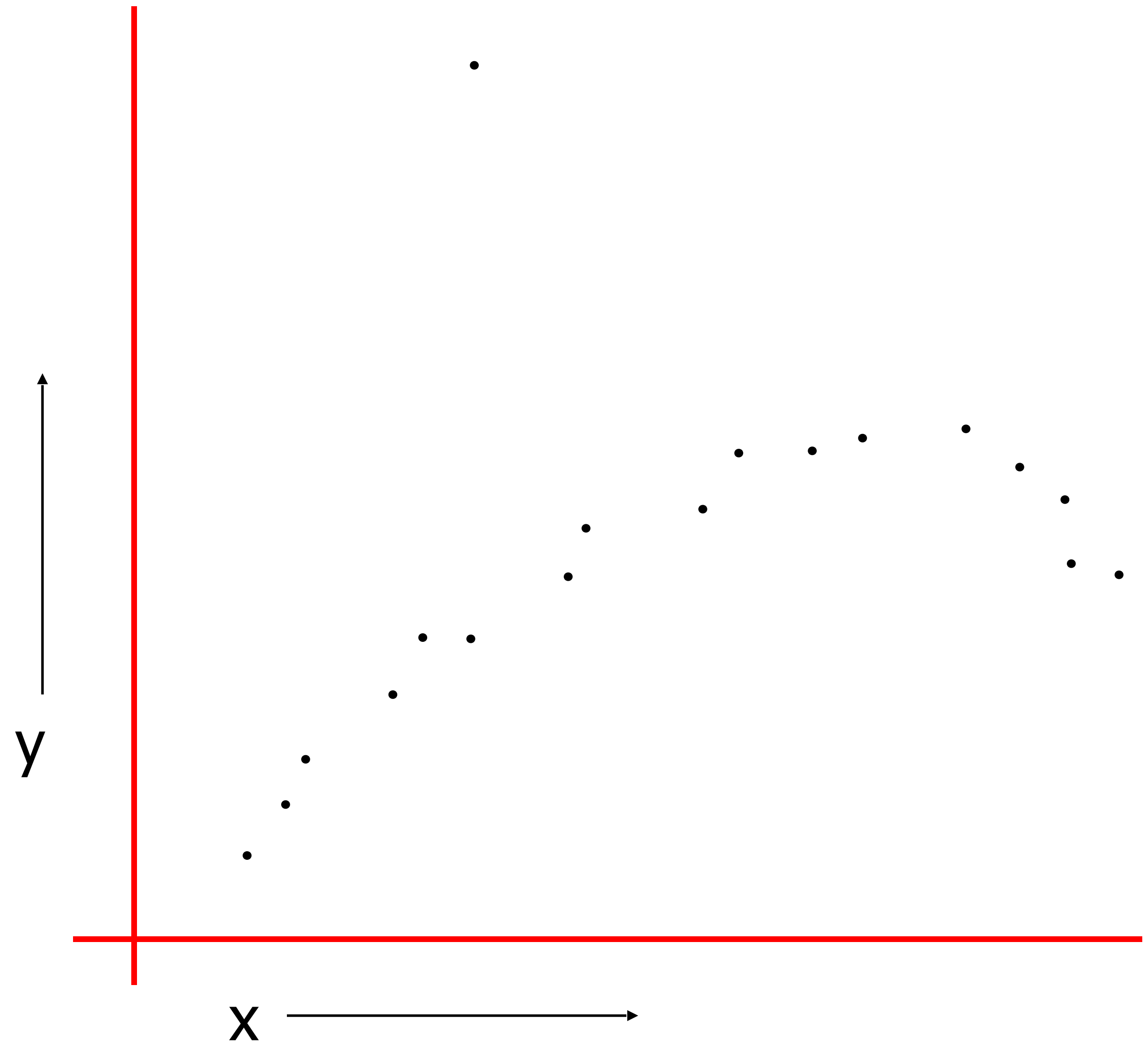


Unhappy

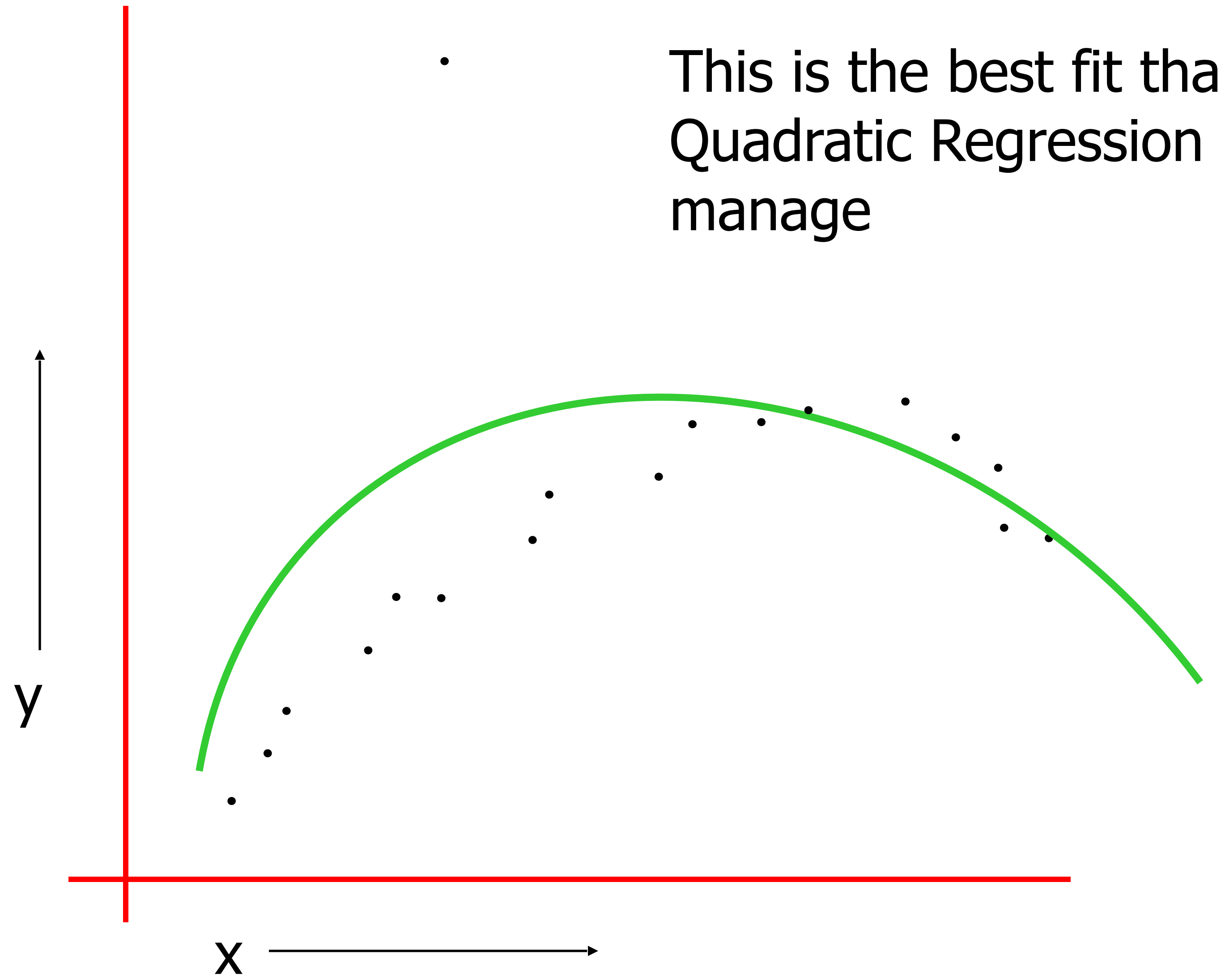
Even before seeing the data, you should understand that this isn't good either..



6: Robust Regression

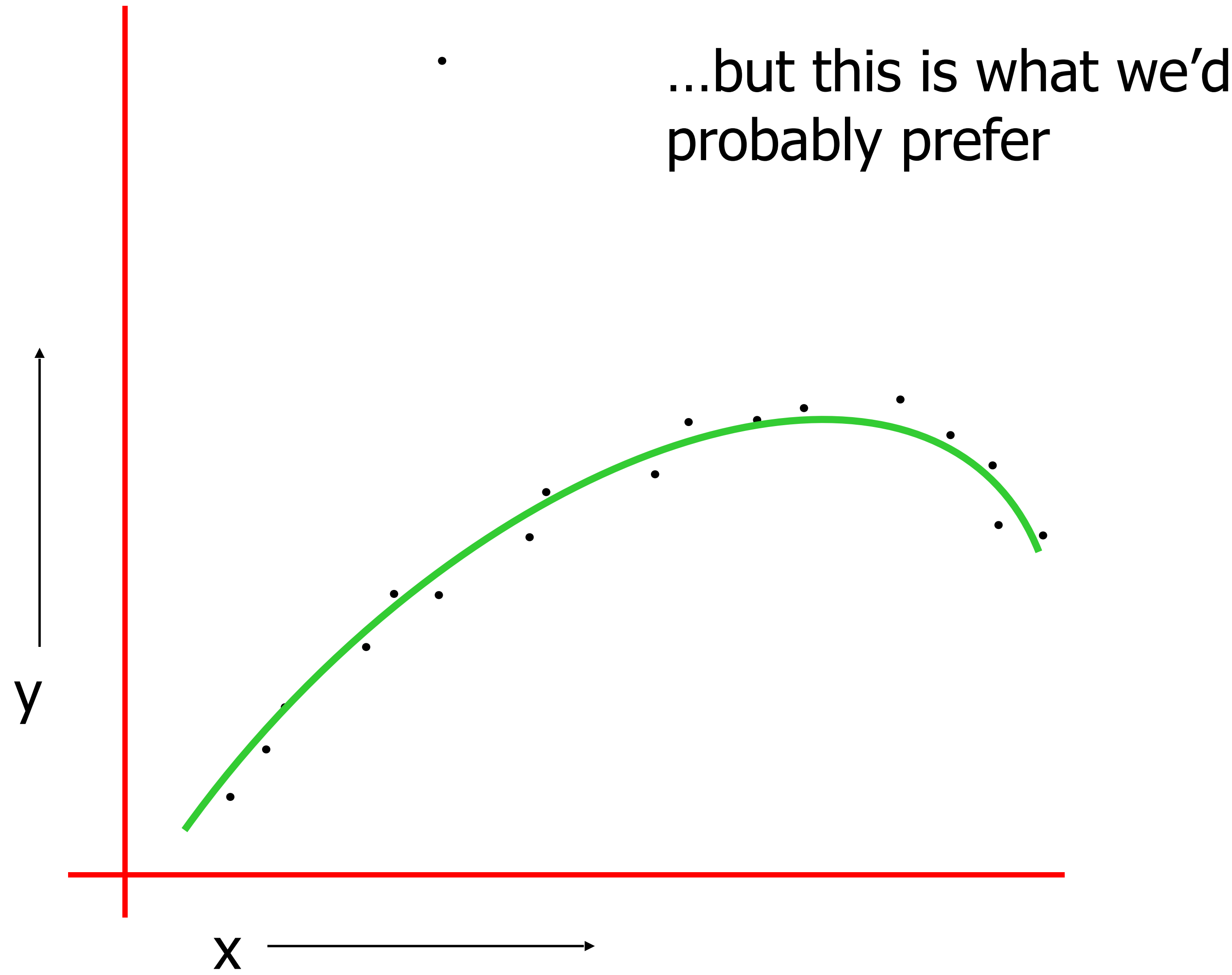


Robust Regression



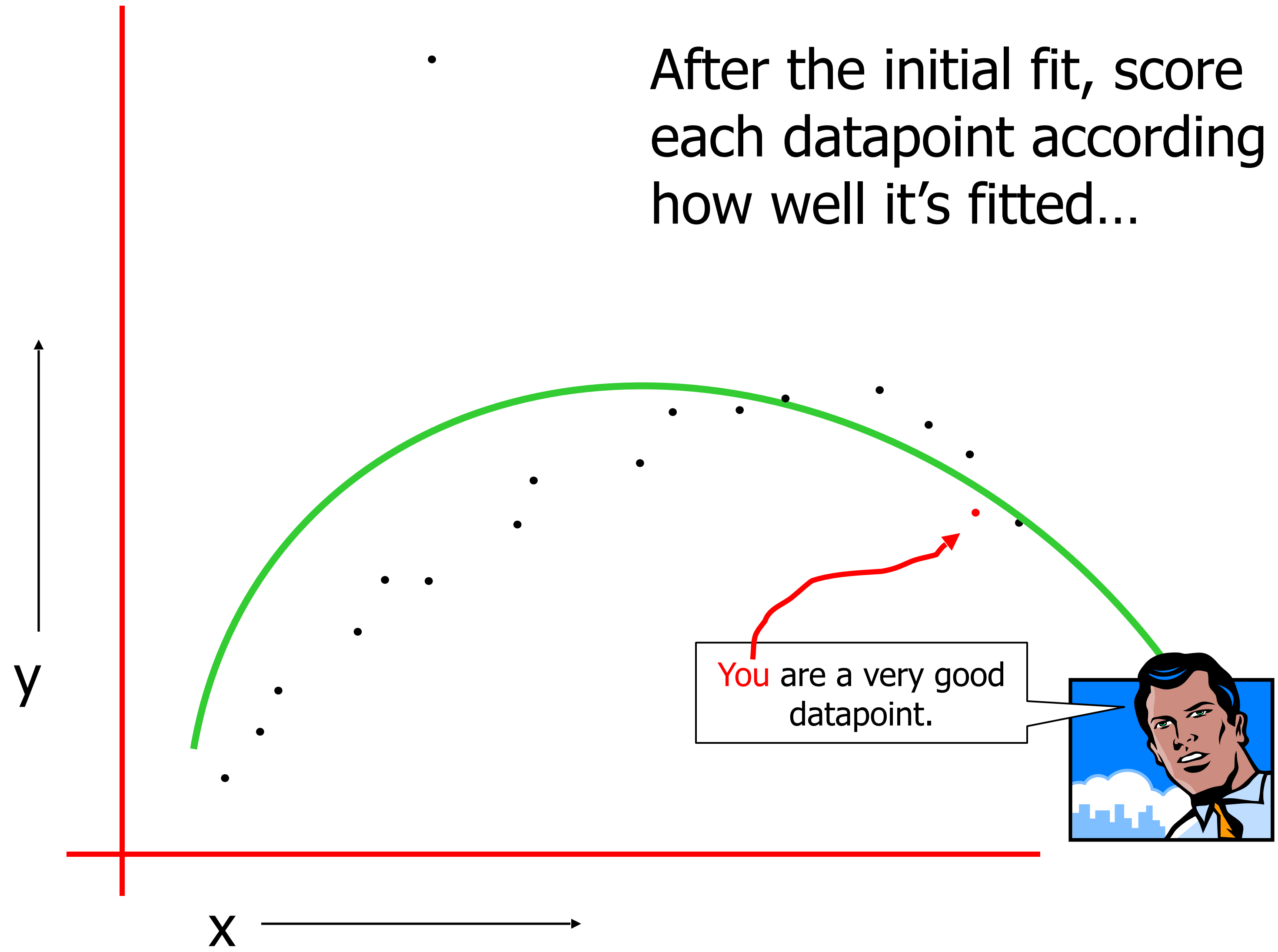
This is the best fit that Quadratic Regression can manage

Robust Regression



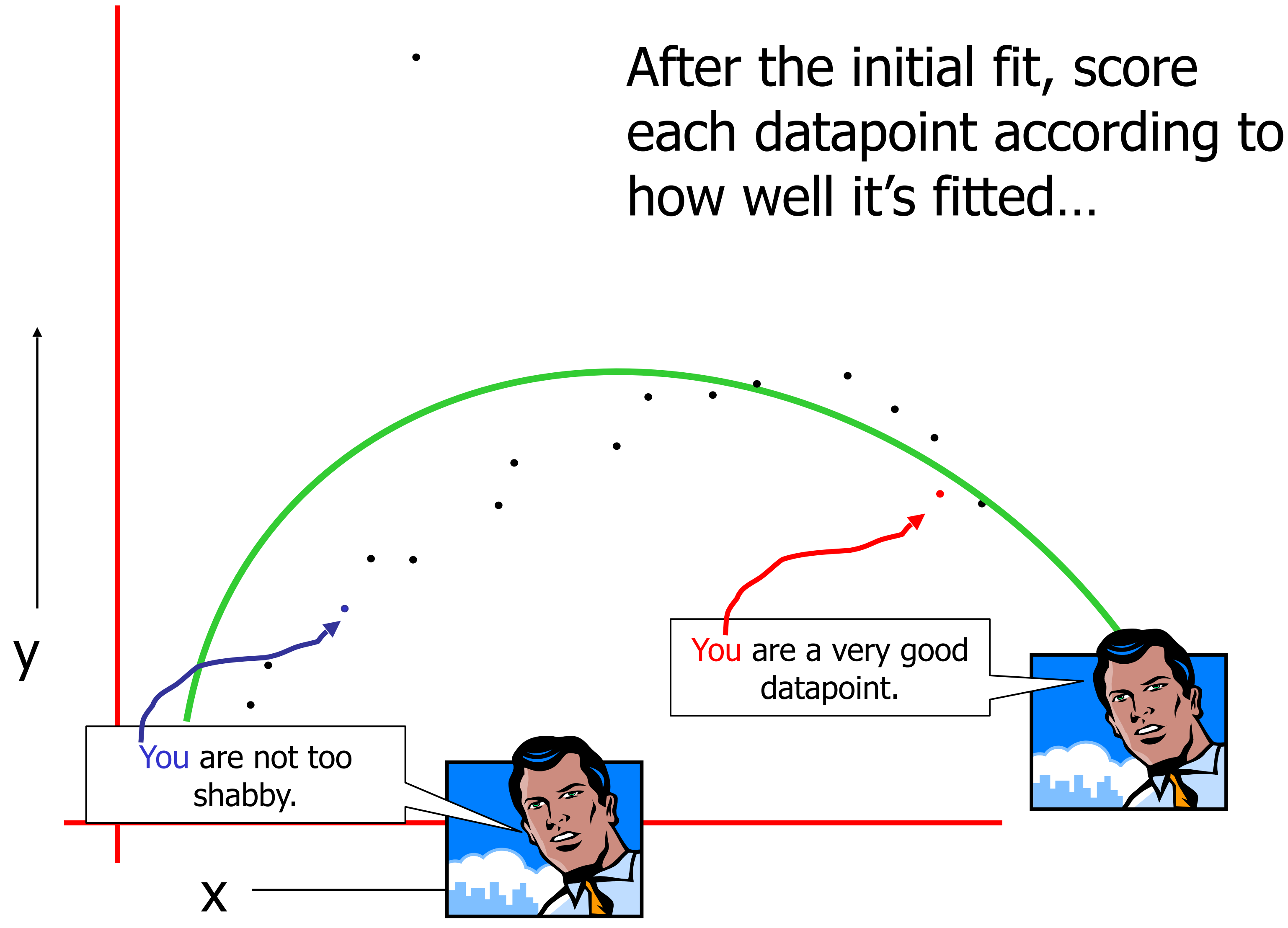
LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it's fitted…



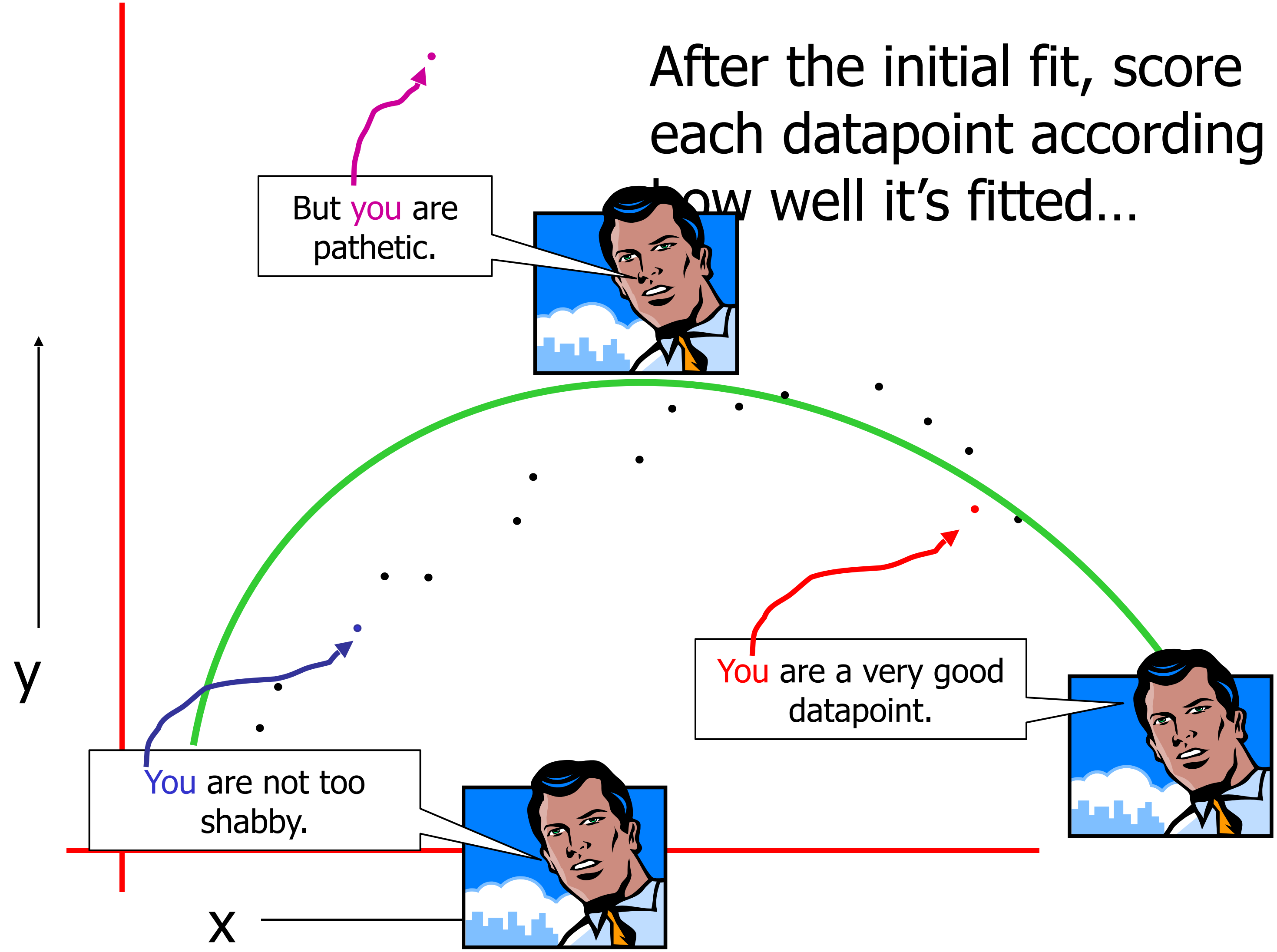
LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it's fitted…

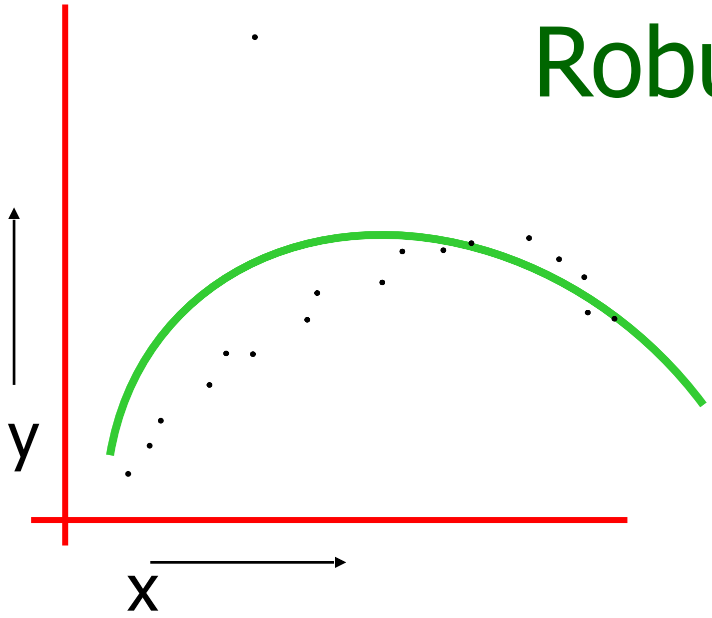


LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it's fitted…



Robust Regression

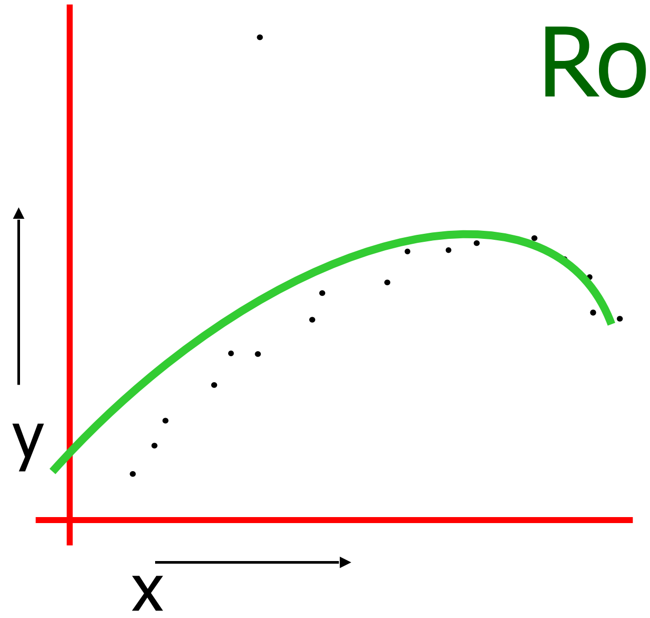


For k = 1 to R…

- Let (x_k, y_k) be the kth datapoint
- Let y^{est}_k be predicted value of y_k
- Let w_k be a weight for datapoint k that is large if the datapoint fits well and small if it fits badly:

$$w_k = \text{KernelFn}([y_k - y^{est}_k]^2)$$

Robust Regression



For k = 1 to R…

- Let (x_k, y_k) be the kth datapoint
- Let y^{est}_k be predicted value of y_k
- Let w_k be a weight for datapoint k that is large if the datapoint fits well and small if it fits badly:

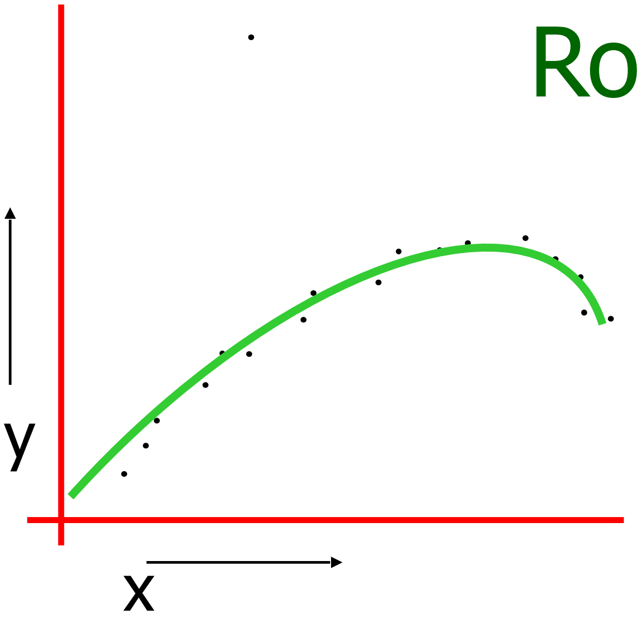
$$w_k = \text{KernelFn}([y_k - y^{est}_k]^2)$$

Then redo the regression using weighted datapoints.

I taught you how to do this in the "Instance-based" lecture (only then the weights depended on distance in input-space)

Guess what happens next?

Robust Regression



For k = 1 to R…

- Let (x_k, y_k) be the kth datapoint
- Let y^{est}_k be predicted value of y_k
- Let w_k be a weight for datapoint k that is large if the datapoint fits well and small if it fits badly:

$$w_k = \text{KernelFn}([y_k - y^{est}_k]^2)$$

Then redo the regression using weighted datapoints.

I taught you how to do this in the "Instance-based" lecture (only then the weights depended on distance in input-space)

Repeat whole thing until converged!

Robust Regression---what we're doing

What regular regression does:

Assume y_k was originally generated using the following recipe:

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + N(0,\sigma^2)$$

Computational task is to find the Maximum Likelihood β_0 , β_1 and β_2

Robust Regression---what we're doing

What LOESS robust regression does:

Assume y_k was originally generated using the following recipe:

With probability p:

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + N(0,\sigma^2)$$

But otherwise

$$y_k \sim N(\mu,\sigma_{huge}^2)$$

Computational task is to find the Maximum Likelihood β_0 , β_1 , β_2 , p, μ and σ_{huge}

Robust Regression---what we're doing

What LOESS robust regression does:

Assume y_k was originally generated using the following recipe:

With probability p:

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + N(0,\sigma^2)$$

But otherwise

$$y_k \sim N(\mu,\sigma_{huge}^2)$$

Computational task is to find the Maximum Likelihood β_0 , β_1 , β_2 , p, μ and σ_{huge}

Mysteriously, the reweighting procedure does this computation for us.

Your first glimpse of two spectacular letters:

E.M.

5: Regression Trees

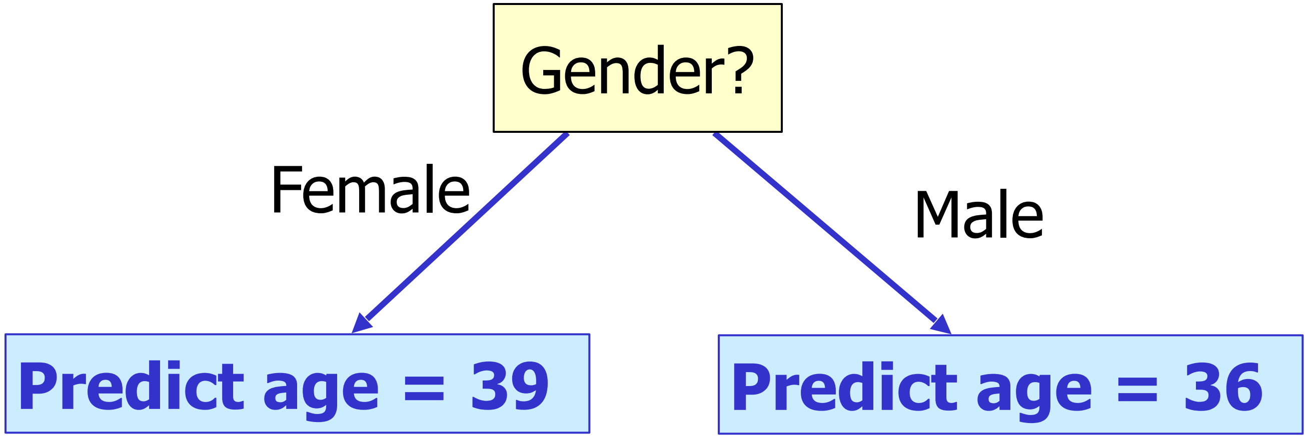
- "Decision trees for regression"

A regression tree leaf

Predict age = 47

Mean age of records matching this leaf node

A one-split regression tree



Choosing the attribute to split on

Gender	Rich?	Num. Children	Num. Beany Babies	Age
Female	No	2	1	38
Male	No	0	0	24
Male	Yes	0	5+	72
:	:	:	:	:

- We can't use information gain.
- What should we use?

Choosing the attribute to split on

Gender	Rich?	Num. Children	Num. Beany Babies	Age
Female	No	2	1	38
Male	No	0	0	24
Male	Yes	0	5+	72
:	:	:	:	:

MSE(Y|X) = The expected squared error if we must predict a record's Y value given only knowledge of the record's X value

If we're told x=j, the smallest expected error comes from predicting the mean of the Y-values among those records in which x=j. Call this mean quantity $\mu_y^{x=j}$

Then…

$$MSE(Y \mid X) = \frac{1}{R} \sum_{j=1}^{N_X} \sum_{(k \text{ such that } x_k = j)} (y_k - \mu_y^{x=j})^2$$

Choosing the attribute to split on

Gender	Rich?	Num. Children	Num. Beany Babies	Age
Female	N			
Male	N			
Male	Ye			
:	:			

Regression tree attribute selection: greedily choose the attribute that minimizes MSE(Y|X)

Guess what we do about real-valued inputs?

Guess how we prevent overfitting

MSE(Y|X) = The expected squared error if we must predict a record's Y value given only knowledge of the record's X value

If we're told x=j, the smallest expected error comes from predicting the mean of the Y-values among those records in which x=j. Call this mean quantity $\mu_y^{x=j}$

Then…

$$MSE(Y \mid X) = \frac{1}{R} \sum_{j=1}^{N_X} \sum_{(k \text{ such that } x_k = j)} (y_k - \mu_y^{x=j})^2$$

Pruning Decision

…property-owner = Yes

Gender?

Do I deserve to live?

Female

Male

Predict age = 39

Predict age = 36

\# property-owning females = 56712

Mean age among POFs = 39

Age std dev among POFs = 12

\# property-owning males = 55800

Mean age among POMs = 36

Age std dev among POMs = 11.5

Use a standard Chi-squared test of the null-hypothesis "these two populations have the same mean" and Bob's your uncle.

Linear Regression Trees

Also known as "Model Trees"

…property-owner = Yes

Gender?

Female

Predict age =

26 + 6 * NumChildren - 2 * YearsEducation

Male

Predict age =

24 + 7 * NumChildren - 2.5 * YearsEducation

Leaves contain linear functions (trained using linear regression on all records matching that leaf)

Split attribute chosen to minimize MSE of regressed children.

Pruning with a different Chi-squared

Linear Regression Trees

Also known as "M…l Trees"

…property-owner = Yes

Gender?

Female

Predict age =

26 + 6 * N…
* YearsE…

…en -
…cation

Detail: You typically ignore any categorical attribute that has been tested on higher up in the tree during the regression. But use all untested attributes, and use real-valued attributes even if they've been tested above

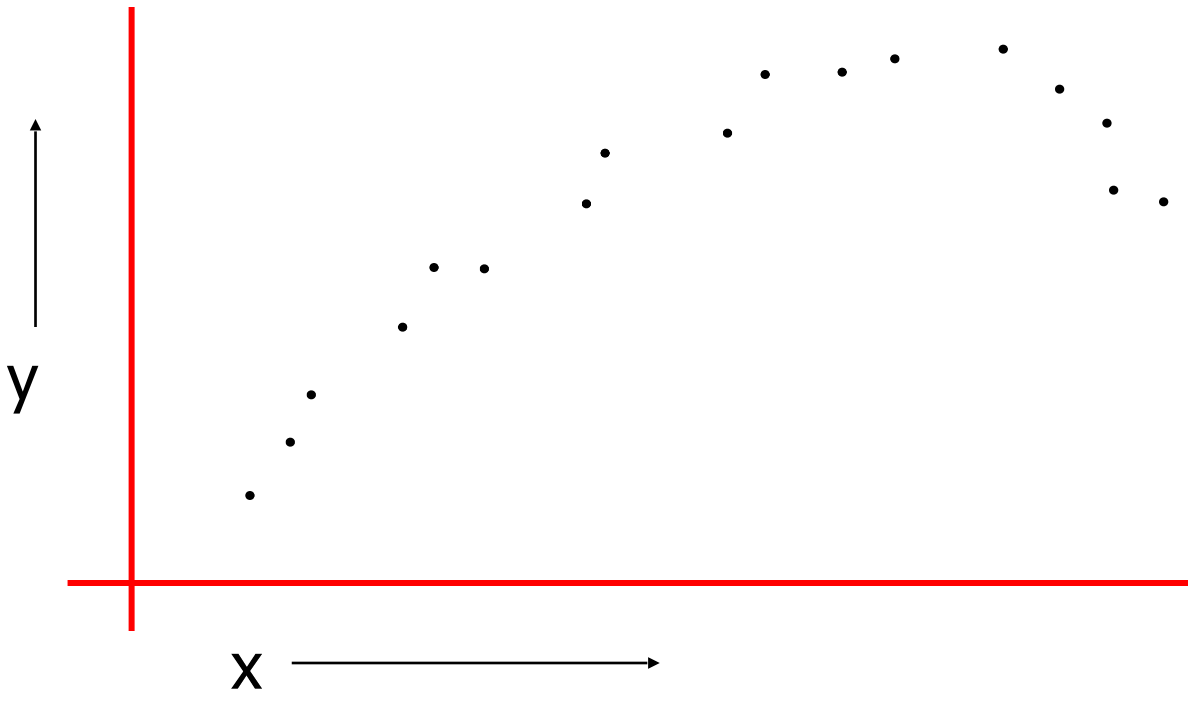
Leaves contain … functions (traine… linear regression … records matching t… leaf)

…tribute chosen to minimize …SE of regressed children.

Pruning with a different Chi-squared

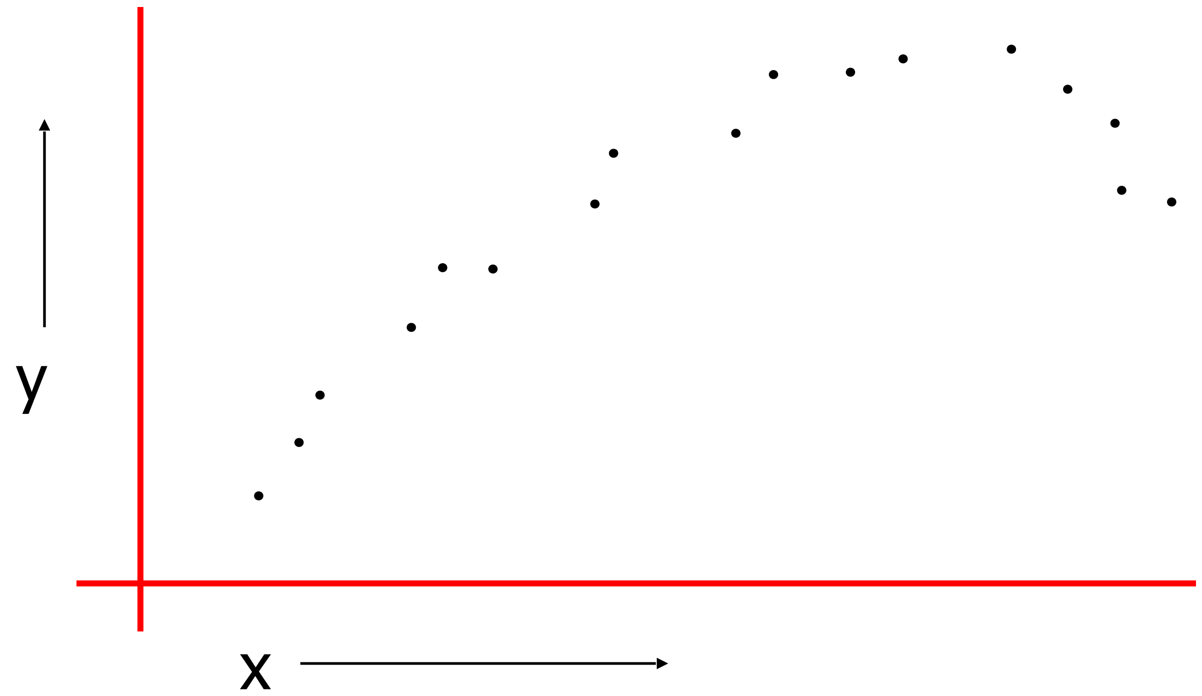
Test your understanding

Assuming regular regression trees, can you sketch a graph of the fitted function $y^{est}(x)$ over this diagram?



Test your understanding

Assuming linear regression trees, can you sketch a graph of the fitted function $y^{est}(x)$ over this diagram?



4: GMDH (c.f. BACON, AIM)

- Group Method Data Handling
- A very simple but very good idea:

1. Do linear regression
2. Use cross-validation to discover whether any quadratic term is good. If so, add it as a basis function and loop.
3. Use cross-validation to discover whether any of a set of familiar functions (log, exp, sin etc) applied to any previous basis function helps. If so, add it as a basis function and loop.
4. Else stop

GMDH (c.f. BACON, AIM)

- Group Method Data Handling
- A very simple but very good idea:

1. Do
2. Us y qu sis function and loop.
3. Use cross-validation to discover whether any of a set of familiar functions (log, exp, sin etc) applied to any previous basis function helps. If so, add it as a basis function and loop.
4. Else stop

Typical learned function:

age^{est} = height - 3.1 sqrt(weight) +

4.3 income / (cos (NumCars))

3: Cascade Correlation

- A super-fast form of Neural Network learning
- Begins with 0 hidden units
- Incrementally adds hidden units, one by one, doing ingeniously little recomputation between each unit addition

Cascade beginning

Begin with a regular dataset

Nonstandard notation:

- $X^{(i)}$ is the i'th attribute
- $x^{(i)}_k$ is the value of the i'th attribute in the k'th record

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2
:	:	:	:	:

Cascade first step

Begin with a regular dataset

Find weights $w^{(0)}_i$ to best fit Y.

I.E. to minimize

$$\sum_{k=1}^{R}(y_k - y_k^{(0)})^2 \text{ where } y_k^{(0)} = \sum_{j=1}^{m} w_j^{(0)} x_k^{(j)}$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2
:	:	:	:	:

Consider our errors…

Begin with a regular dataset

Find weights $w^{(0)}_i$ to best fit Y.

I.E. to minimize

$$\sum_{k=1}^{R} (y_k - y_k^{(0)})^2 \text{ where } y_k^{(0)} = \sum_{j=1}^{m} w_j^{(0)} x_k^{(j)}$$

$$\text{Define } e_k^{(0)} = y_k - y_k^{(0)}$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$
:	:	:	:	:	:	:

Create a hidden unit…

Find weights $u^{(0)}_i$ to define a new basis function $H^{(0)}(x)$ of the inputs.

Make it specialize in predicting the errors in our original fit:

Find $\{u^{(0)}_i\}$ to maximize correlation between $H^{(0)}(x)$ and $E^{(0)}$ where

$$H^{(0)}(\mathbf{x}) = g\left(\sum_{j=1}^{m} u_j^{(0)} x^{(j)}\right)$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$
:	:	:	:	:	:	:	:

Cascade next step

Find weights $w^{(1)}_i\ p^{(1)}_0$ to better fit Y.

I.E. to minimize

$$\sum_{k=1}^{R}(y_k - y_k^{(1)})^2 \text{ where } y_k^{(1)} = \sum_{j=1}^{m} w_j^{(0)} x_k^{(j)} + p_j^{(0)} h_k^{(0)}$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$
$x^{(0)}{}_1$	$x^{(1)}{}_1$	…	$x^{(m)}{}_1$	y_1	$y^{(0)}{}_1$	$e^{(0)}{}_1$	$h^{(0)}{}_1$	$y^{(1)}{}_1$
$x^{(0)}{}_2$	$x^{(1)}{}_2$	…	$x^{(m)}{}_2$	y_2	$y^{(0)}{}_2$	$e^{(0)}{}_2$	$h^{(0)}{}_2$	$y^{(1)}{}_2$
:	:	:	:	:	:	:	:	:

Now look at new errors

Find weights $w^{(1)}_i$ $p^{(1)}_0$ to better fit Y.

$$\text{Define } e_k^{(1)} = y_k - y_k^{(1)}$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$	$E^{(1)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$	$y^{(1)}_1$	$e^{(1)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$	$y^{(1)}_2$	$e^{(1)}_2$
:	:	:	:	:	:	:	:	:	:

Create next hidden unit…

Find weights $u^{(1)}_i$ $v^{(1)}_0$ to define a new basis function $H^{(1)}(x)$ of the inputs.

Make it specialize in predicting the errors in our original fit:

Find $\{u^{(1)}_i, v^{(1)}_0\}$ to maximize correlation between $H^{(1)}(x)$ and $E^{(1)}$ where

$$H^{(1)}(\mathbf{x}) = g\left(\sum_{j=1}^{m} u_j^{(1)} x^{(j)} + v_0^{(1)} h^{(0)} \right)$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$	$E^{(1)}$	$H^{(1)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$	$y^{(1)}_1$	$e^{(1)}_1$	$h^{(1)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$	$y^{(1)}_2$	$e^{(1)}_2$	$h^{(1)}_2$
:	:	:	:	:	:	:	:	:	:	:

Cascade n'th step

Find weights $w^{(n)}_i$ $p^{(n)}_j$ to better fit Y.

I.E. to minimize

$$\sum_{k=1}^{R}(y_k - y_k^{(n)})^2 \text{ where } y_k^{(n)} = \sum_{j=1}^{m} w_j^{(n)} x_k^{(j)} + \sum_{j=1}^{n-1} p_j^{(n)} h_k^{(j)}$$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$	$E^{(1)}$	$H^{(1)}$	…	$Y^{(n)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$	$y^{(1)}_1$	$e^{(1)}_1$	$h^{(1)}_1$	…	$y^{(n)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$	$y^{(1)}_2$	$e^{(1)}_2$	$h^{(1)}_2$	…	$y^{(n)}_2$
:	:	:	:	:	:	:	:	:	:	:	:	:

Now look at new errors

Find weights $w^{(n)}_i$ $p^{(n)}_j$ to better fit Y.

I.E. to minimize

Define $e_k^{(n)} = y_k - y_k^{(n)}$

$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$	$E^{(1)}$	$H^{(1)}$	…	$Y^{(n)}$	$E^{(n)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$	$y^{(1)}_1$	$e^{(1)}_1$	$h^{(1)}_1$	…	$y^{(n)}_1$	$e^{(n)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$	$y^{(1)}_2$	$e^{(1)}_2$	$h^{(1)}_2$	…	$y^{(n)}_2$	$e^{(n)}_2$
:	:	:	:	:	:	:	:	:	:	:	:	:	:

Create n'th hidden unit…

Find weights $u^{(n)}_i$ $v^{(n)}_i$ to define a new basis function $H^{(n)}(x)$ of the inputs.

Make it specialize in predicting the errors in our previous fit:

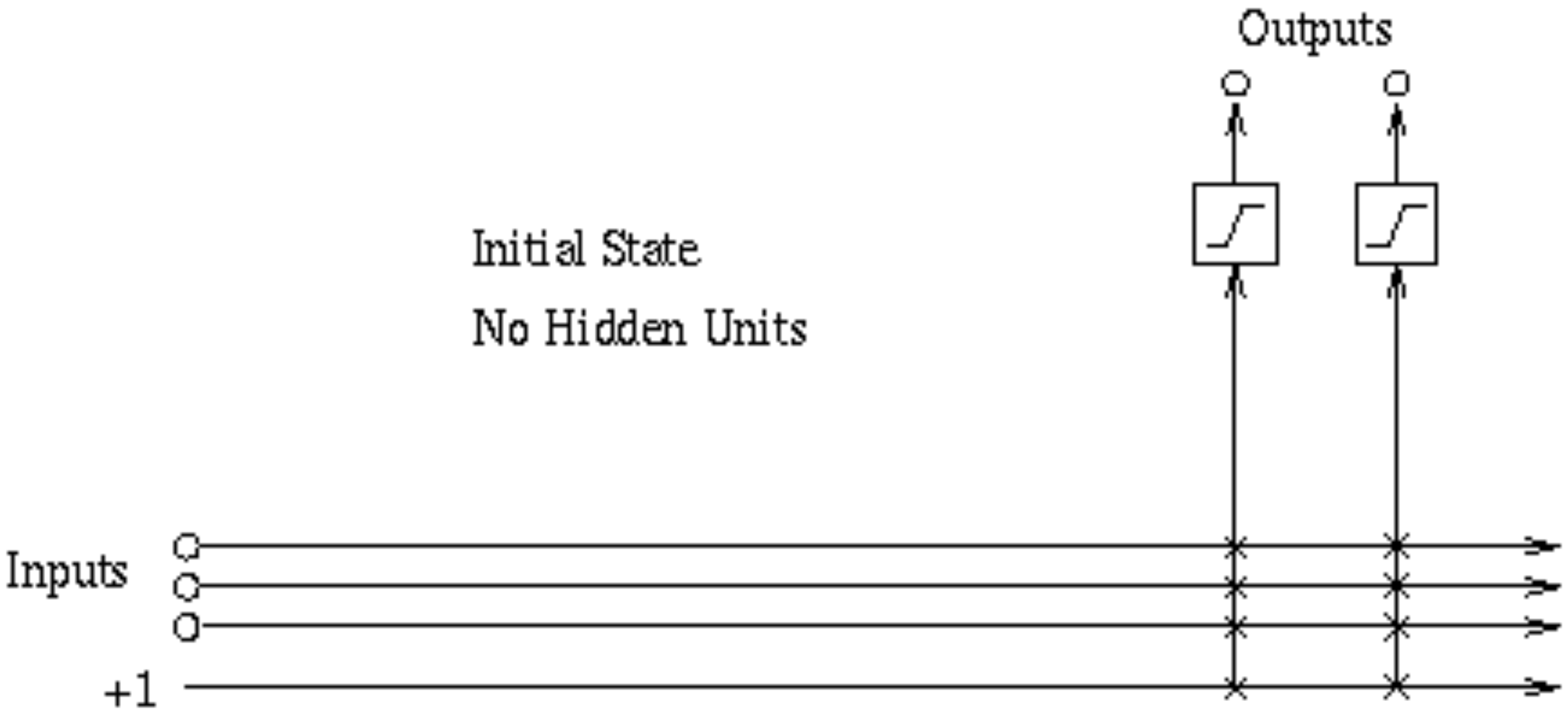
Find $\{u^{(n)}_i, v^{(n)}_j\}$ to maximize correlation between $H^{(n)}(x)$ and $E^{(n)}$ where

$$H^{(n)}(\mathbf{x}) = g\left(\sum_{j=1}^{m} u_j^{(n)} x^{(j)} + \sum_{j=1}^{n-1} v_j^{(n)} h^{(j)}\right)$$

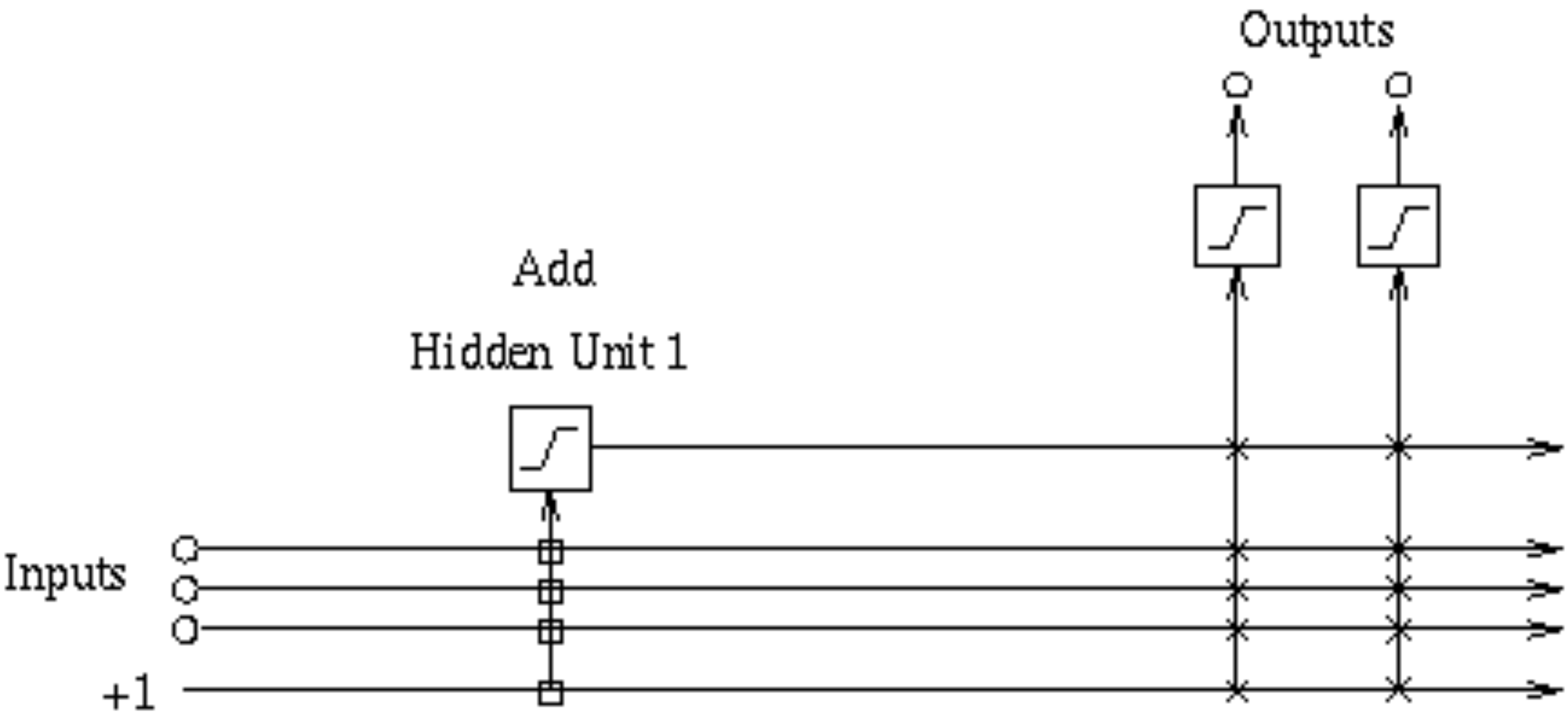
$X^{(0)}$	$X^{(1)}$	…	$X^{(m)}$	Y	$Y^{(0)}$	$E^{(0)}$	$H^{(0)}$	$Y^{(1)}$	$E^{(1)}$	$H^{(1)}$	…	$Y^{(n)}$	$E^{(n)}$	$H^{(n)}$
$x^{(0)}_1$	$x^{(1)}_1$	…	$x^{(m)}_1$	y_1	$y^{(0)}_1$	$e^{(0)}_1$	$h^{(0)}_1$	$y^{(1)}_1$	$e^{(1)}_1$	$h^{(1)}_1$	…	$y^{(n)}_1$	$e^{(n)}_1$	$h^{(n)}_1$
$x^{(0)}_2$	$x^{(1)}_2$	…	$x^{(m)}_2$	y_2	$y^{(0)}_2$	$e^{(0)}_2$	$h^{(0)}_2$	$y^{(1)}_2$	$e^{(1)}_2$	$h^{(1)}_2$	…	$y^{(n)}_2$	$e^{(n)}_2$	$h^{(n)}_2$
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:

Continue until satisfied with fit…

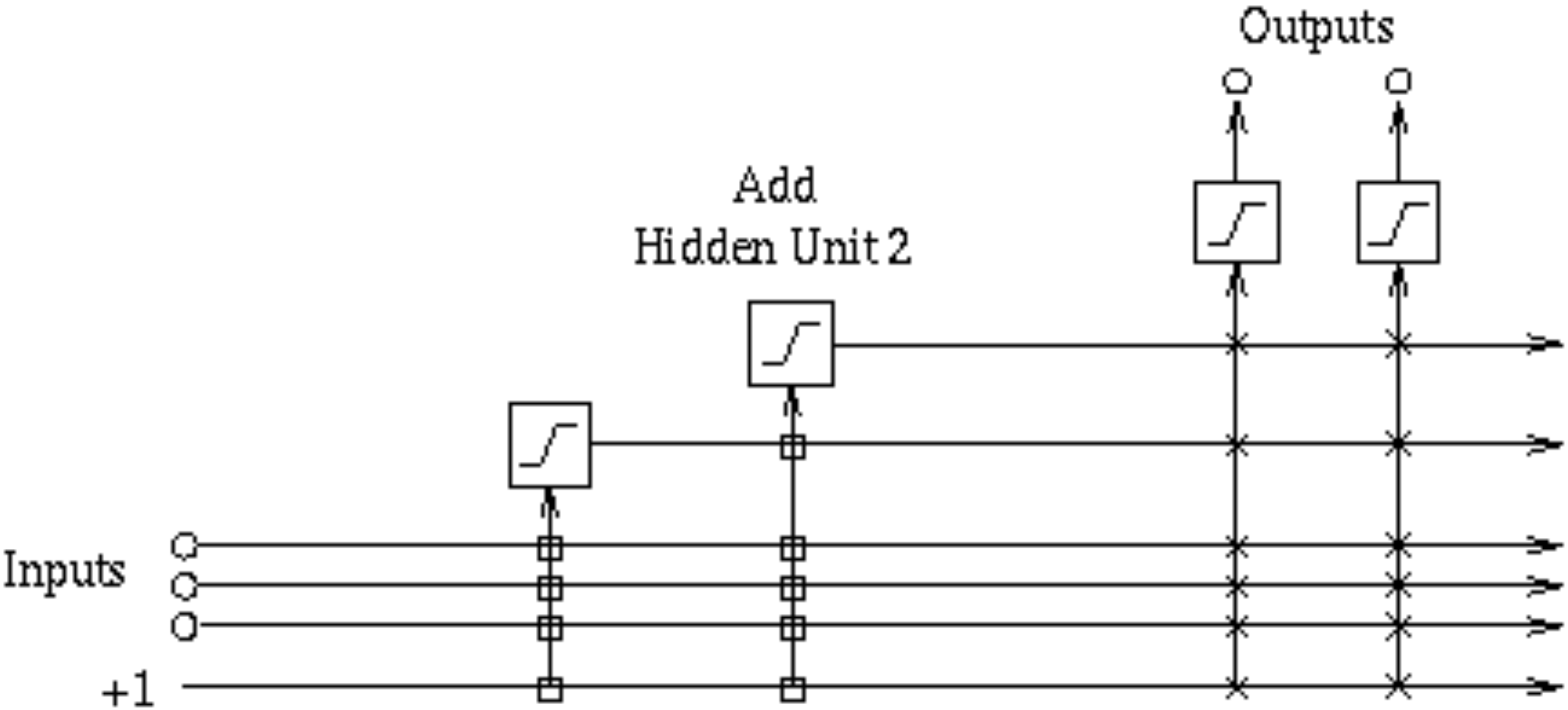
Visualizing first iteration



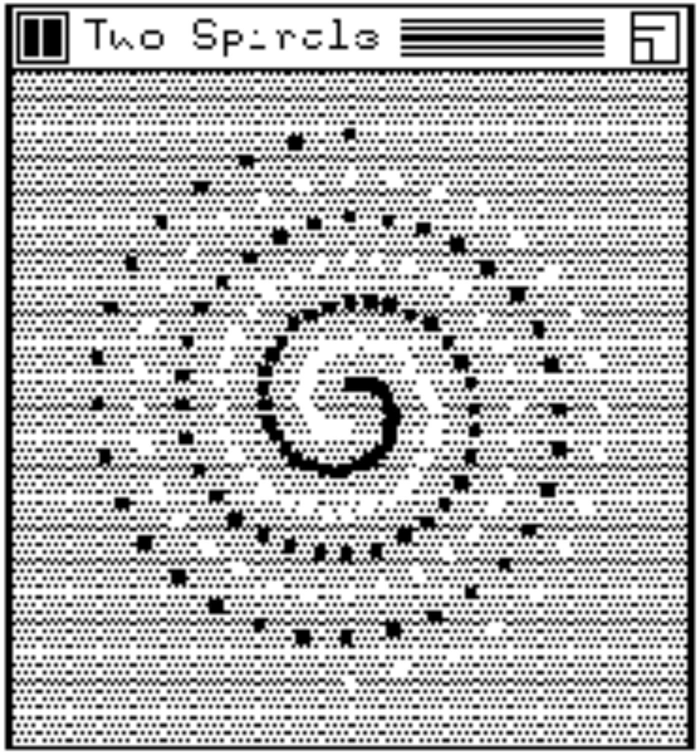
Visualizing second iteration



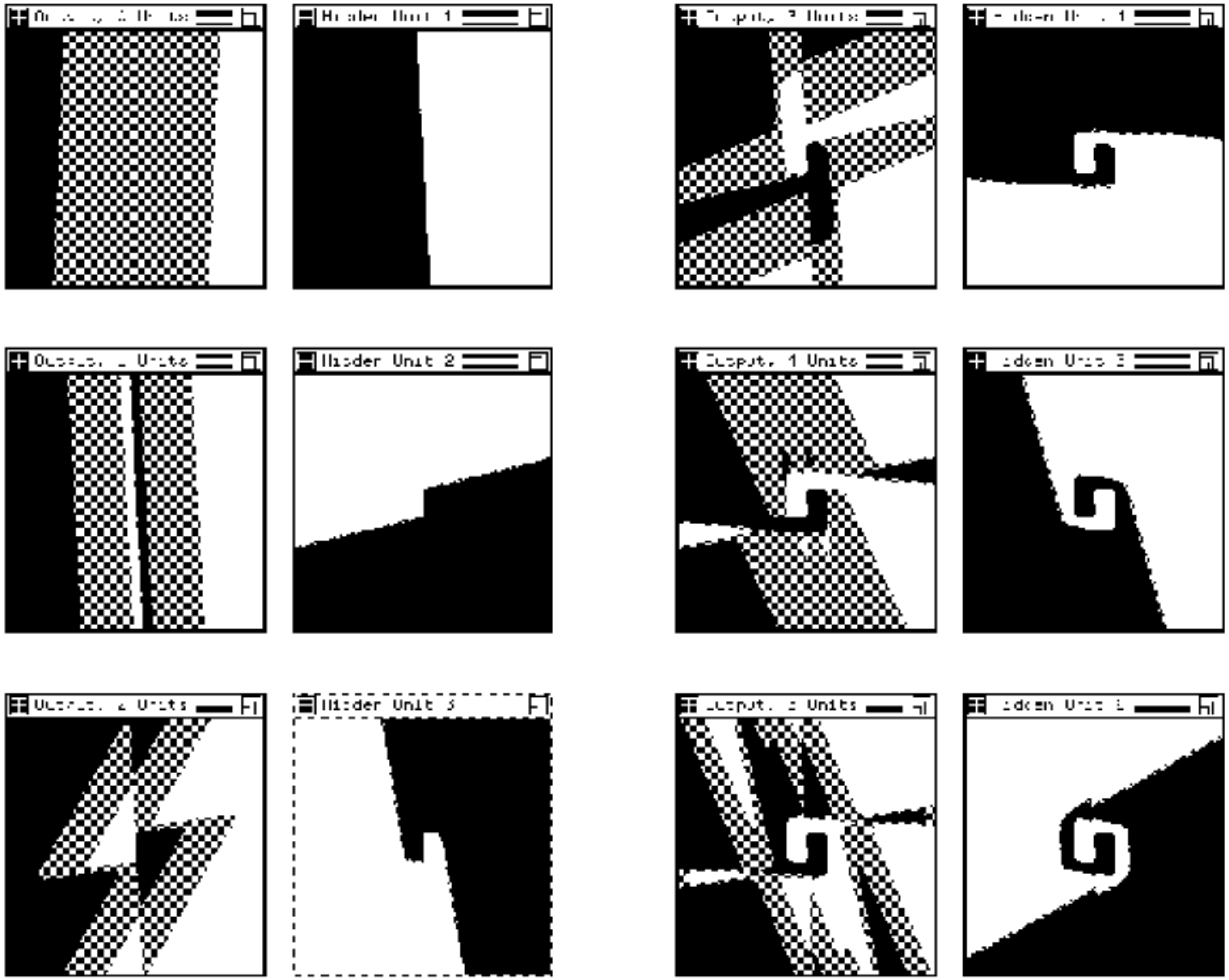
Visualizing third iteration



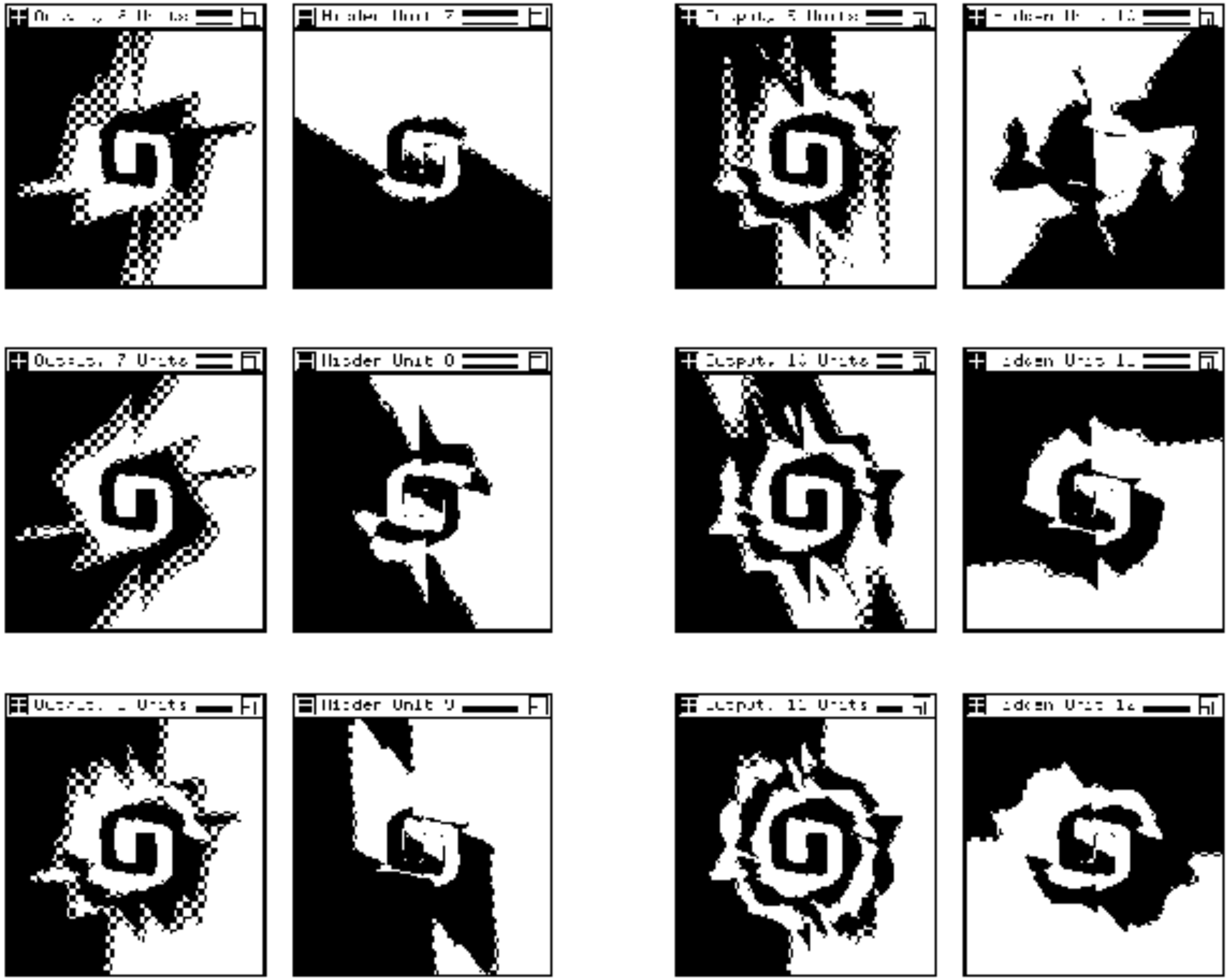
Example: Cascade Correlation for Classification



Training two spirals: Steps 1-6



Training two spirals: Steps 2-12



If you like Cascade Correlation…

See Also

- Projection Pursuit

  In which you add together many non-linear non-parametric scalar functions of carefully chosen directions

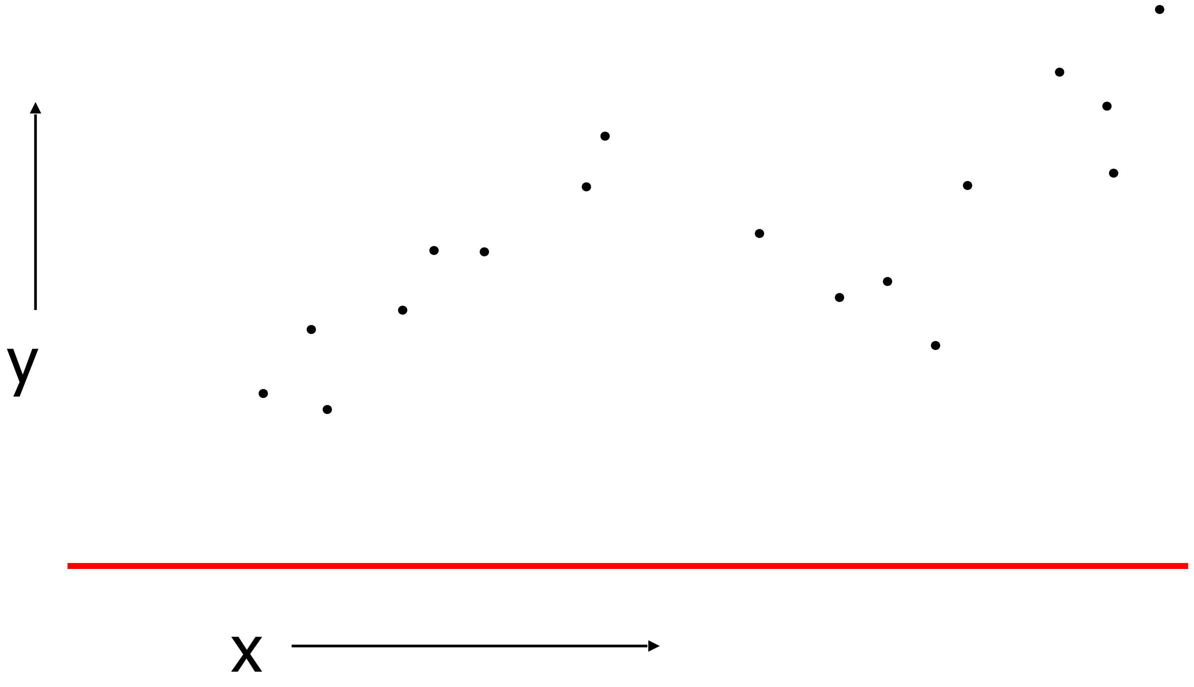
  Each direction is chosen to maximize error-reduction from the best scalar function

- ADA-Boost

  An additive combination of regression trees in which the n+1'th tree learns the error of the n'th tree

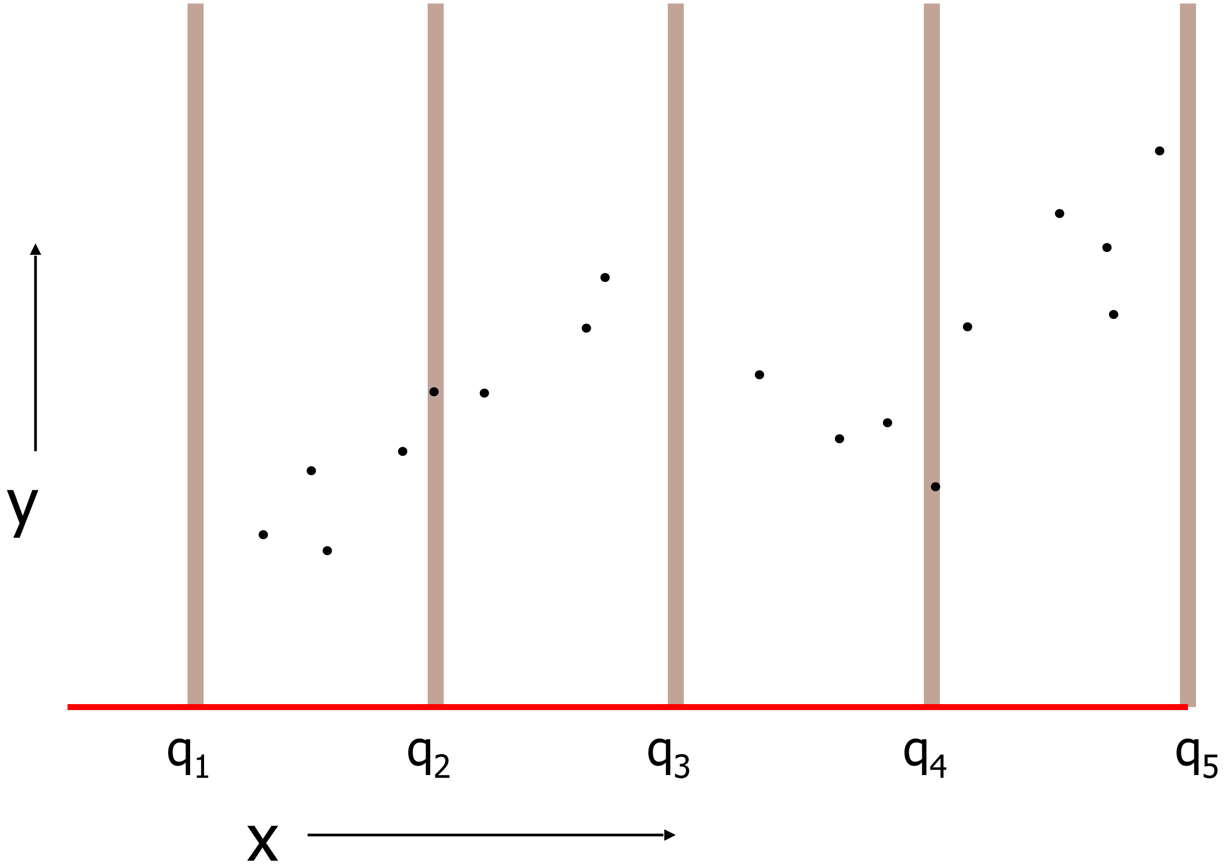
2: Multilinear Interpolation

Consider this dataset. Suppose we wanted to create a continuous and piecewise linear fit to the data



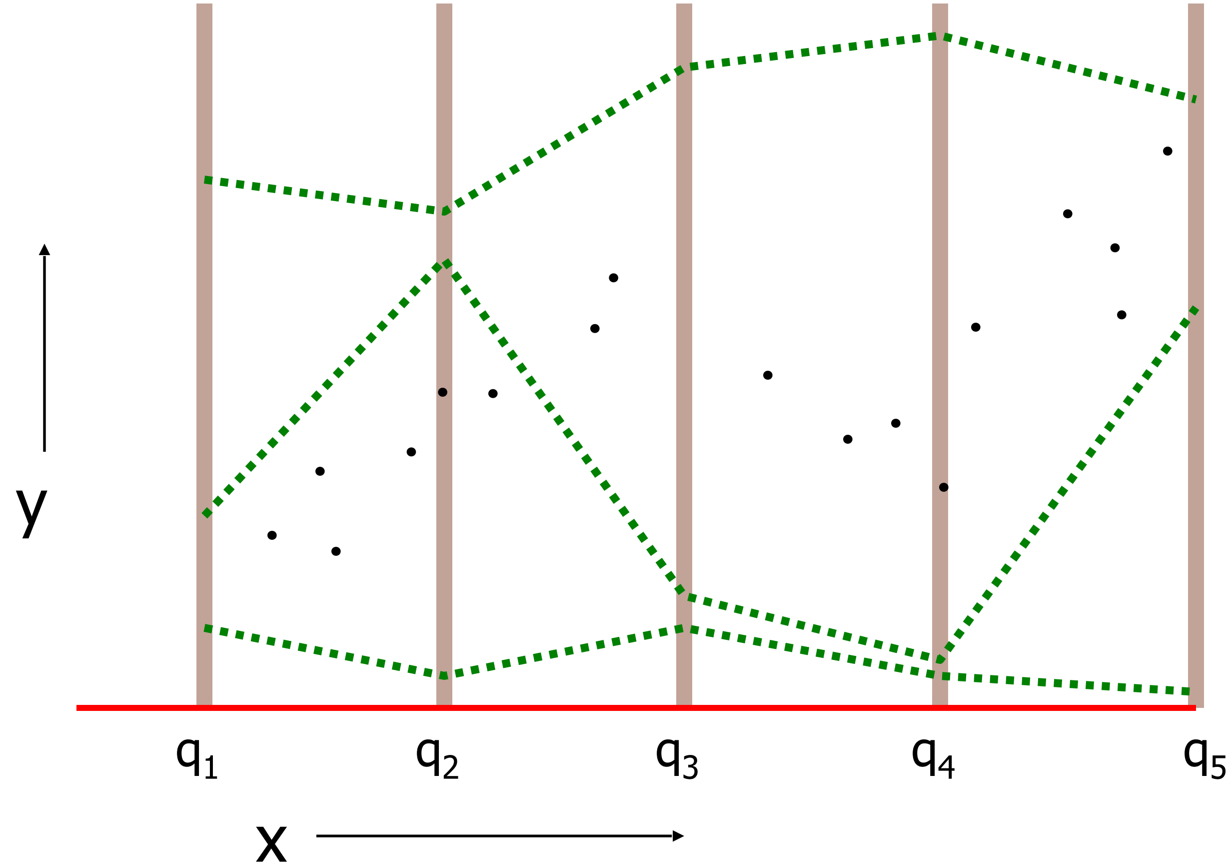
Multilinear Interpolation

Create a set of knot points: selected X-coordinates (usually equally spaced) that cover the data



Multilinear Interpolation

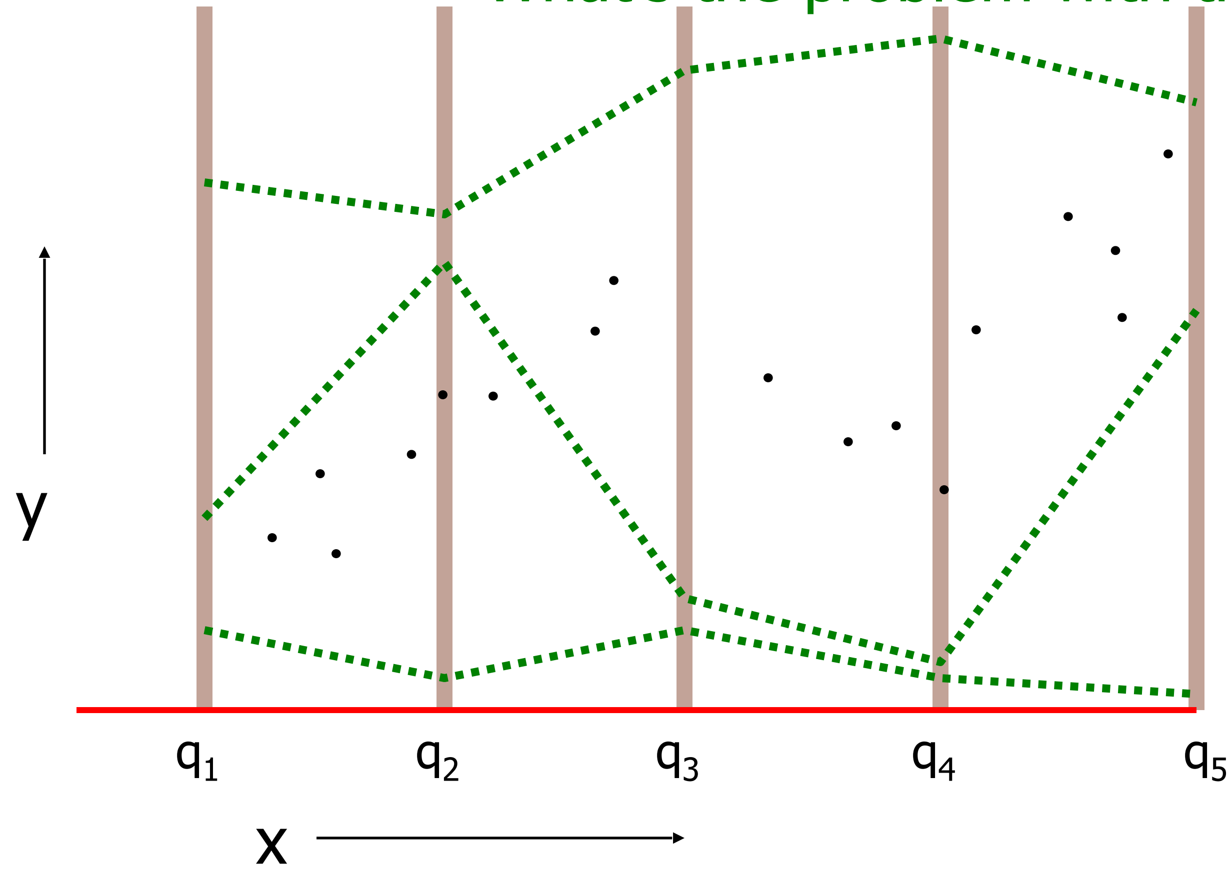
We are going to assume the data was generated by a noisy version of a function that can only bend at the knots. Here are 3 examples (none fits the data well)



How to find the best fit?

Idea 1: Simply perform a separate regression in each segment for each part of the curve

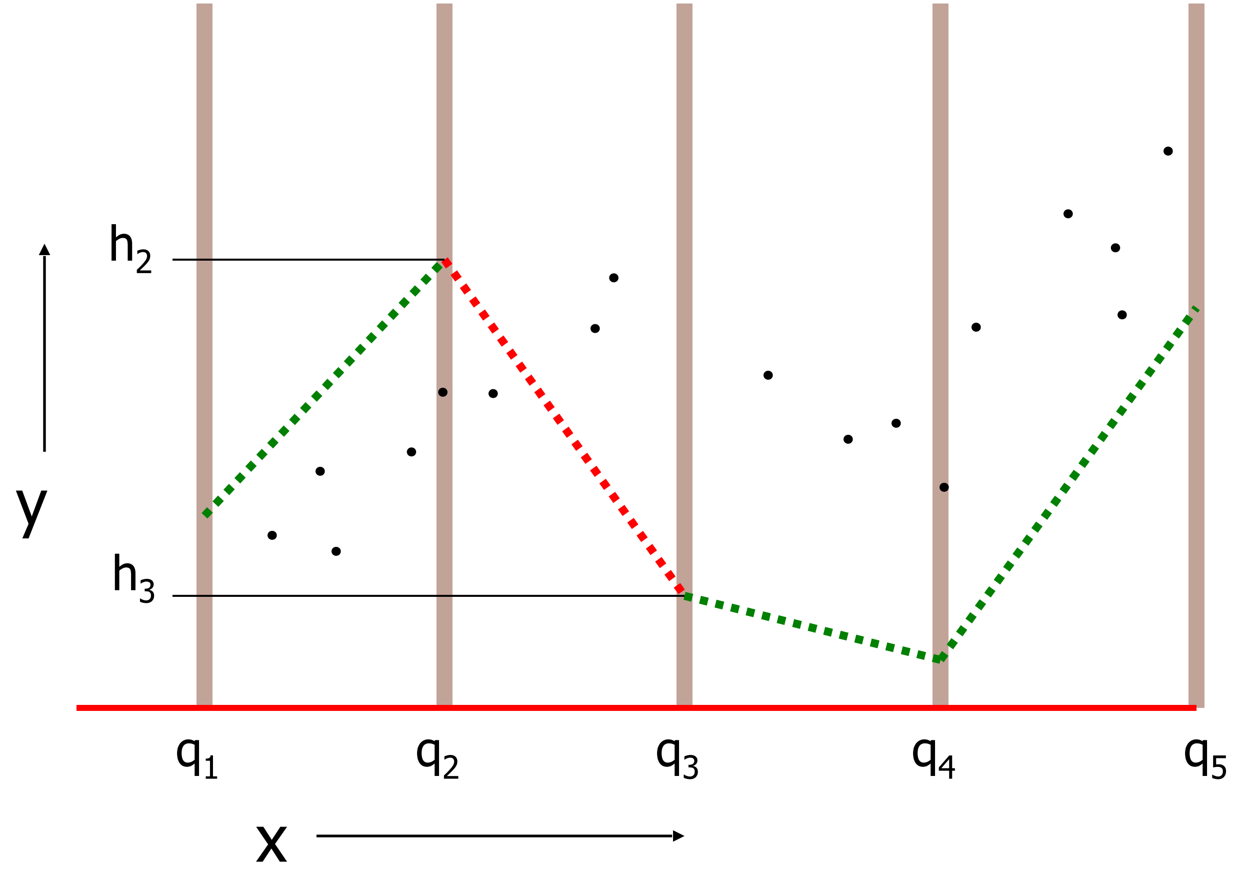
What's the problem with this idea?



How to find the best fit?

Let's look at what goes on in the red segment

$$y^{est}(x) = \frac{(q_3 - x)}{w} h_2 + \frac{(q_2 - x)}{w} h_3 \text{ where } w = q_3 - q_2$$

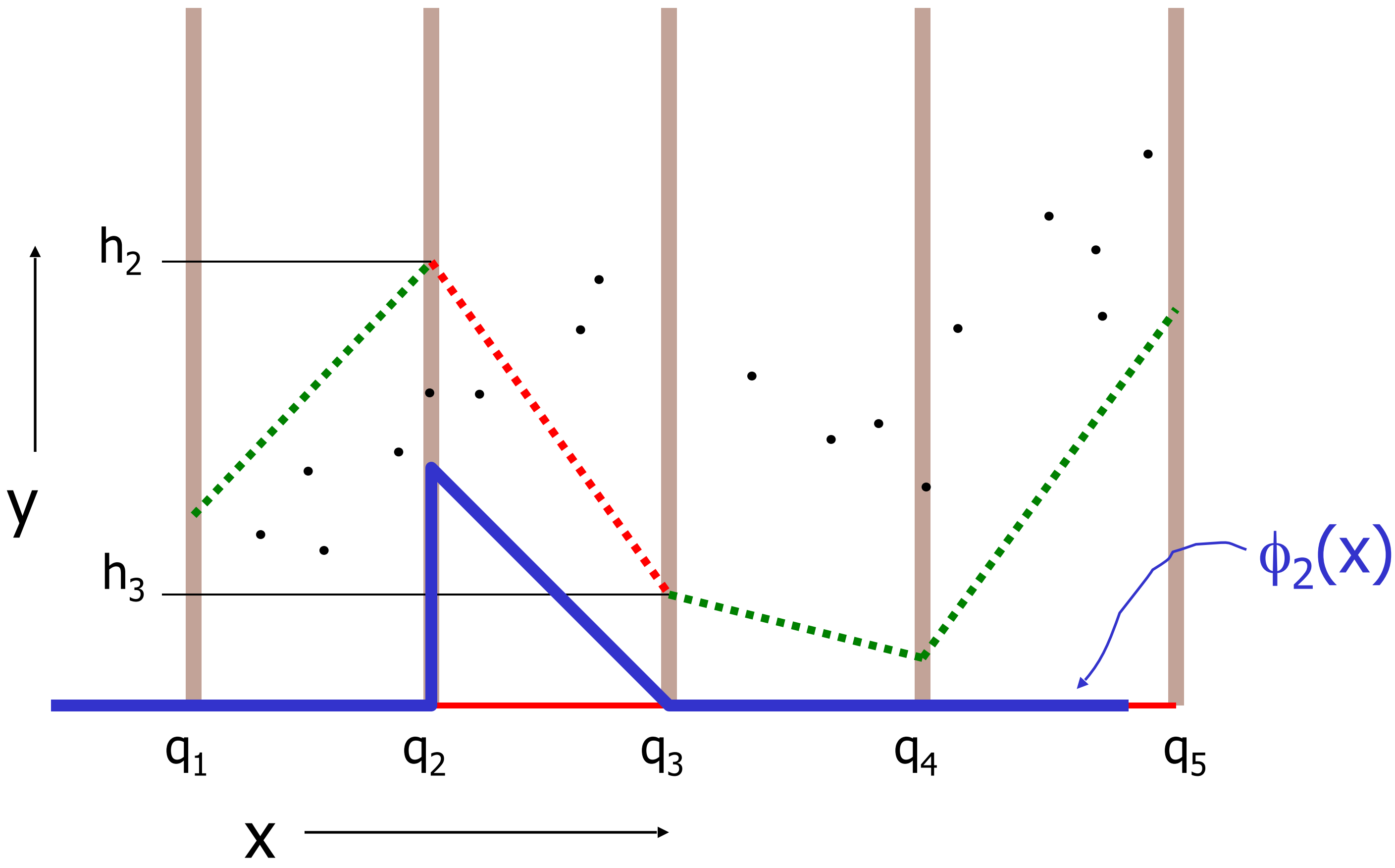


How to find the best fit?

In the red segment…

$$y^{est}(x) = h_2\varphi_2(x) + h_3\varphi_3(x)$$

$$\text{where } \varphi_2(x) = 1 - \frac{x - q_2}{w}, \varphi_3(x) = 1 - \frac{q_3 - x}{w}$$

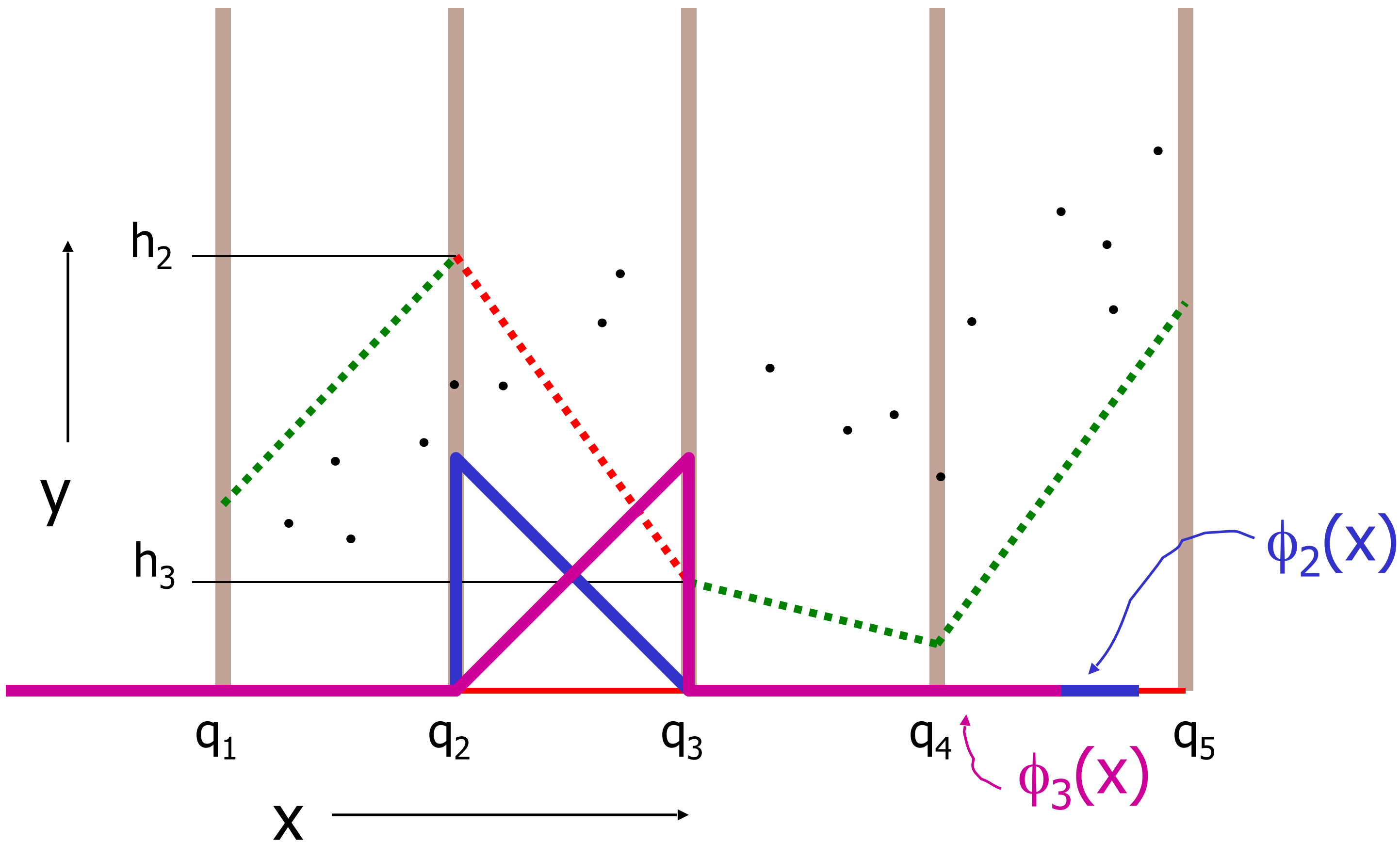


How to find the best fit?

In the red segment…

$$y^{est}(x) = h_2\varphi_2(x) + h_3\varphi_3(x)$$

$$\text{where } \varphi_2(x) = 1 - \frac{x - q_2}{w}, \varphi_3(x) = 1 - \frac{q_3 - x}{w}$$

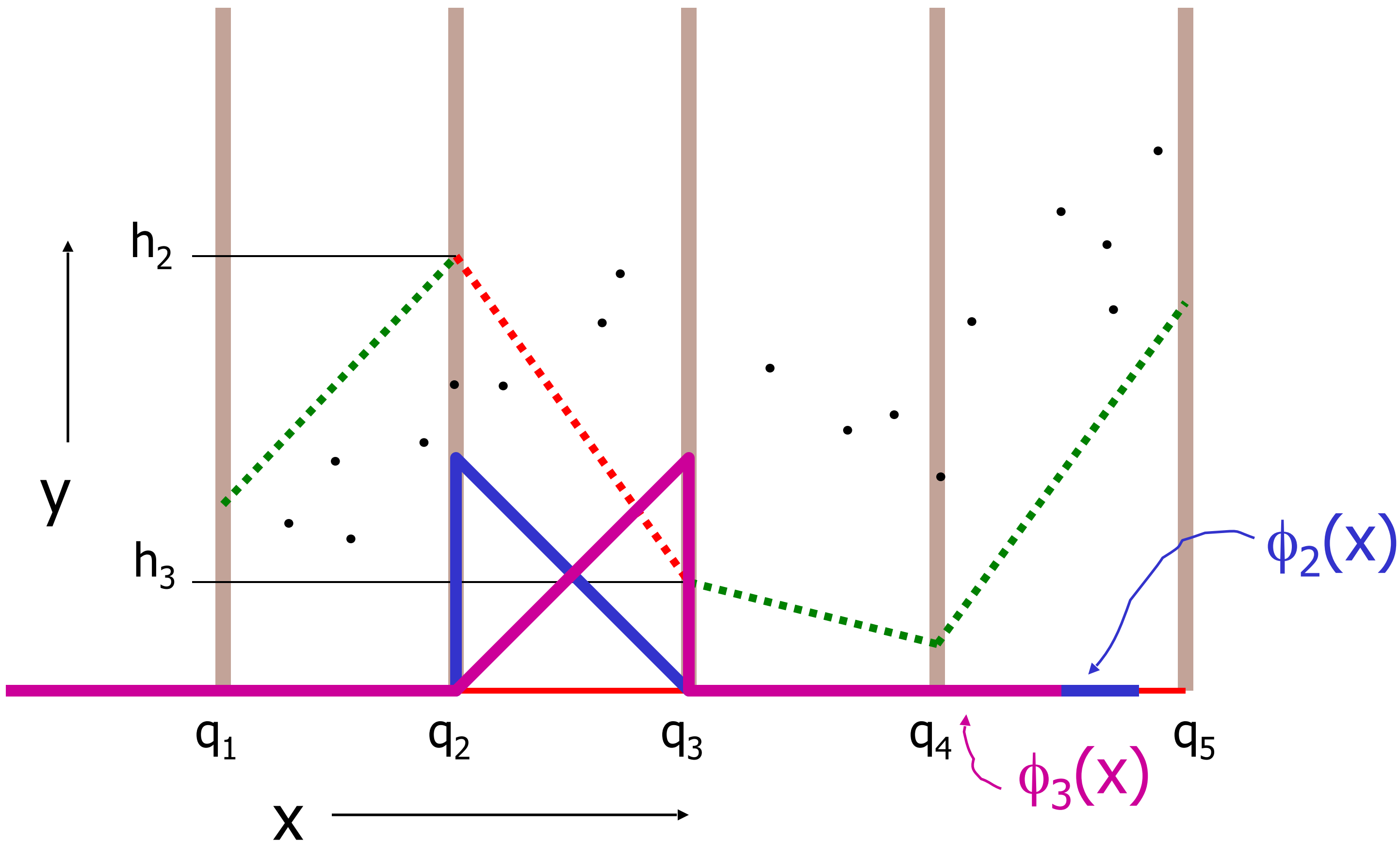


How to find the best fit?

In the red segment…

$$y^{est}(x) = h_2\varphi_2(x) + h_3\varphi_3(x)$$

$$\text{where } \varphi_2(x) = 1 - \frac{|x - q_2|}{w}, \varphi_3(x) = 1 - \frac{|x - q_3|}{w}$$

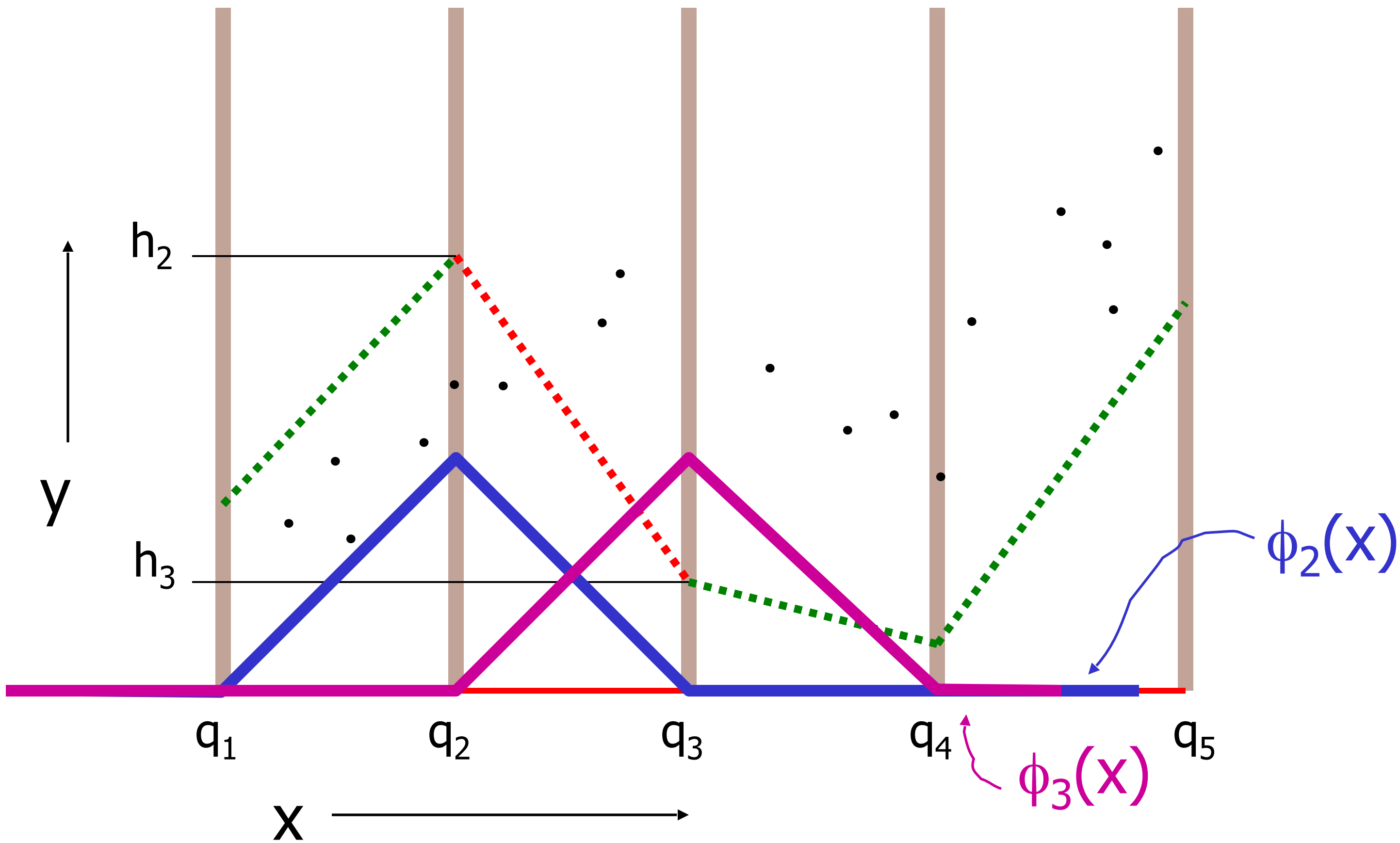


How to find the best fit?

In the red segment…

$$y^{est}(x) = h_2\varphi_2(x) + h_3\varphi_3(x)$$

$$\text{where } \varphi_2(x) = 1 - \frac{|x - q_2|}{w},\ \varphi_3(x) = 1 - \frac{|x - q_3|}{w}$$

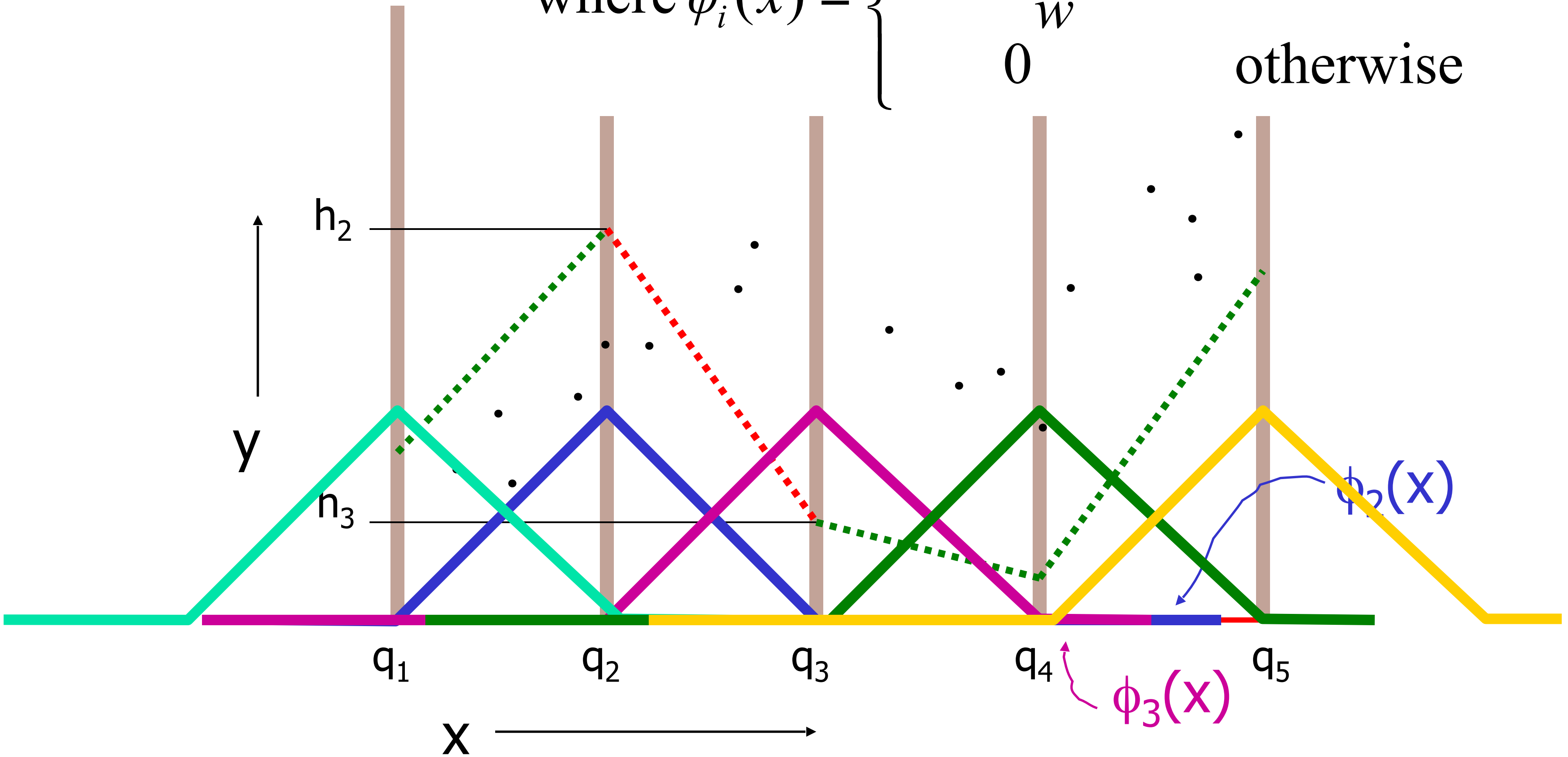


How to find the best fit?

In **general**

$$y^{est}(x) = \sum_{i=1}^{N_K} h_i \varphi_i(x)$$

$$\text{where } \varphi_i(x) = \begin{cases} 1 - \dfrac{|x - q_i|}{w} & \text{if } |x - q_i| < w \\ 0 & \text{otherwise} \end{cases}$$

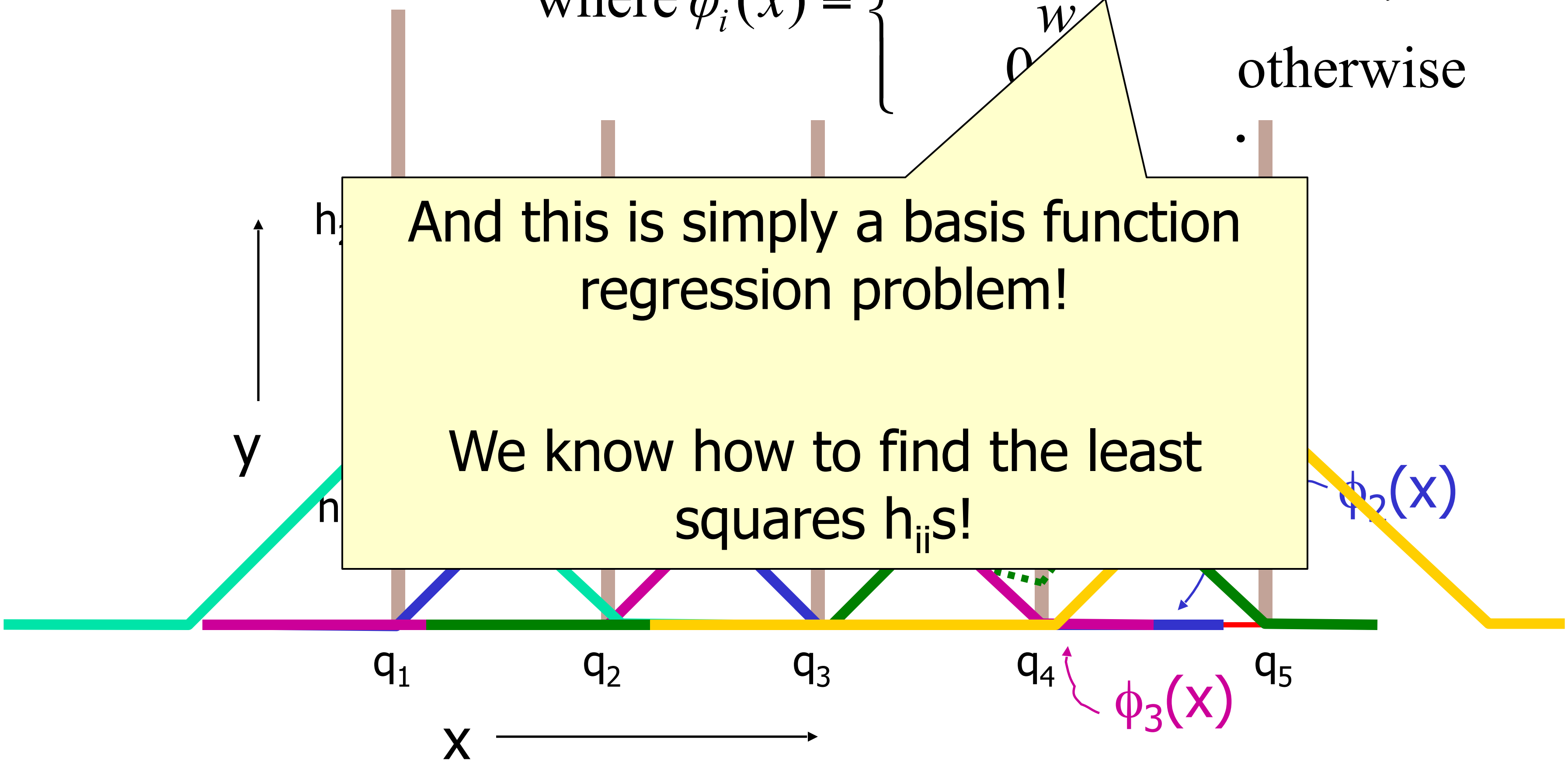


How to find the best fit?

In **general**

$$y^{est}(x) = \sum_{i=1}^{N_K} h_i \varphi_i(x)$$

$$\text{where } \varphi_i(x) = \begin{cases} 1 - \dfrac{|x - q_i|}{w} & \text{if } |x - q_i| < w \\ 0 & \text{otherwise} \end{cases}$$

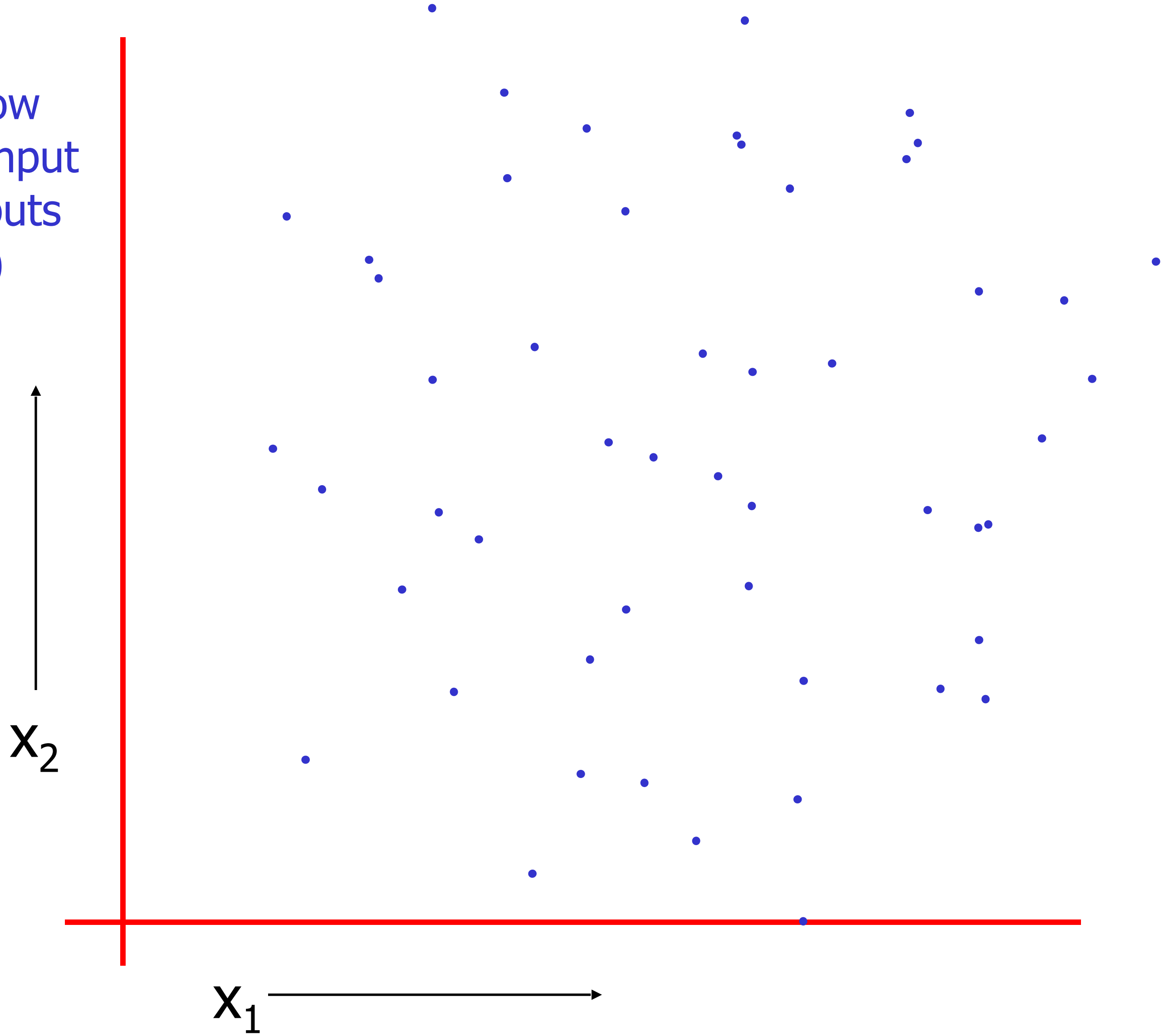


And this is simply a basis function regression problem!

We know how to find the least squares h_{ii}s!

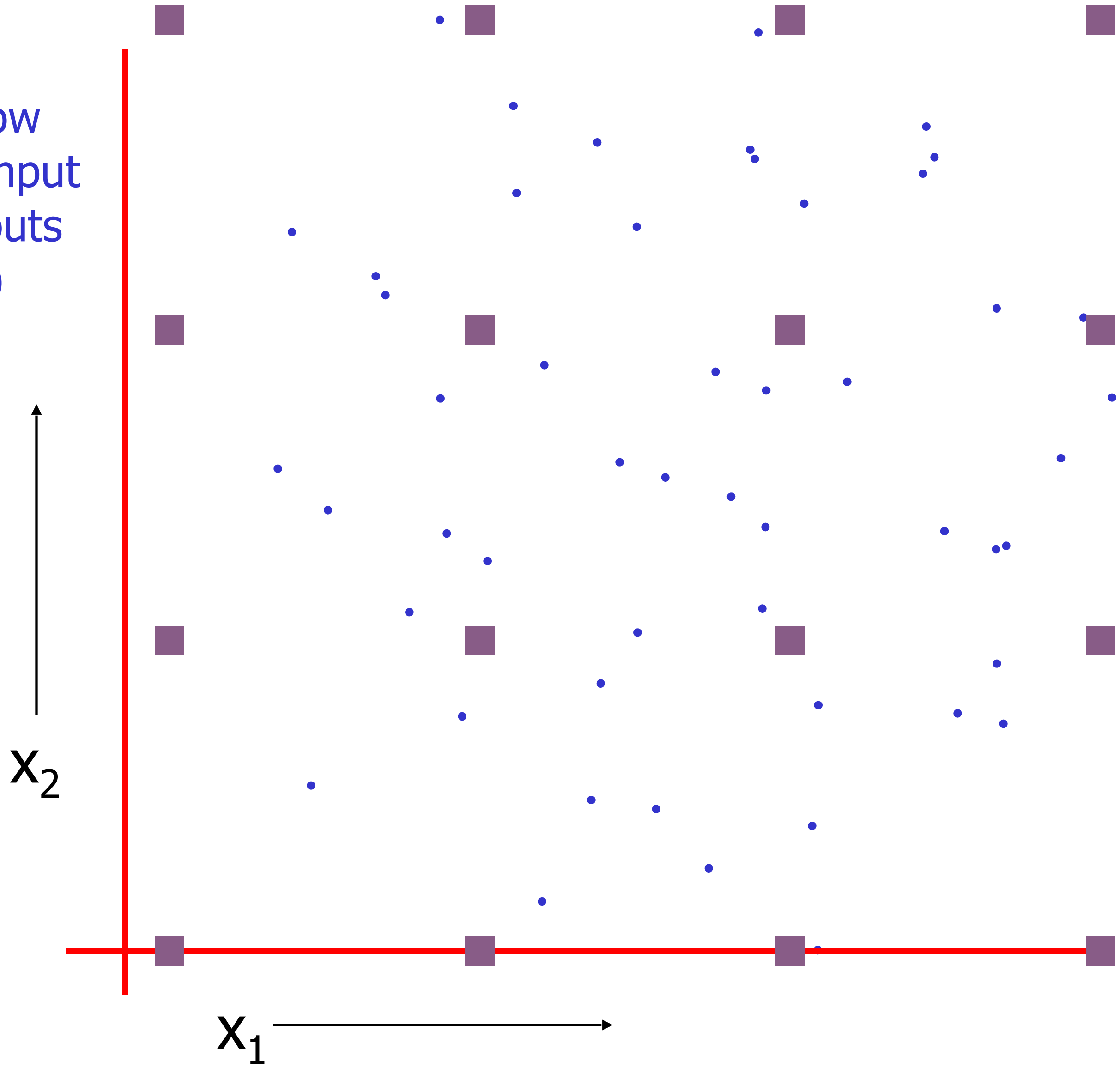
In two dimensions…

Blue dots show locations of input vectors (outputs not depicted)



In two dimensions…

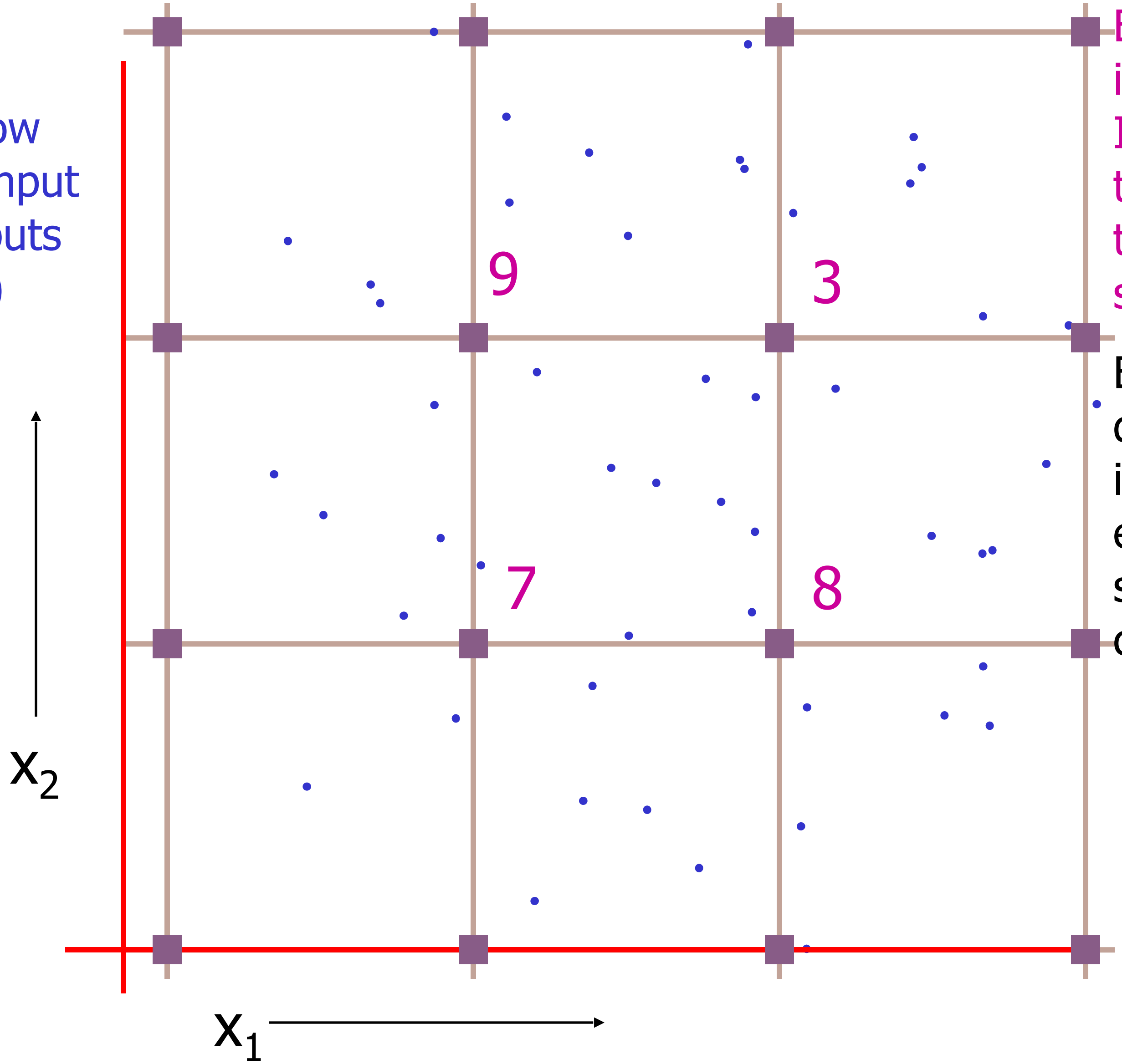
Blue dots show locations of input vectors (outputs not depicted)



Each purple dot is a knot point. It will contain the height of the estimated surface

In two dimensions…

Blue dots show locations of input vectors (outputs not depicted)



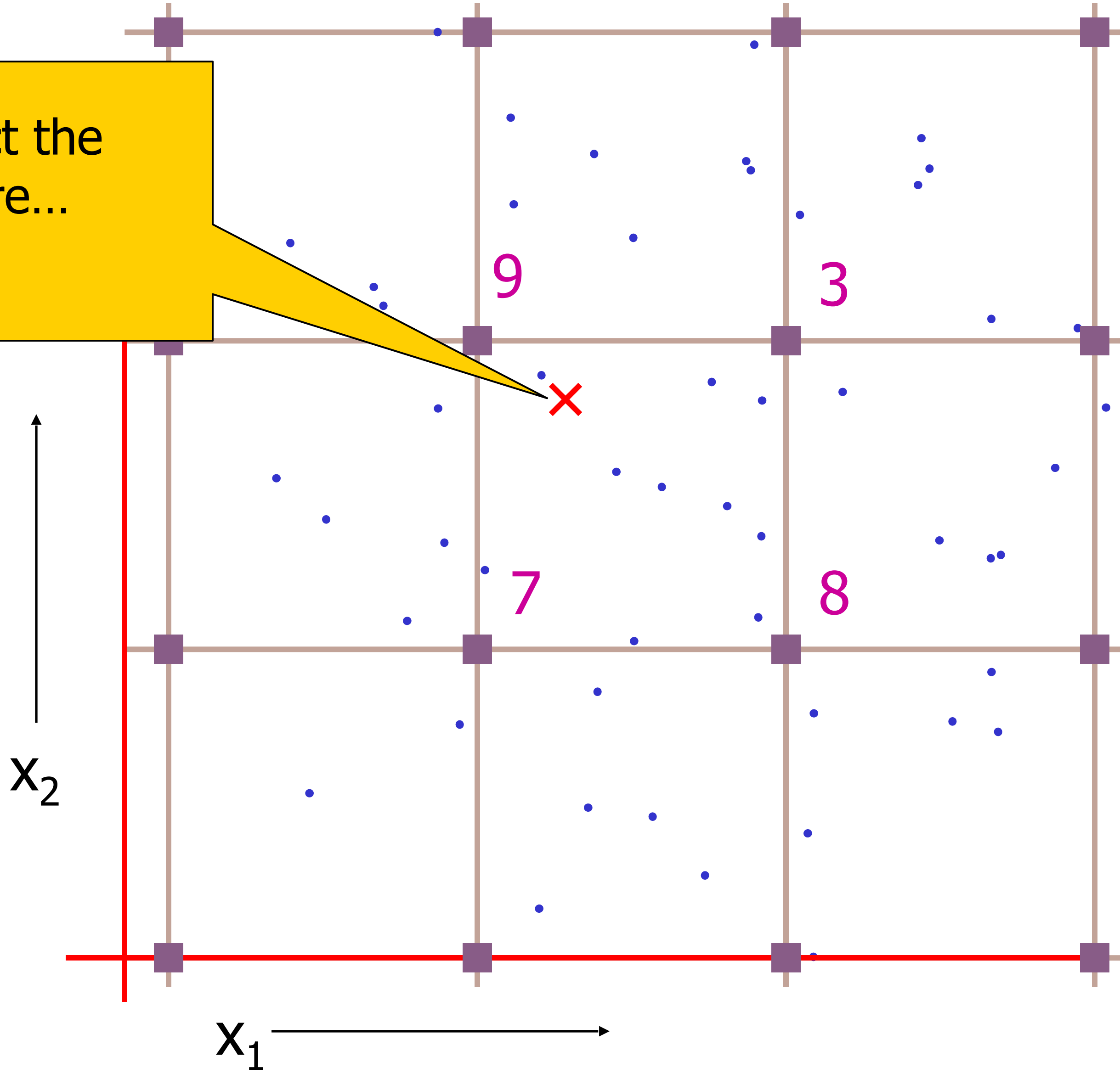
Each purple dot is a knot point. It will contain the height of the estimated surface

But how do we do the interpolation to ensure that the surface is continuous?

In two dimensions…

Bl
lo
ve
nc

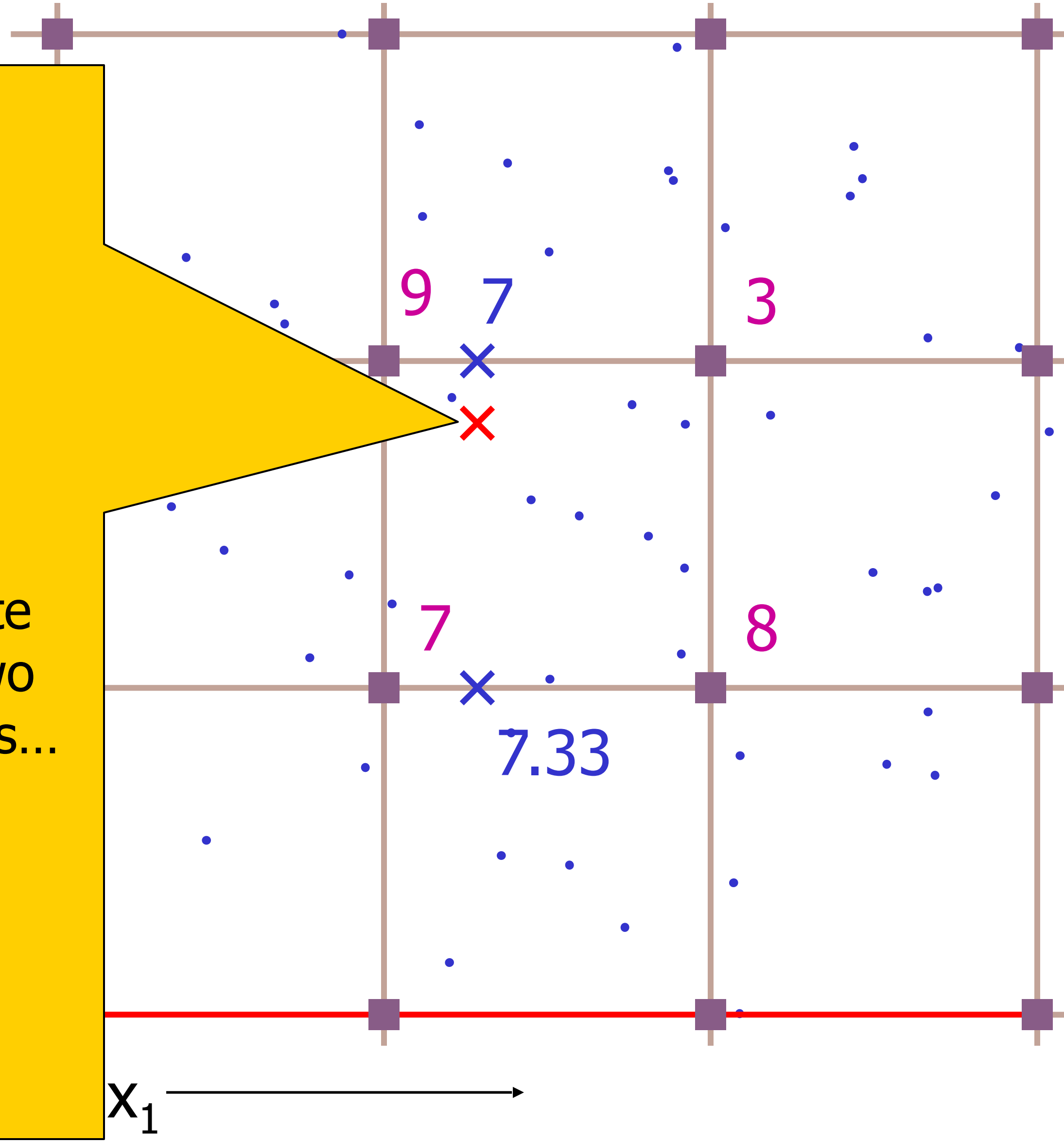
To predict the value here…



Each purple dot is a knot point. It will contain the height of the estimated surface

But how do we do the interpolation to ensure that the surface is continuous?

In two dimensions…



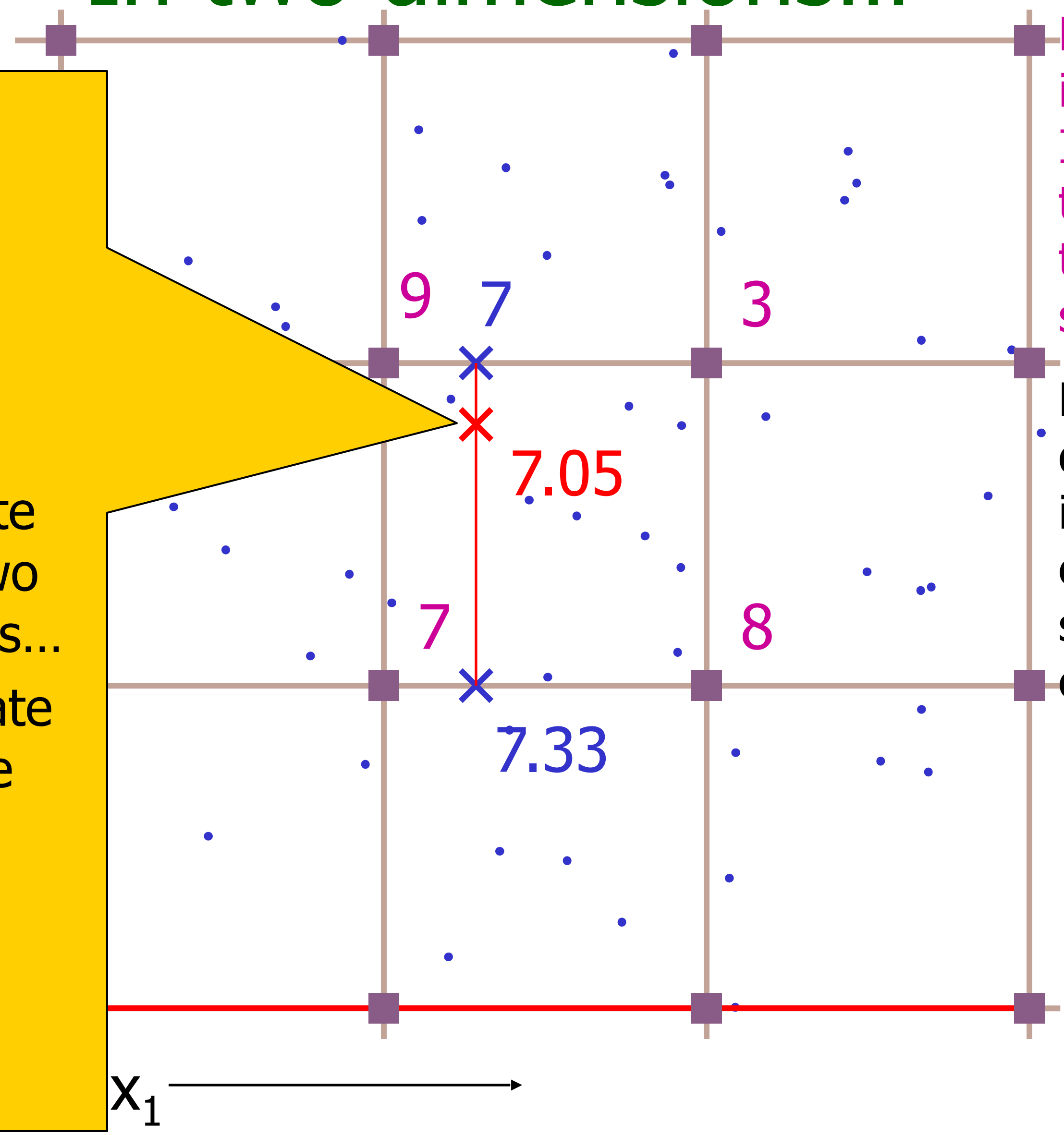
To predict the value here…

First interpolate its value on two opposite edges…

Each purple dot is a knot point. It will contain the height of the estimated surface

But how do we do the interpolation to ensure that the surface is continuous?

In two dimensions…



Bl
lo
ve
no

To predict the value here…

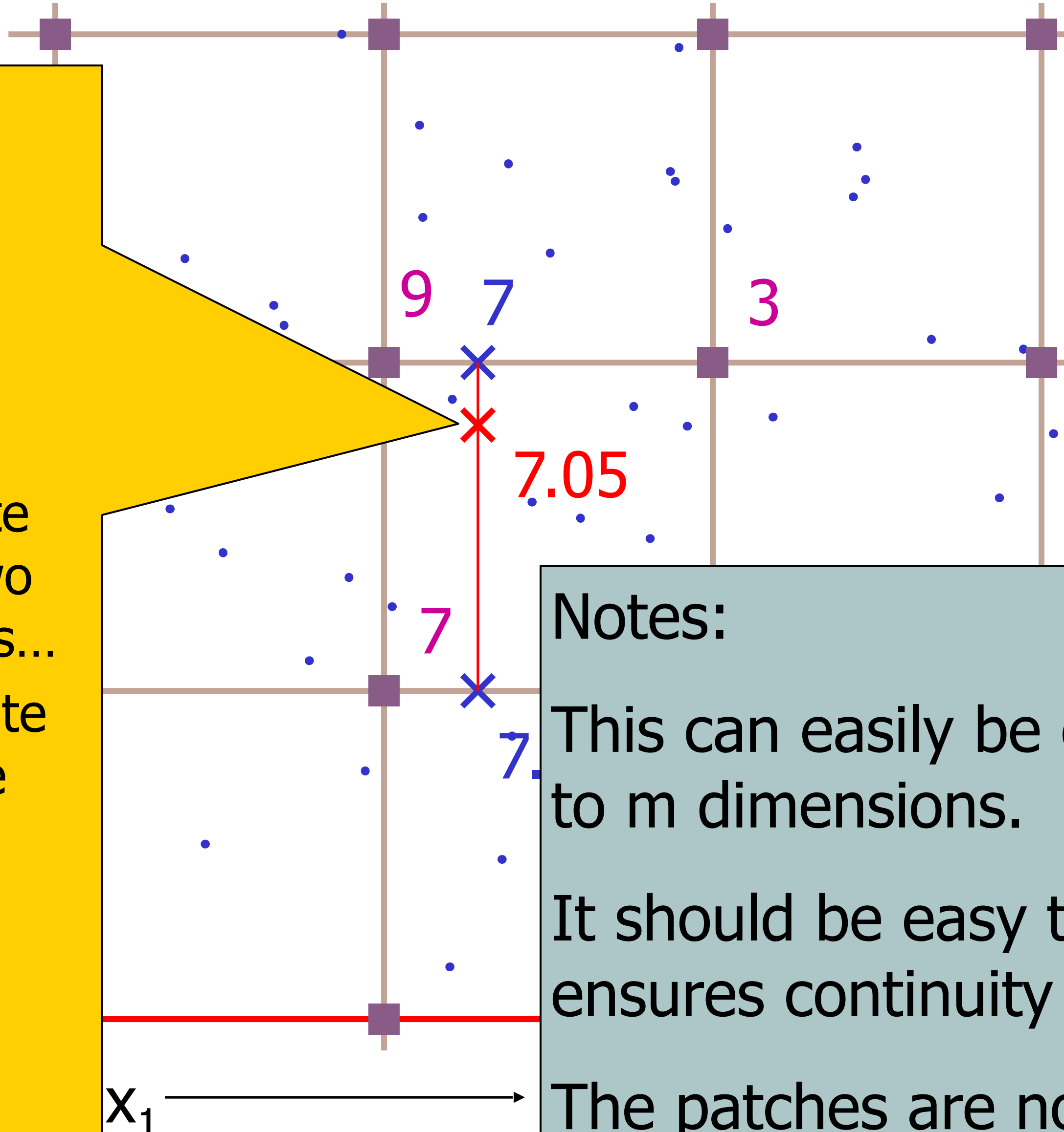
First interpolate its value on two opposite edges…

Then interpolate between those two values

Each purple dot is a knot point. It will contain the height of the estimated surface

But how do we do the interpolation to ensure that the surface is continuous?

In two dimensions…



Each purple dot is a knot point. It will contain the height of the estimated surface

But how do we do the interpolation to ensure that the

To predict the value here…

First interpolate its value on two opposite edges…

Then interpolate between those two values

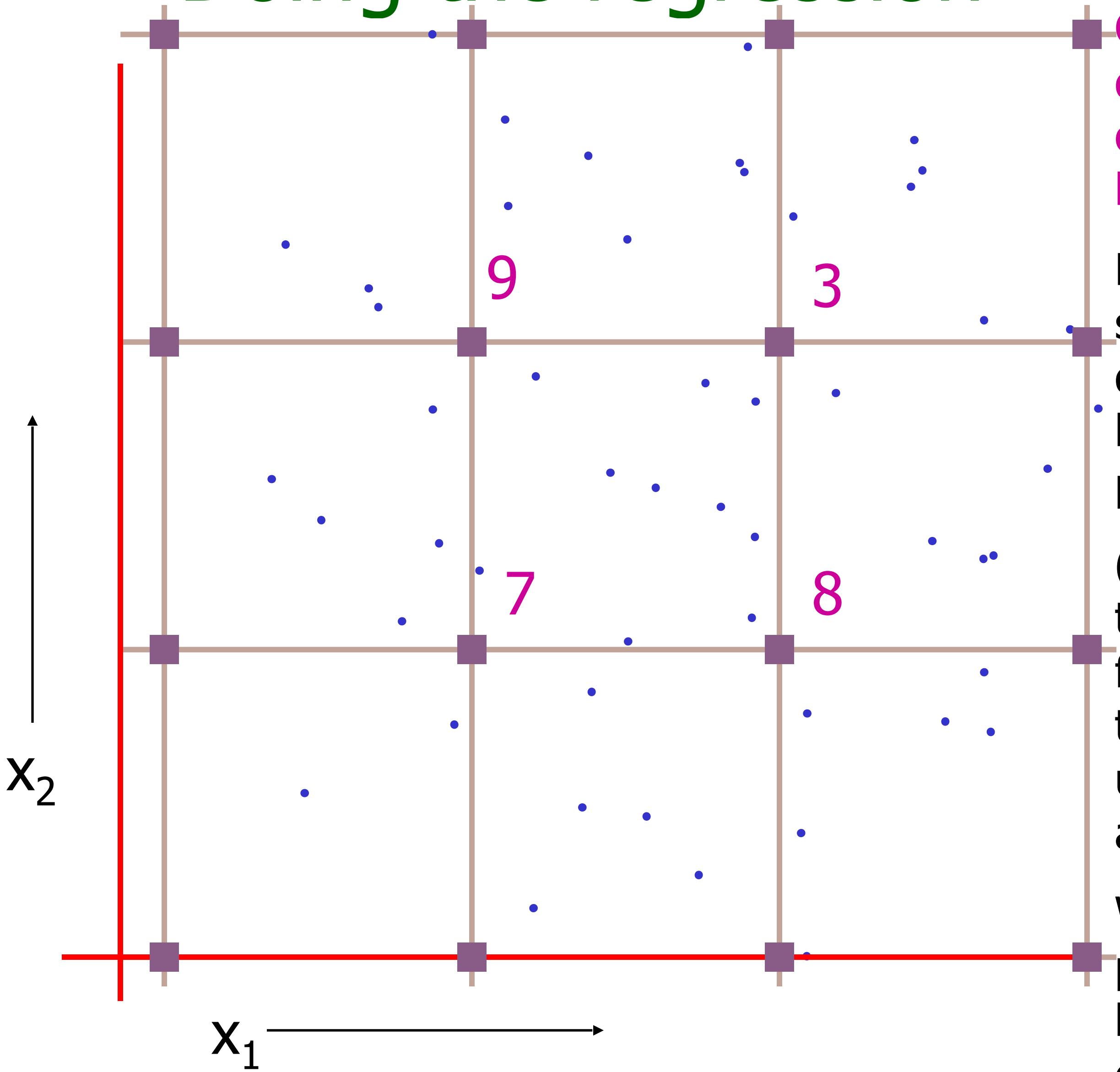
Notes:

This can easily be generalized to m dimensions.

It should be easy to see that it ensures continuity

The patches are not linear

Doing the regression



Given data, how do we find the optimal knot heights?

Happily, it's simply a two-dimensional basis function problem.

(Working out the basis functions is tedious, unilluminating, and easy)

What's the problem in higher dimensions?

1: MARS

- Multivariate Adaptive Regression Splines
- Invented by Jerry Friedman (one of Andrew's heroes)
- Simplest version:

Let's assume the function we are learning is of the following form:

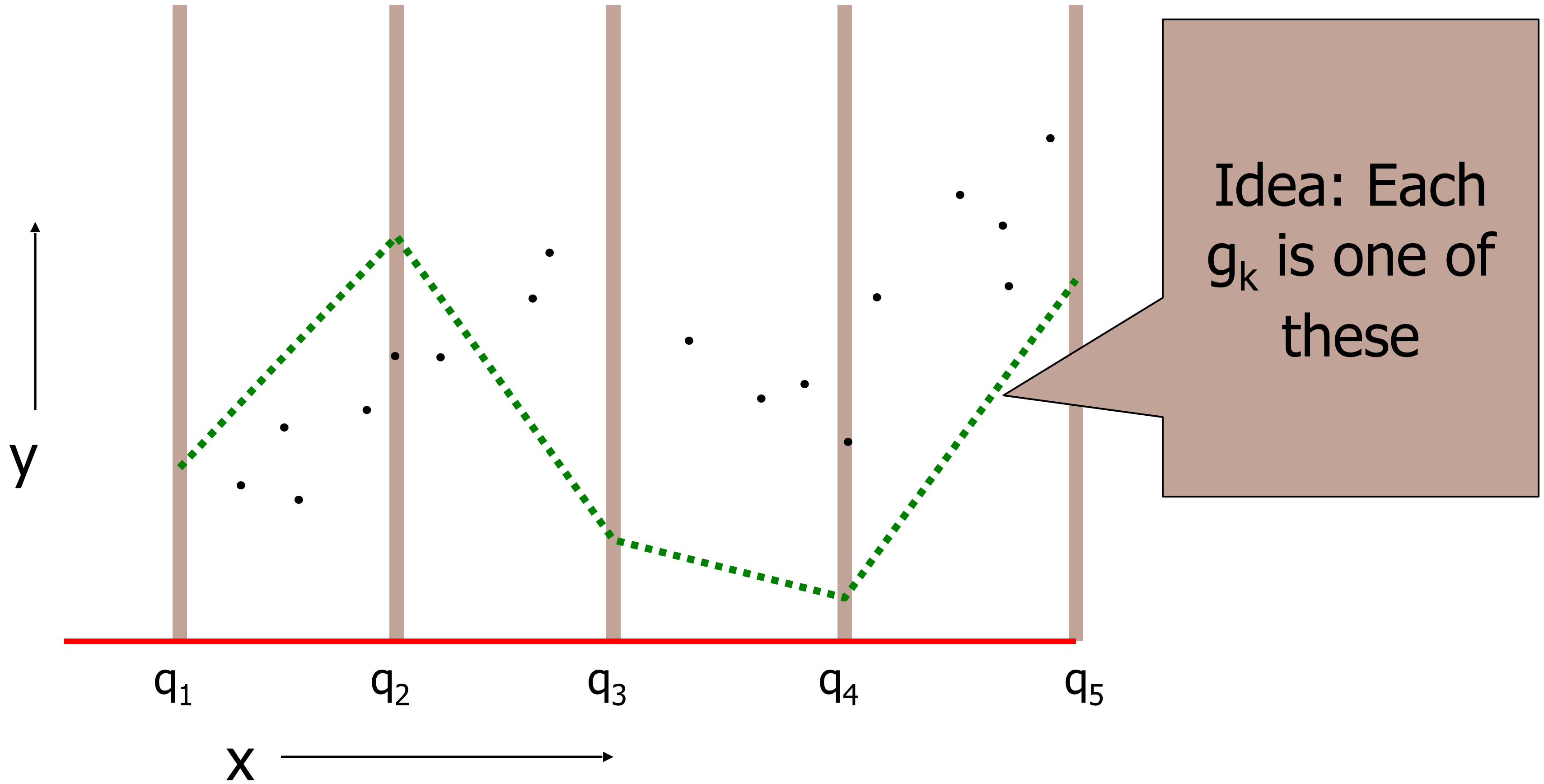
$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} g_k(x_k)$$

Instead of a linear combination of the inputs, it's a linear combination of non-linear functions of individual inputs

MARS

$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} g_k(x_k)$$

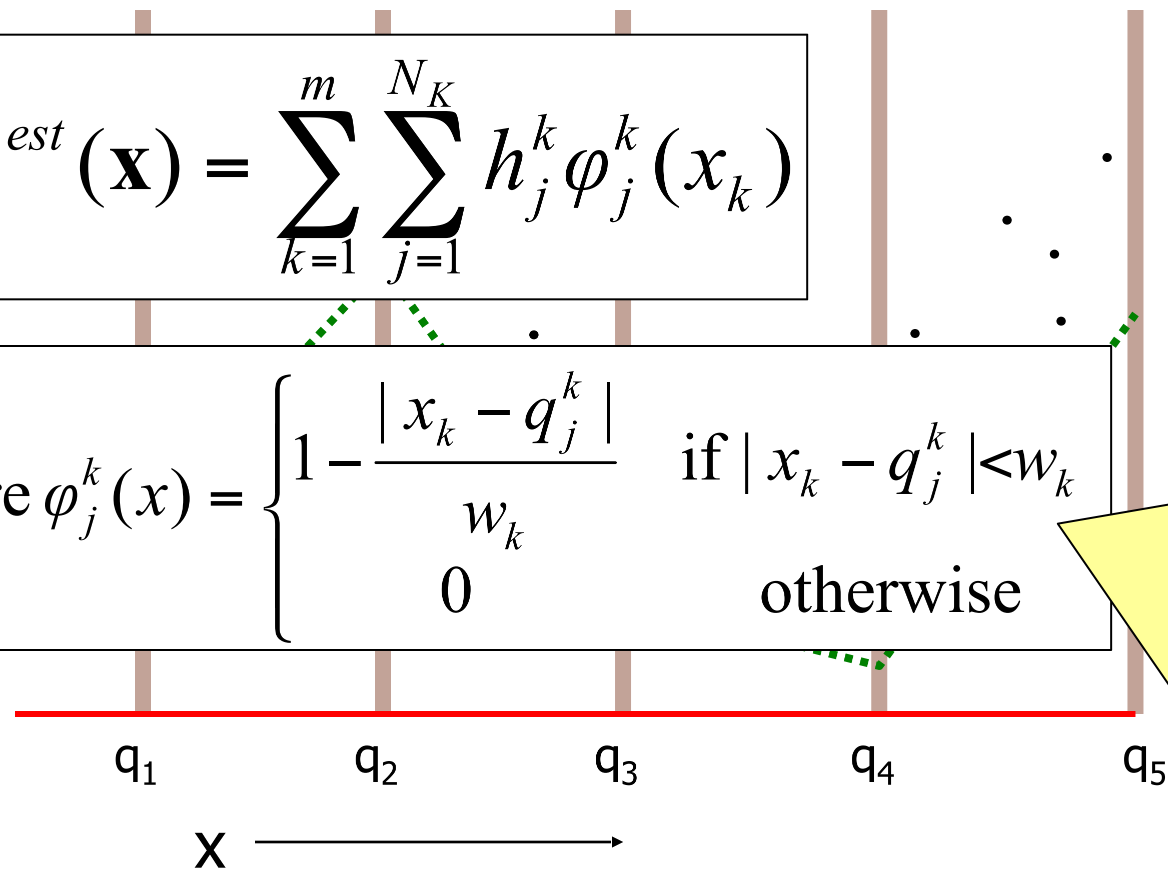
Instead of a linear combination of the inputs, it's a linear combination of non-linear functions of individual inputs



MARS

$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} g_k(x_k)$$

Instead of a linear combination of the inputs, it's a linear combination of non-linear functions of individual inputs



$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} \sum_{j=1}^{N_K} h_j^k \varphi_j^k(x_k)$$

$$\text{where } \varphi_j^k(x) = \begin{cases} 1 - \dfrac{|x_k - q_j^k|}{w_k} & \text{if } |x_k - q_j^k| < w_k \\ 0 & \text{otherwise} \end{cases}$$

q^k_j : The location of the j'th knot in the k'th dimension

h^k_j : The regressed height of the j'th knot in the k'th dimension

w^k: The spacing between knots in the kth dimension

That's not complicated enough!

- Okay, now let's get serious. We'll allow arbitrary "two-way interactions":

$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} g_k(x_k) + \sum_{k=1}^{m} \sum_{t=k+1}^{m} g_{kt}(x_k, x_t)$$

The function we're learning is allowed to be a sum of non-linear functions over all one-d and 2-d subsets of attributes

Can still be expressed as a linear combination of basis functions

Thus learnable by linear regression

Full MARS: Uses cross-validation to choose a subset of subspaces, knot resolution and other parameters.

If you like MARS…

…See also CMAC (Cerebellar Model Articulated Controller) by James Albus (another of Andrew’s heroes)

- Many of the same gut-level intuitions
- But entirely in a neural-network, biologically plausible way
  - (All the low dimensional functions are by means of lookup tables, trained with a delta-rule and using a clever blurred update and hash-tables)

Where are we now?

Inputs → Inference Engine Learn	$P(E_1 \mid E_2)$	Joint DE, Bayes Net Structure Learning
Inputs → Classifier	Predict category	Dec Tree, Sigmoid Perceptron, Sigmoid N.Net, Gauss/Joint BC, Gauss Naïve BC, N.Neigh, Bayes Net Based BC, Cascade Correlation
Inputs → Density Estimator	Prob-ability	Joint DE, Naïve DE, Gauss/Joint DE, Gauss Naïve DE, Bayes Net Structure Learning
Inputs → Regressor	Predict real no.	Linear Regression, Polynomial Regression, Perceptron, Neural Net, N.Neigh, Kernel, LWR, RBFs, Robust Regression, Cascade Correlation, Regression Trees, GMDH, Multilinear Interp, MARS

What You Should Know

- For each of the eight methods you should be able to summarize briefly what they do and outline how they work.
- You should understand them well enough that given access to the notes you can quickly re-understand them at a moments notice
- But you don’t have to memorize all the details
- In the right context any one of these eight might end up being really useful to you one day! You should be able to recognize this when it occurs.

Citations

Radial Basis Functions

T. Poggio and F. Girosi, Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks, Science, 247, 978--982, 1989

LOESS

W. S. Cleveland, Robust Locally Weighted Regression and Smoothing Scatterplots, Journal of the American Statistical Association, 74, 368, 829-836, December, 1979

GMDH etc

http://www.inf.kiev.ua/GMDH-home/

P. Langley and G. L. Bradshaw and H. A. Simon, Rediscovering Chemistry with the BACON System, Machine Learning: An Artificial Intelligence Approach, R. S. Michalski and J. G. Carbonnell and T. M. Mitchell, Morgan Kaufmann, 1983

Regression Trees etc

L. Breiman and J. H. Friedman and R. A. Olshen and C. J. Stone, Classification and Regression Trees, Wadsworth, 1984

J. R. Quinlan, Combining Instance-Based and Model-Based Learning, Machine Learning: Proceedings of the Tenth International Conference, 1993

Cascade Correlation etc

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1990. http://citeseer.nj.nec.com/fahlman91cascadecorrelation.html

J. H. Friedman and W. Stuetzle, Projection Pursuit Regression, Journal of the American Statistical Association, 76, 376, December, 1981

MARS

J. H. Friedman, Multivariate Adaptive Regression Splines, Department for Statistics, Stanford University, 1988, Technical Report No. 102