Predicting Real-valued outputs: an introduction to Regression

Andrew W. Moore

Professor

School of Computer Science

Carnegie Mellon University

www.cs.cmu.edu/~awm

awm@cs.cmu.edu

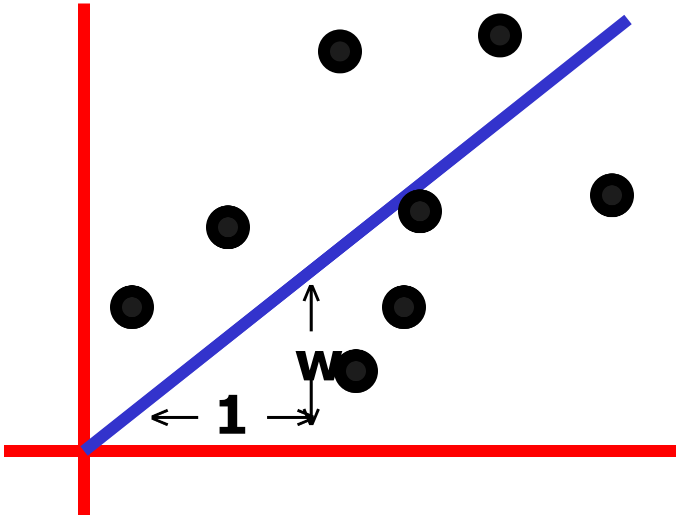
412-268-7599

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

This is reordered material from the Neural Nets lecture and the "Favorite Regression Algorithms" lecture

Single-Parameter Linear Regression

Linear Regression



DATASET

inputs	outputs
$x_1 = 1$	$y_1 = 1$
$x_2 = 3$	$y_2 = 2.2$
$x_3 = 2$	$y_3 = 2$
$x_4 = 1.5$	$y_4 = 1.9$
$x_5 = 4$	$y_5 = 3.1$

Linear regression assumes that the expected value of the output given an input, E[y|x], is linear.

Simplest case: Out(x) = wx for some unknown w.

Given the data, we can estimate w.

1-parameter linear regression

Assume that the data is formed by

$$y_i = wx_i + noise_i$$

where…

- the noise signals are independent
- the noise has a normal distribution with mean 0 and unknown variance σ^2

p(y|w,x) has a normal distribution with

- mean wx
- variance σ^2

Bayesian Linear Regression

$$p(y|w,x) = \text{Normal (mean } wx, \text{ var } \sigma^2)$$

We have a set of datapoints (x_1,y_1) (x_2,y_2) … (x_n,y_n) which are EVIDENCE about w.

We want to infer w from the data.

$$p(w|x_1, x_2, x_3,\ldots x_n, y_1, y_2\ldots y_n)$$

- You can use BAYES rule to work out a posterior distribution for w given the data.
- Or you could do Maximum Likelihood Estimation

Maximum likelihood estimation of w

Asks the question:

"For which value of w is this data most likely to have happened?"

<=>

For what w is

$p(y_1, y_2 \ldots y_n \mid x_1, x_2, x_3, \ldots x_n, w)$ maximized?

<=>

For what w is

$$\prod_{i=1}^{n} p(y_i \mid w, x_i) \text{ maximized?}$$

For what w is

$$\prod_{i=1}^{n} p(y_i | w, x_i) \text{ maximized?}$$

For what w is

$$\prod_{i=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2\right) \text{ maximized?}$$

For what w is

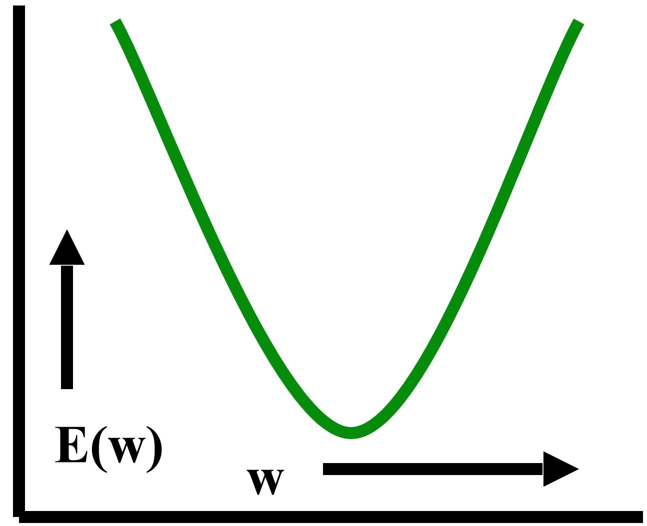
$$\sum_{i=1}^{n} -\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2 \text{ maximized?}$$

For what w is

$$\sum_{i=1}^{n} \left(y_i - wx_i\right)^2 \text{ minimized?}$$

Linear Regression

The maximum likelihood w is the one that minimizes sum-of-squares of <u>residuals</u>



$$E = \sum_i \left(y_i - wx_i\right)^2$$

$$= \sum_i y_i^2 - \left(2\sum x_i y_i\right)w + \left(\sum x_i^2\right)w^2$$

We want to minimize a quadratic function of w.

Linear Regression

Easy to show the sum of squares is minimized when

$$w = \frac{\sum x_i y_i}{\sum x_i^{\,2}}$$

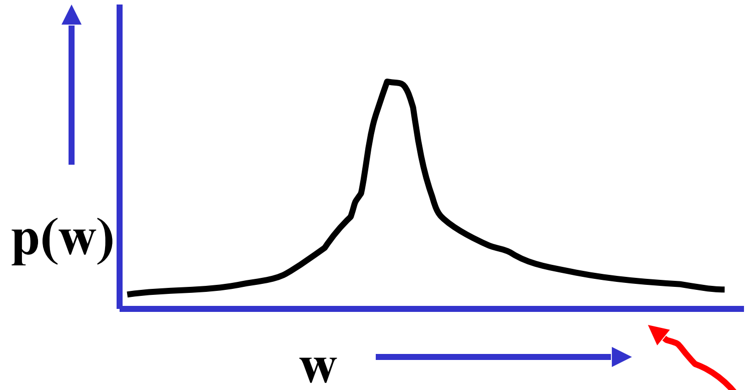
The maximum likelihood model is $\mathrm{Out}(x) = wx$

We can use it for prediction

Linear Regression

Easy to show the sum of squares is minimized when

$$w = \frac{\sum x_i y_i}{\sum x_i^2}$$



The maximum likelihood model is $\text{Out}(x) = wx$

We can use it for prediction

Note: In Bayesian stats you'd have ended up with a prob dist of w

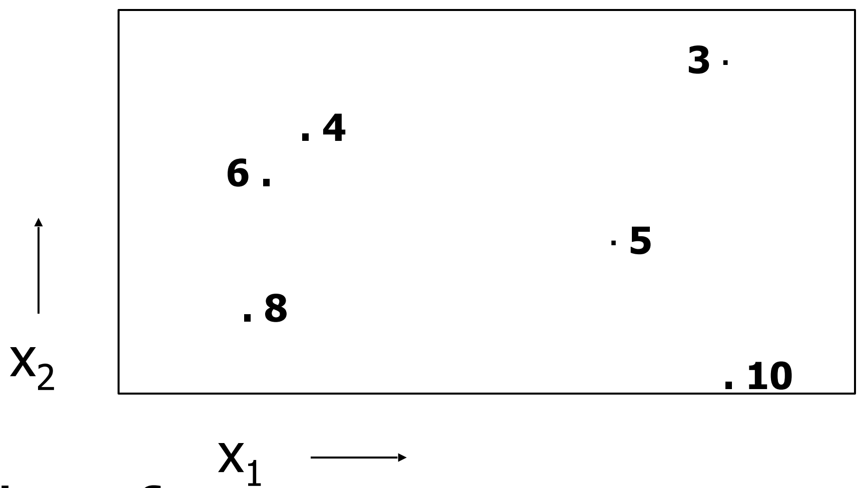
And predictions would have given a prob dist of expected output

Often useful to know your confidence. Max likelihood can give some kinds of confidence too.

Multivariate Linear Regression

Multivariate Regression

What if the inputs are vectors?



2-d input example

Dataset has form

$$
\begin{array}{ll}
\mathbf{x}_1 & y_1 \\
\mathbf{x}_2 & y_2 \\
\mathbf{x}_3 & y_3 \\
.: & : \\
. & \\
\mathbf{x}_R & y_R
\end{array}
$$

Multivariate Regression

Write matrix X and Y thus:

$$\mathbf{x} = \begin{bmatrix} \ldots\ldots\mathbf{x}_1\ldots\ldots \\ \ldots\ldots\mathbf{x}_2\ldots\ldots \\ \boxed{?} \\ \ldots\ldots\mathbf{x}_R\ldots\ldots \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ & & \boxed{?} & \\ x_{R1} & x_{R2} & \ldots & x_{Rm} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \boxed{?} \\ y_R \end{bmatrix}$$

(there are R datapoints. Each input has m components)

The linear regression model assumes a vector **w** such that

$$\text{Out}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = w_1x[1] + w_2x[2] + \ldots.w_mx[D]$$

The max. likelihood **w** is $\mathbf{w} = (X^TX)^{-1}(X^TY)$

Multivariate Regression

Write matrix X and Y thus:

$$\mathbf{x} = \begin{bmatrix} \ldots\ldots\mathbf{x}_1\ldots\ldots \\ \ldots\ldots\mathbf{x}_2\ldots\ldots \\ \boxed{?} \\ \ldots\ldots\mathbf{x}_R\ldots\ldots \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ & & \boxed{?} & \\ x_{R1} & x_{R2} & \ldots & x_{Rm} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \boxed{?} \\ y_R \end{bmatrix}$$

(there are R datapoints. Each input

IMPORTANT EXERCISE: PROVE IT !!!!!

The linear regression model assumes a vector **w** such that

$$\text{Out}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = w_1x[1] + w_2x[2] + \ldots.w_mx[D]$$

The max. likelihood **w** is **w** = $(X^TX)^{-1}(X^TY)$

Multivariate Regression (con't)

The max. likelihood $\mathbf{w}$ is $\mathbf{w} = (X^T X)^{-1}(X^T Y)$

$X^T X$ is an m x m matrix: i,j'th elt is $\displaystyle\sum_{k=1}^{R} x_{ki} x_{kj}$

$X^T Y$ is an m-element vector: i'th elt $\displaystyle\sum_{k=1}^{R} x_{ki} y_k$

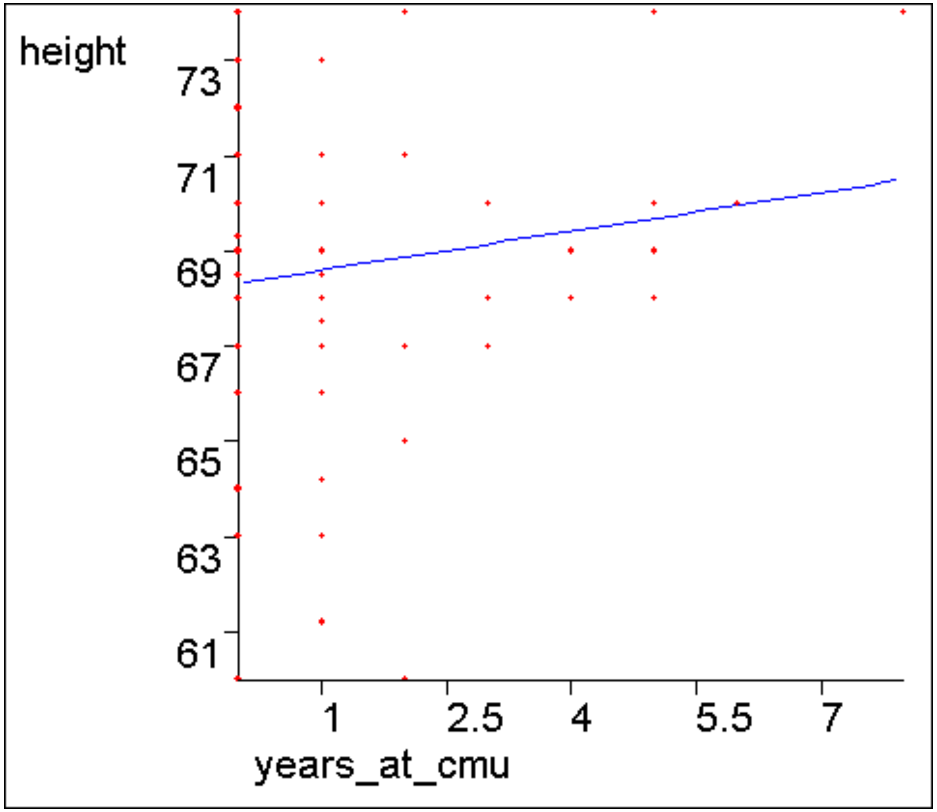
Constant Term in Linear Regression

What about a constant term?

We may expect linear data that does not go through the origin.

Statisticians and Neural Net Folks all agree on a simple obvious hack.

Can you guess??



The constant term

- The trick is to create a fake input "X_0" that always takes the value 1

X_1	X_2	Y
2	4	16
3	4	17
5	5	20

Before:

$Y = w_1X_1 + w_2X_2$

…has to be a poor model

X_0	X_1	X_2	Y
1	2	4	16
1	3	4	17
1	5	5	20

After:

$Y = w_0X_0 + w_1X_1 + w_2X_2$

$= w_0 + w_1X_1 + w_2X_2$

…has a fine constant term

In this example, You should be able to see the MLE w_0 , w_1 and w_2 by inspection

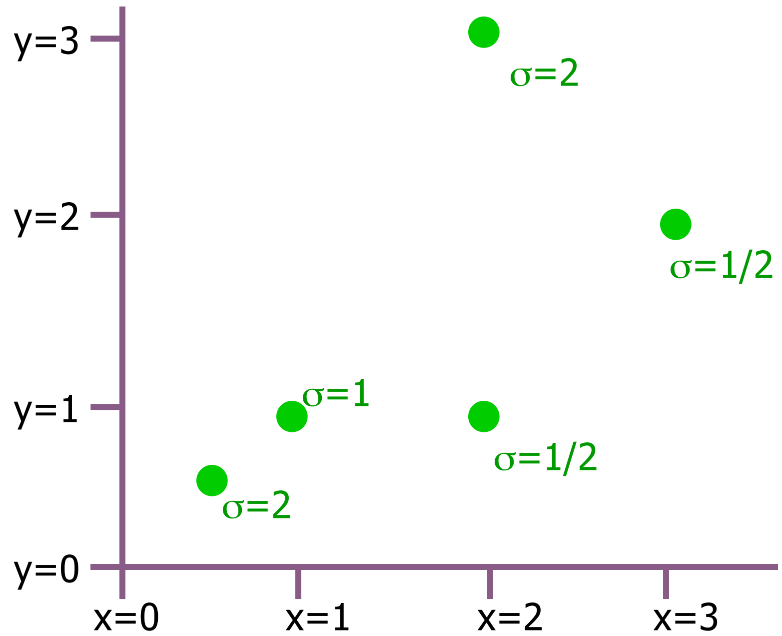
Heteroscedasticity...

Linear Regression with varying noise

Regression with varying noise

- Suppose you know the variance of the noise that was added to each datapoint.

x_i	y_i	σ_i^2
½	½	4
1	1	1
2	1	1/4
2	3	4
3	2	1/4



Assume $$y_i \sim N(wx_i, \sigma_i^2)$$

What's the MLE estimate of w?

MLE estimation with varying noise

$$\underset{w}{\operatorname{argmax}} \log p(y_1, y_2, \ldots, y_R \mid x_1, x_2, \ldots, x_R, \sigma_1^2, \sigma_2^2, \ldots, \sigma_R^2, w) =$$

$$\underset{w}{\operatorname{argmin}} \sum_{i=1}^{R} \frac{(y_i - wx_i)^2}{\sigma_i^2} =$$

Assuming independence among noise and then plugging in equation for Gaussian and simplifying.

$$\left(w \text{ such that } \sum_{i=1}^{R} \frac{x_i (y_i - wx_i)}{\sigma_i^2} = 0 \right) =$$

Setting dLL/dw equal to zero

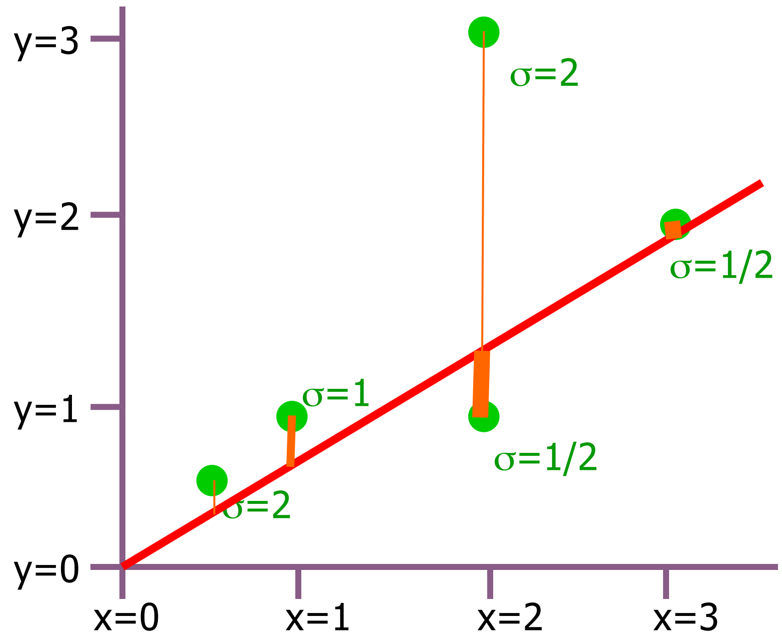
$$\frac{\left(\sum_{i=1}^{R} \frac{x_i y_i}{\sigma_i^2} \right)}{\left(\sum_{i=1}^{R} \frac{x_i^2}{\sigma_i^2} \right)}$$

Trivial algebra

This is Weighted Regression

- We are asking to minimize the weighted sum of squares

$$\underset{w}{\operatorname{argmin}} \sum_{i=1}^{R} \frac{(y_i - wx_i)^2}{\sigma_i^2}$$



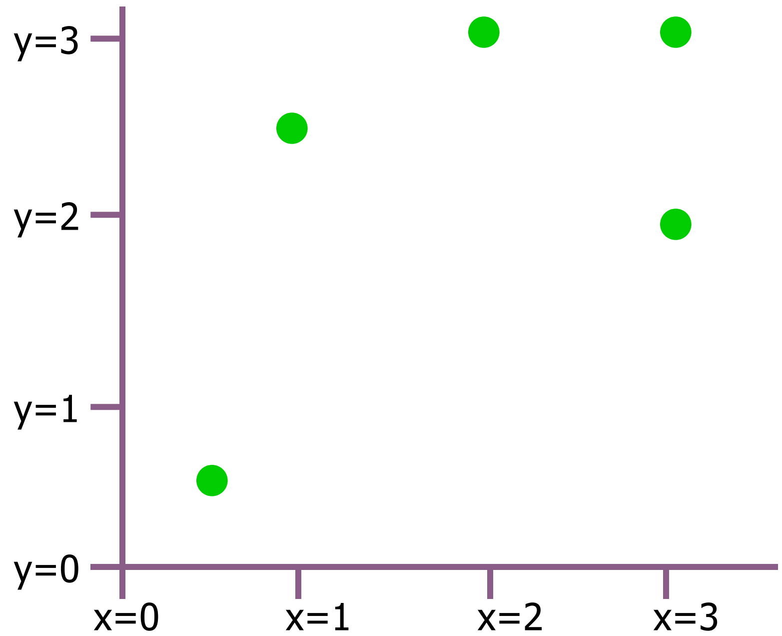
where weight for i'th datapoint is $\frac{1}{\sigma_i^2}$

Non-linear Regression

Non-linear Regression

- Suppose you know that y is related to a function of x in such a way that the predicted values have a non-linear dependence on w, e.g:

x_i	y_i
½	½
1	2.5
2	3
3	2
3	3



Assume $$y_i \sim N(\sqrt{w + x_i}, \sigma^2)$$

What's the MLE estimate of w?

Non-linear MLE estimation

$$\underset{w}{\operatorname{argmax}} \log p(y_1, y_2, \ldots, y_R \mid x_1, x_2, \ldots, x_R, \sigma, w) =$$

$$\underset{w}{\operatorname{argmin}} \sum_{i=1}^{R} \left(y_i - \sqrt{w + x_i} \right)^2 =$$

Assuming i.i.d. and then plugging in equation for Gaussian and simplifying.

$$\left(w \text{ such that } \sum_{i=1}^{R} \frac{y_i - \sqrt{w + x_i}}{\sqrt{w + x_i}} = 0 \right) =$$

Setting dLL/dw equal to zero

Non-linear MLE estimation

$$\underset{w}{\operatorname{argmax}} \log p(y_1, y_2, \ldots, y_R \mid x_1, x_2, \ldots, x_R, \sigma, w) =$$

$$\underset{w}{\operatorname{argmin}} \sum_{i=1}^{R} \left(y_i - \sqrt{w + x_i} \right)^2 =$$

Assuming i.i.d. and then plugging in equation for Gaussian and simplifying.

$$\left(w \text{ such that } \sum_{i=1}^{R} \frac{y_i - \sqrt{w + x_i}}{\sqrt{w + x_i}} = 0 \right) =$$

Setting dLL/dw equal to zero

We're down the algebraic toilet

So guess what we do?

Non-linear MLE estimation

$$\underset{w}{\operatorname{argmax}} \log p(y_1, y_2, \ldots, y_R \mid x_1, x_2, \ldots, x_R, \sigma, w) =$$

$$w + x_i \Big)^2 =$$

$$+ x_i \quad = 0 \Bigg) =$$

Assuming i.i.d. and then plugging in equation for Gaussian and simplifying.

Setting dLL/dw equal to zero

Common (but not only) approach:

Numerical Solutions:

- Line Search
- Simulated Annealing
- Gradient Descent
- Conjugate Gradient
- Levenberg Marquart
- Newton's Method

We're down the algebraic toilet

Also, special purpose statistical-optimization-specific tricks such as E.M. (See Gaussian Mixtures lecture for introduction)

So guess what we do?

Polynomial Regression

Polynomial Regression

So far we've mainly been dealing with linear regression

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

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

(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
:	:	:

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

$y_1=7$..

$$\mathbf{Z}=\begin{bmatrix} 1 & 3 & 2 & 9 & 6 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ : & & & & & : \end{bmatrix} \qquad \mathbf{y}=\begin{bmatrix} 7 \\ 3 \\ : \end{bmatrix}$$

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

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

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

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
:	:	:

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

$$\mathbf{Z}=\begin{bmatrix} 1 & 3 & 2 & 9 & 6 & \ldots \\ 1 & 1 & 1 & 1 & 1 & \ldots \\ : & & & & & \ldots \end{bmatrix} \quad \mathbf{y}=\begin{bmatrix} 7 \\ 3 \\ : \end{bmatrix}$$

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

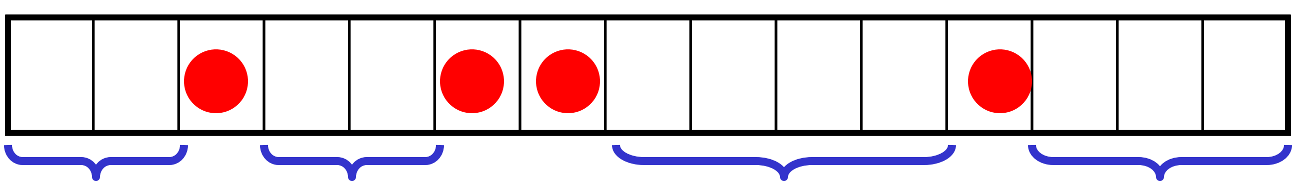
$$x_1^{q_1} x_2^{q_2} \dots x_m^{q_m} \text{ where } \sum_{i=1}^{m} q_i \le Q$$

= the number of unique terms of the form

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

= 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



Radial Basis Functions

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=

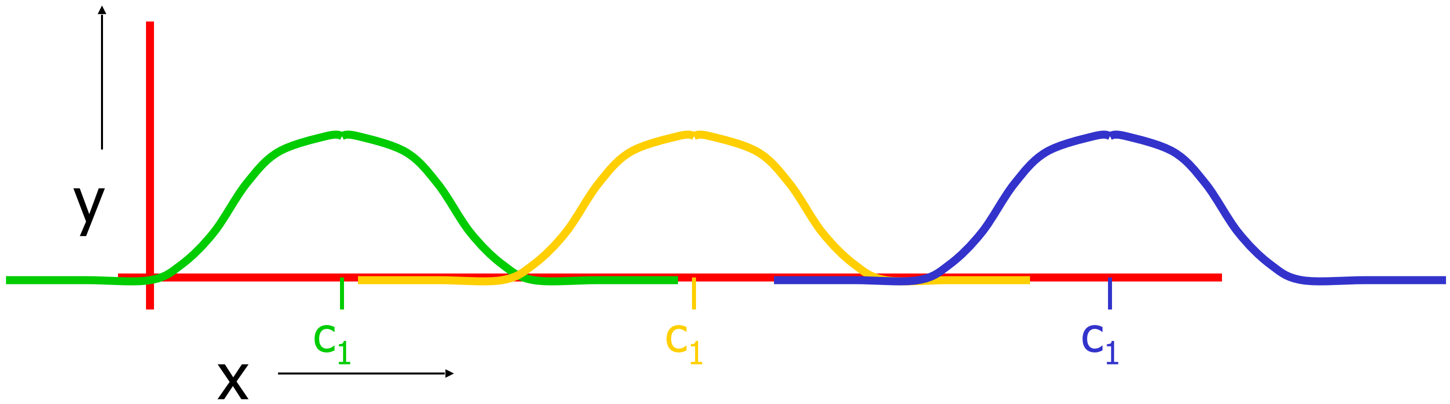
7
3
:

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

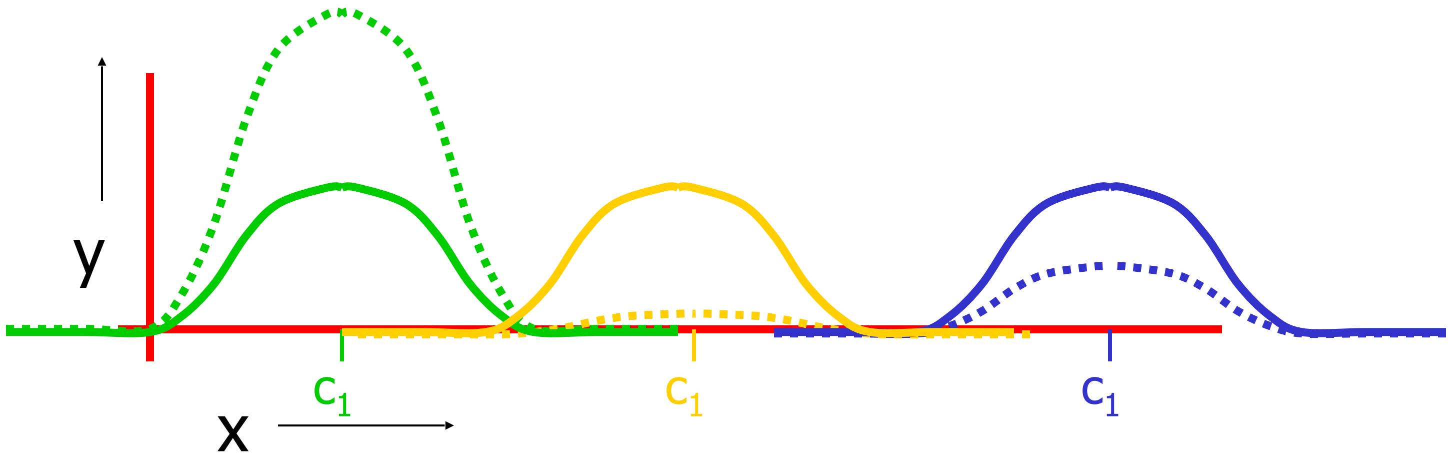


$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$

where

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

Example

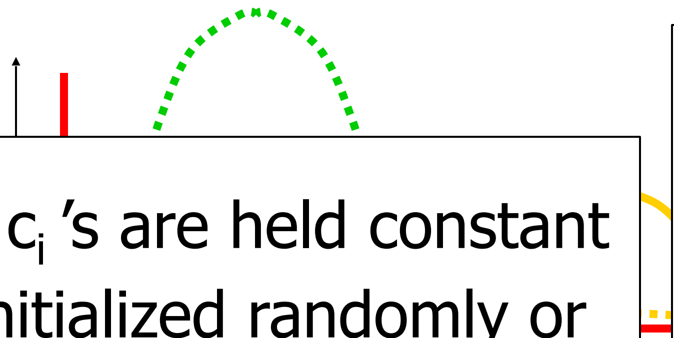


$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$

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*

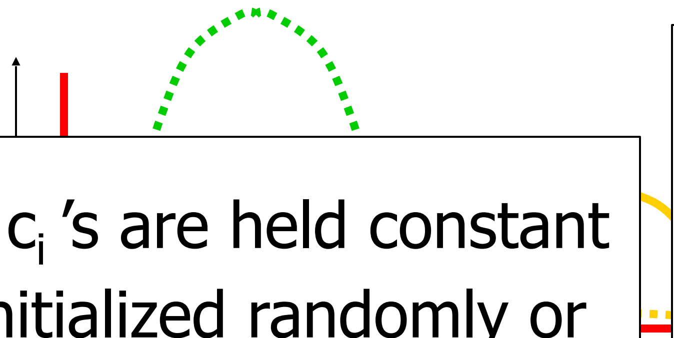
*Usually much better than the crappy overlap on my diagram

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

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

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

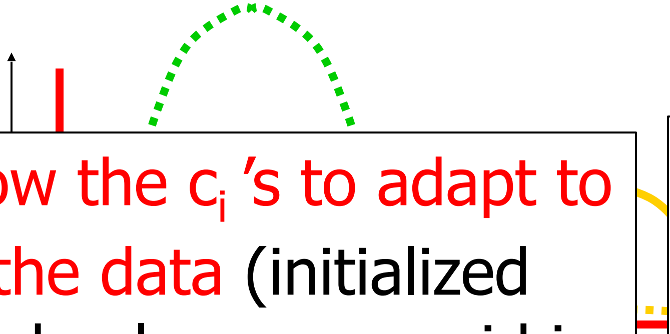
where

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

then given Q basis functions, define the matrix Z such that $Z_{kj} = \text{KernelFunction}(\ | x_k - c_i | \ / \ KW)$ 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)

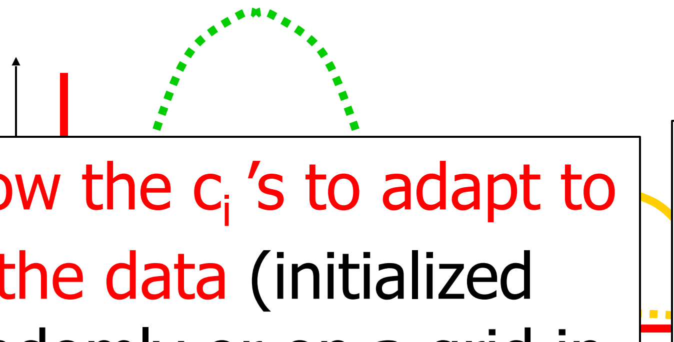
$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) \ldots 5\phi_3(x)$

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)

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

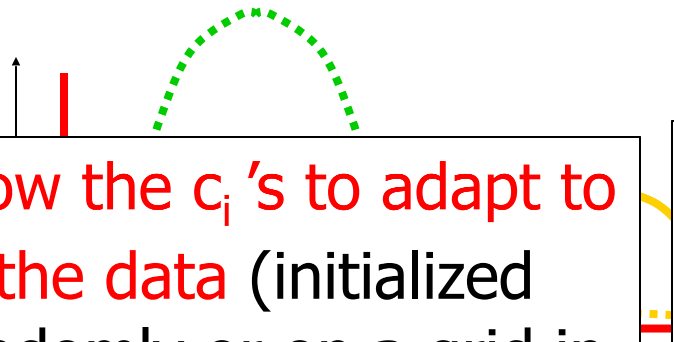
where

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

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 ?

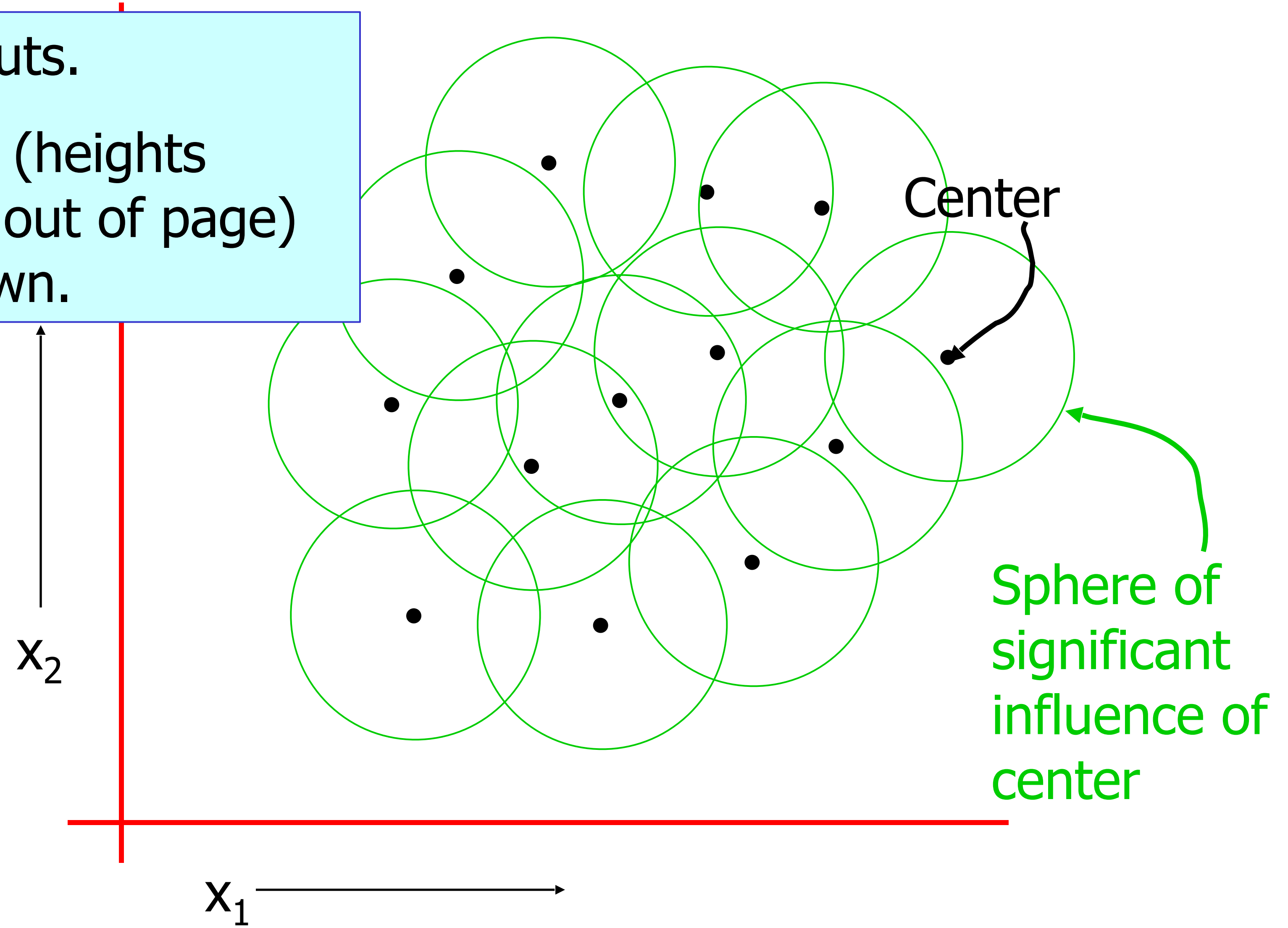
Answer: Gradient Descent

(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

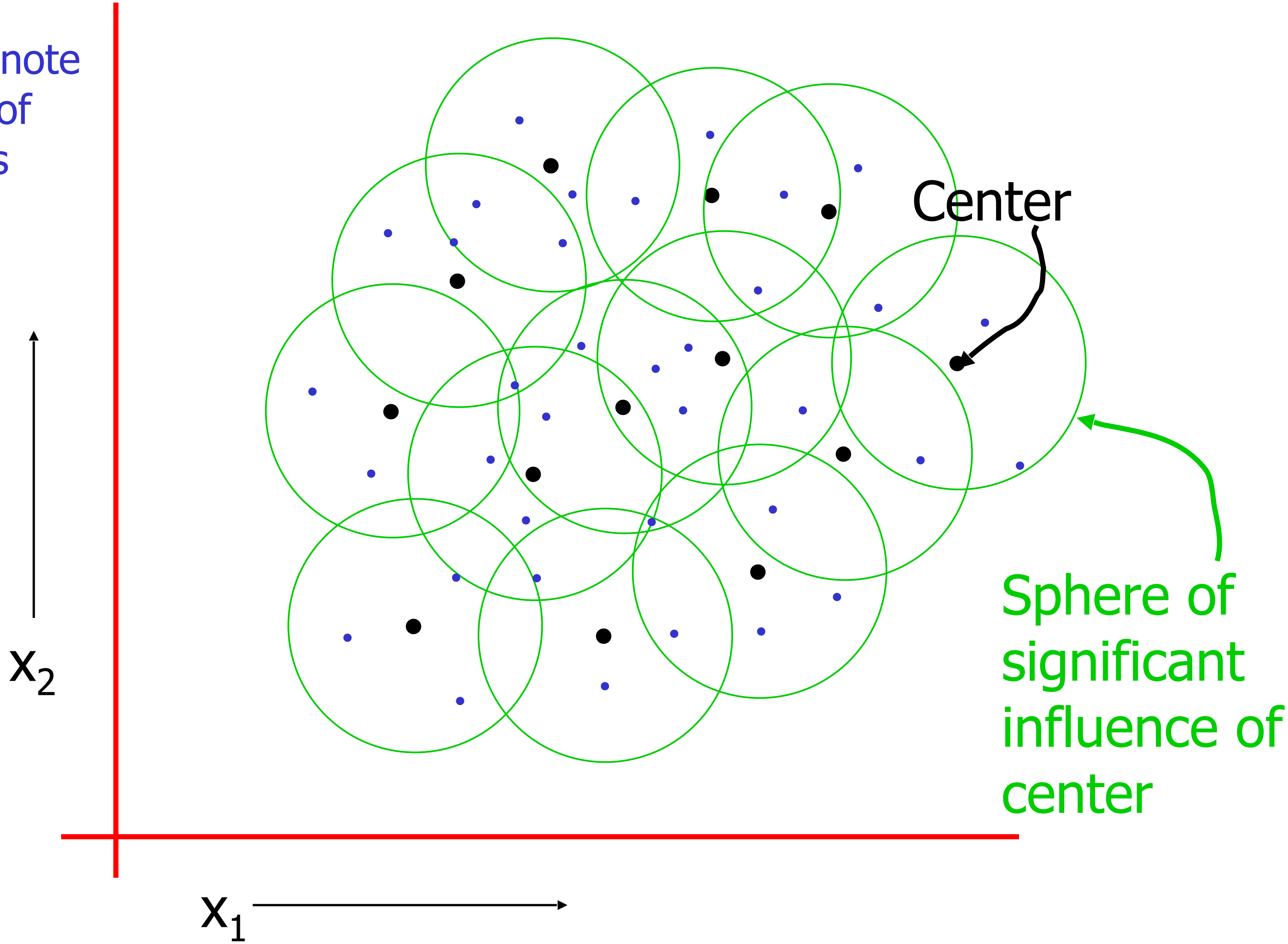
Two inputs.

Outputs (heights sticking out of page) not shown.



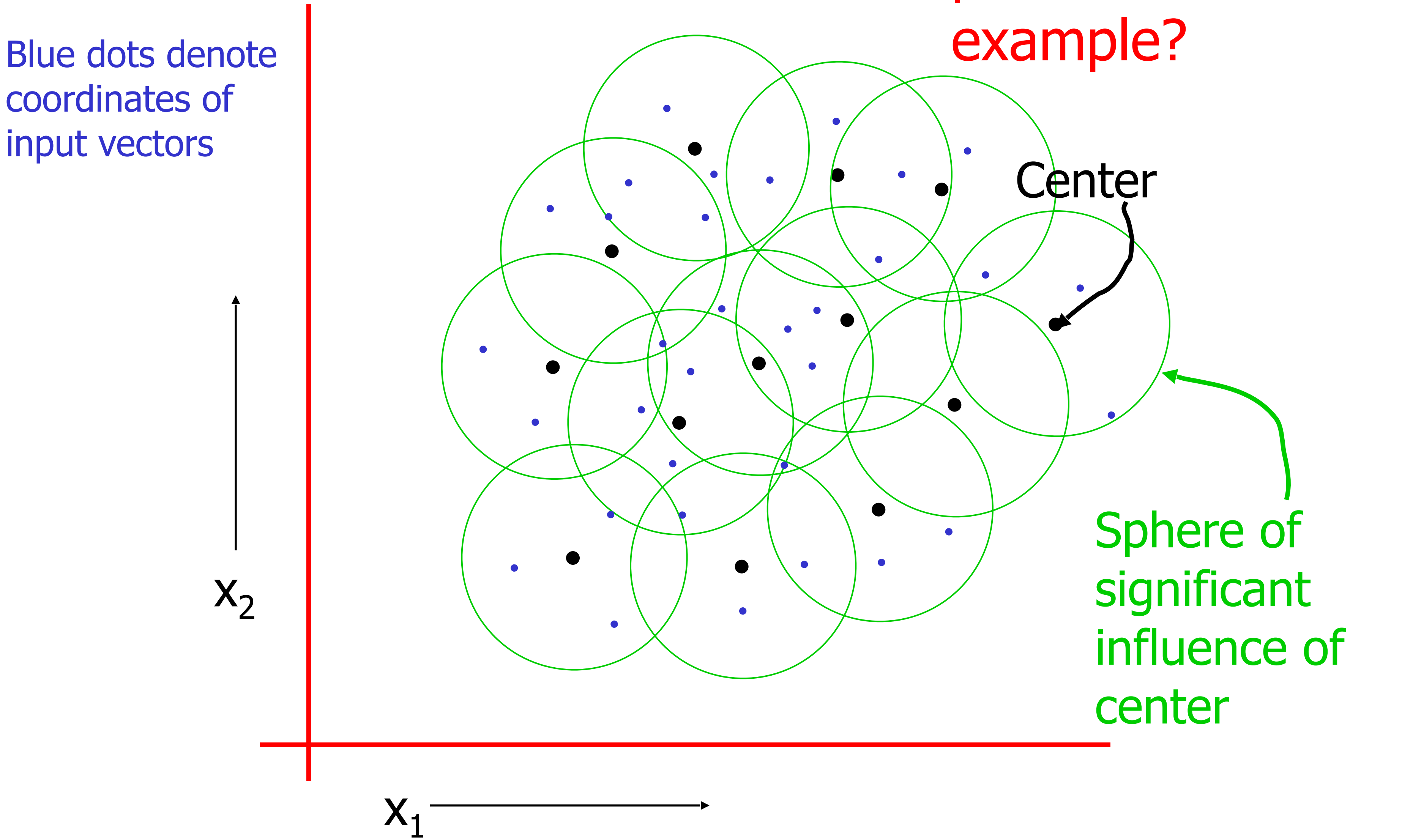
Happy RBFs in 2-d

Blue dots denote coordinates of input vectors



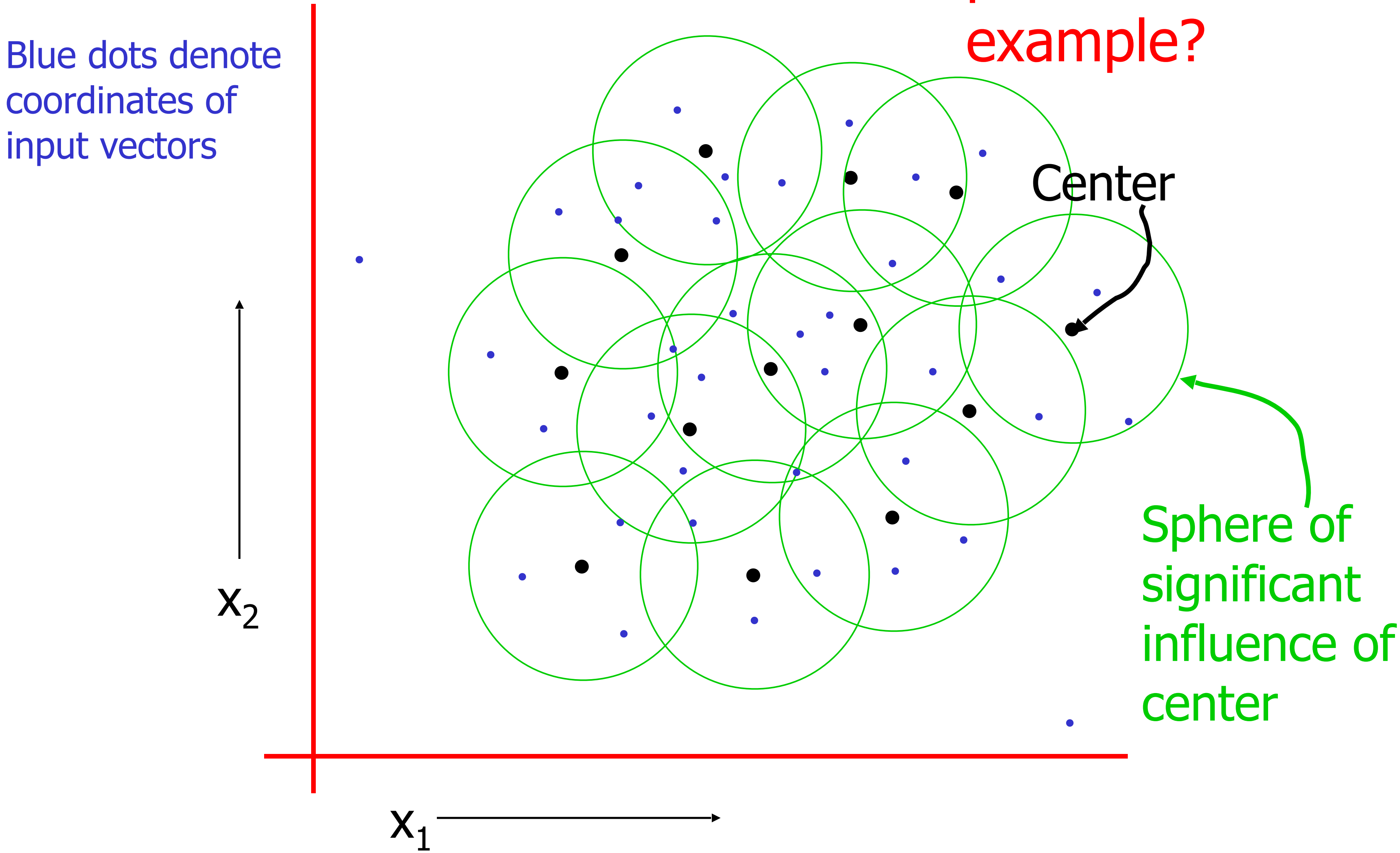
Crabby RBFs in 2-d

What's the problem in this example?



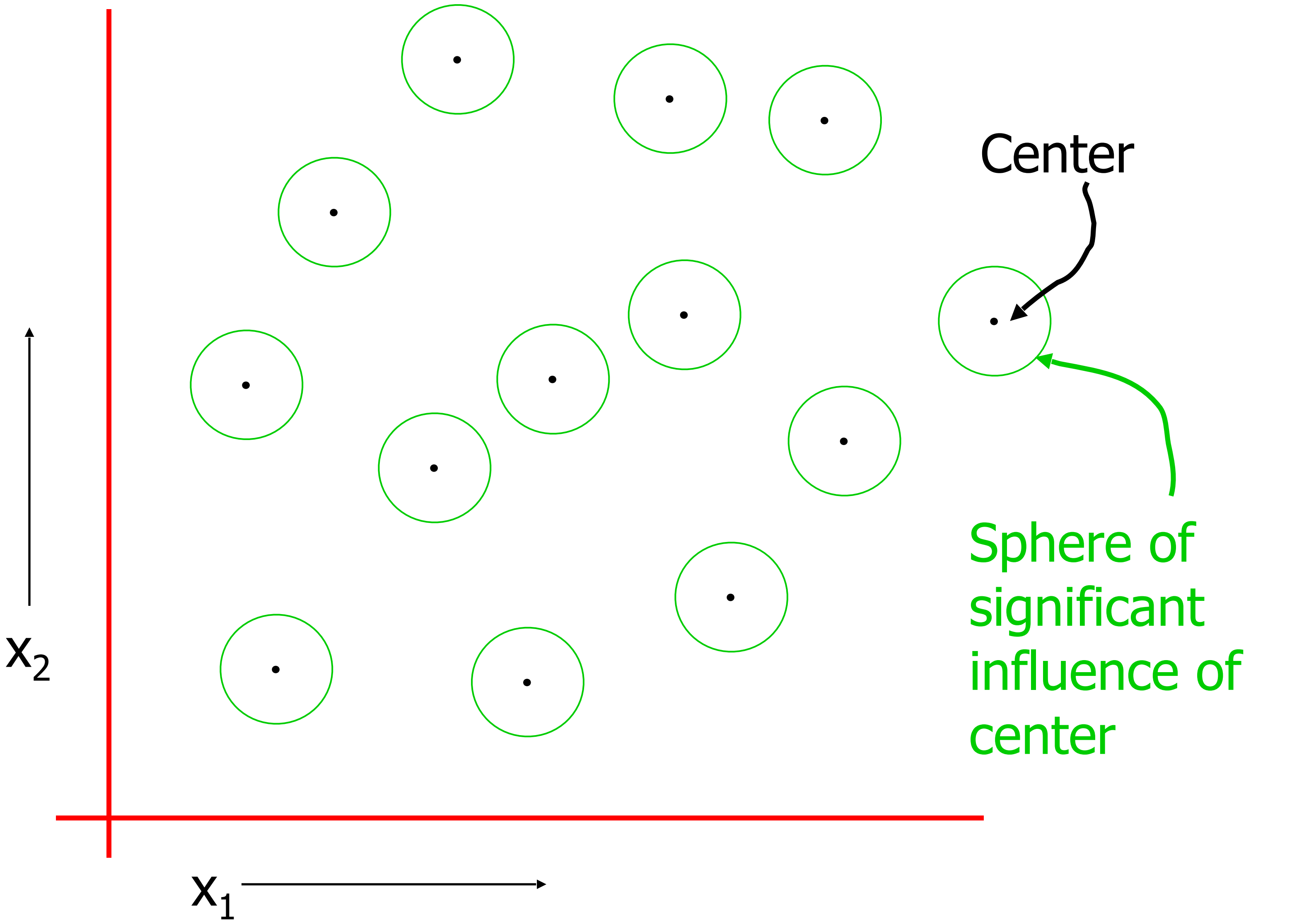
More crabby RBFs

And what's the problem in this example?



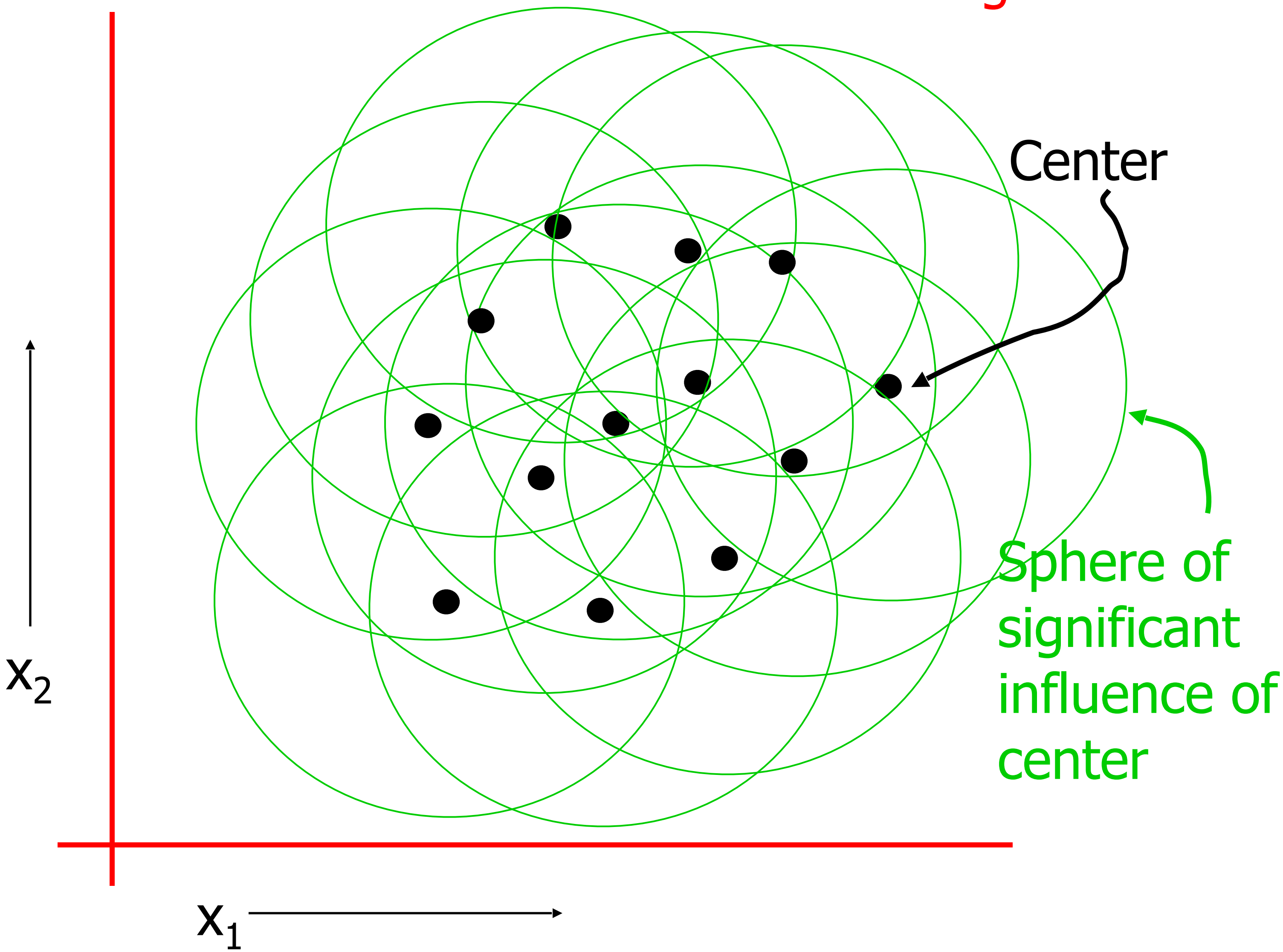
Hopeless!

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



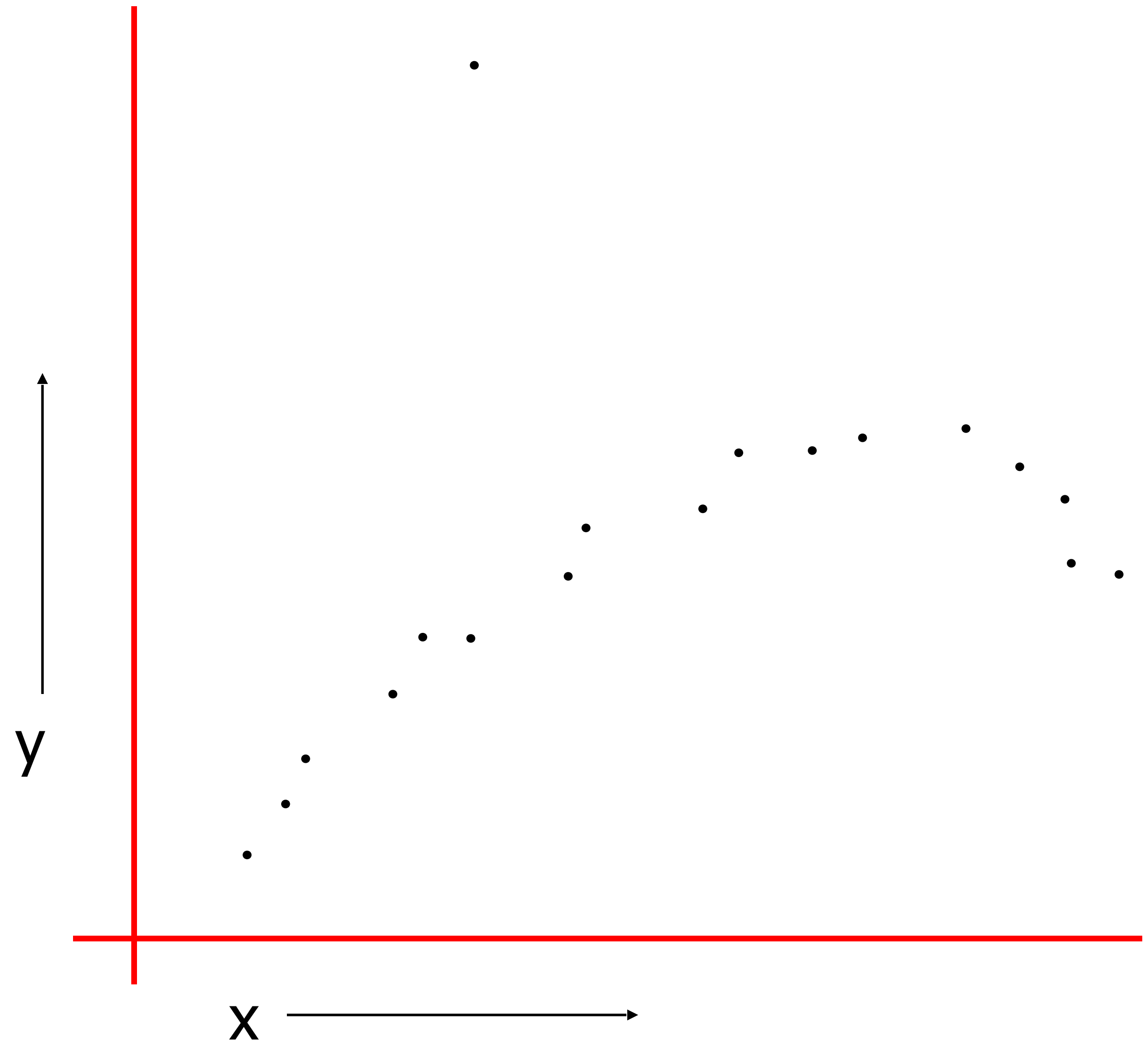
Unhappy

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

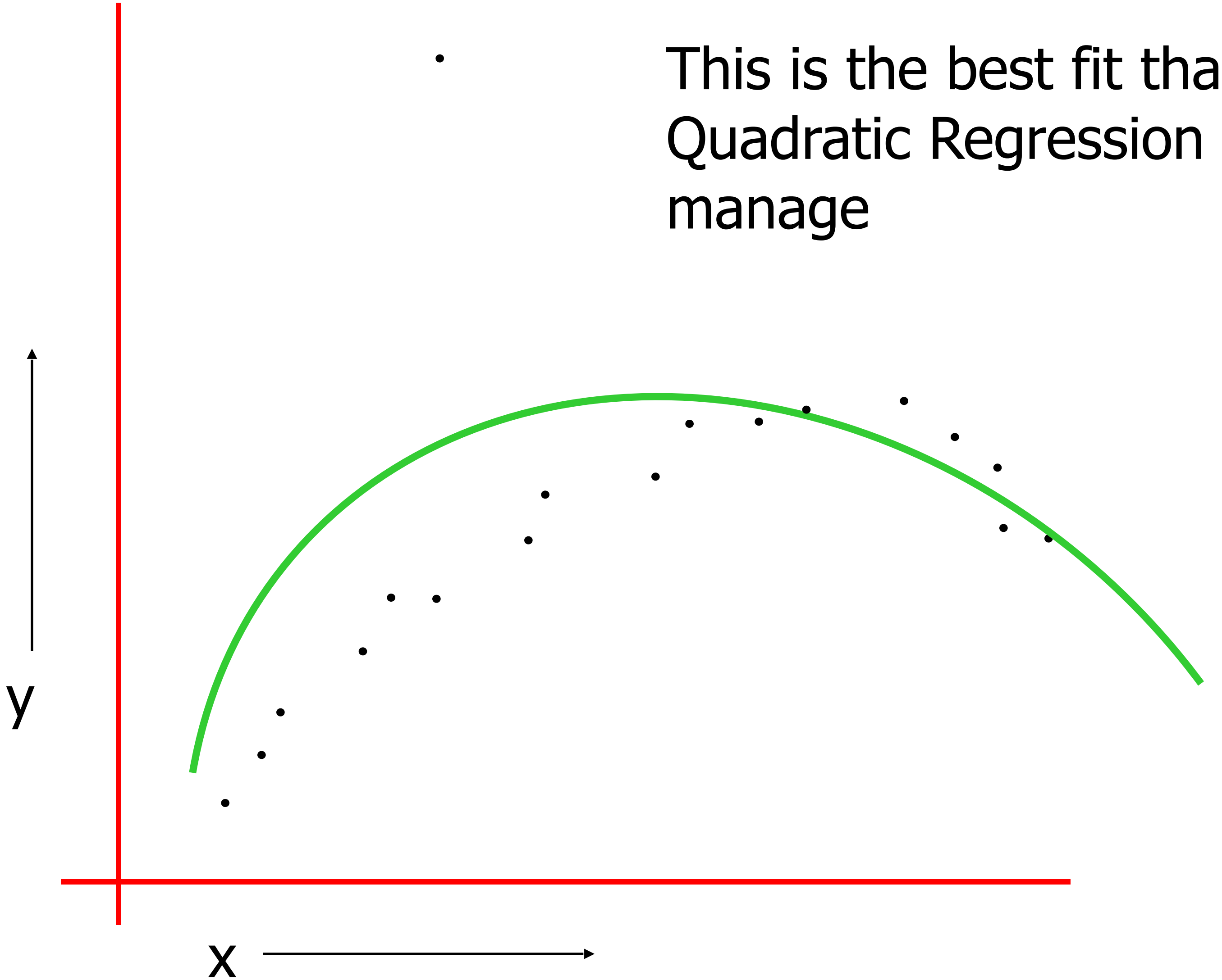


Robust Regression

Robust Regression

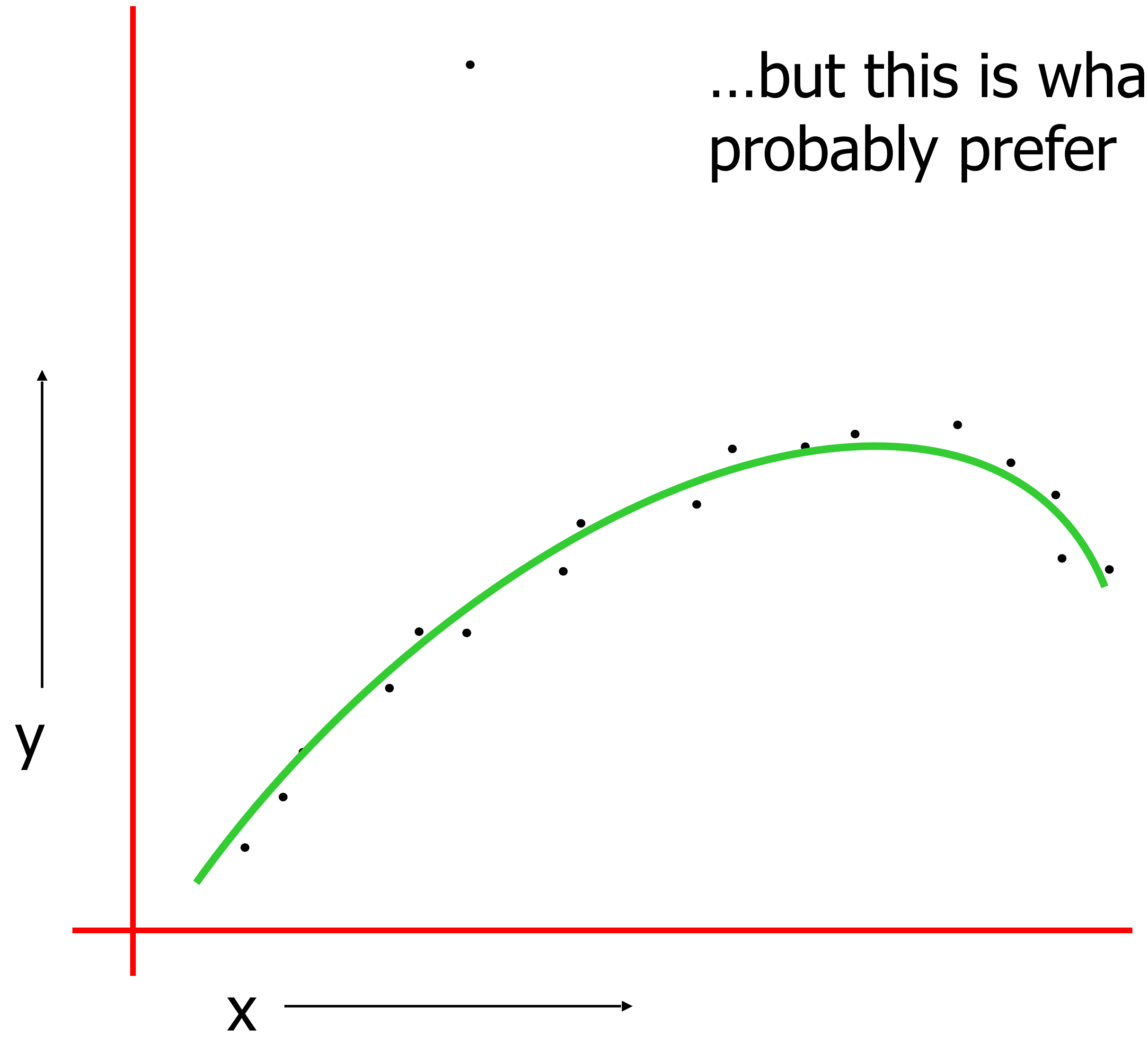


Robust Regression



This is the best fit that Quadratic Regression can manage

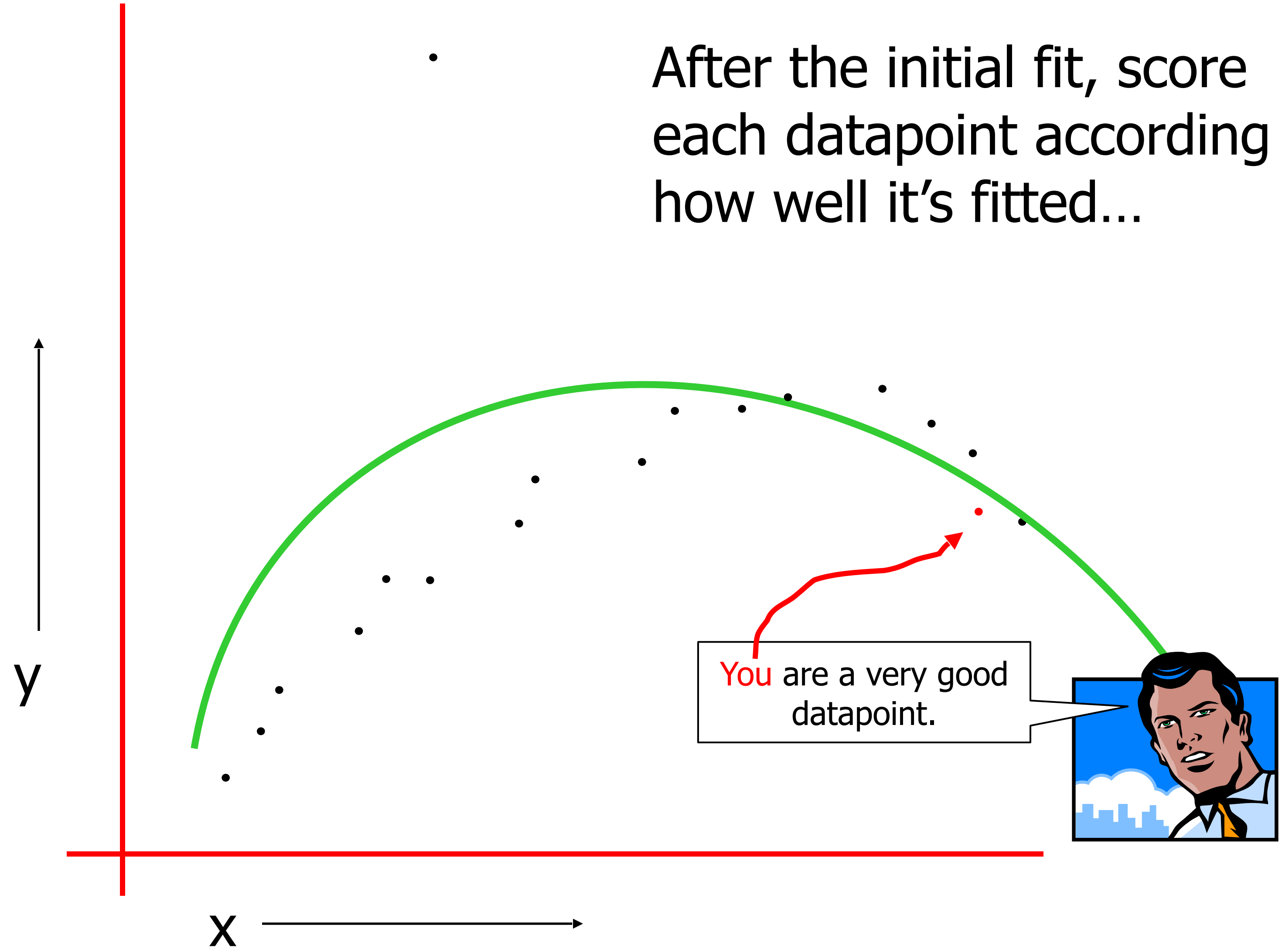
Robust Regression



…but this is what we'd probably prefer

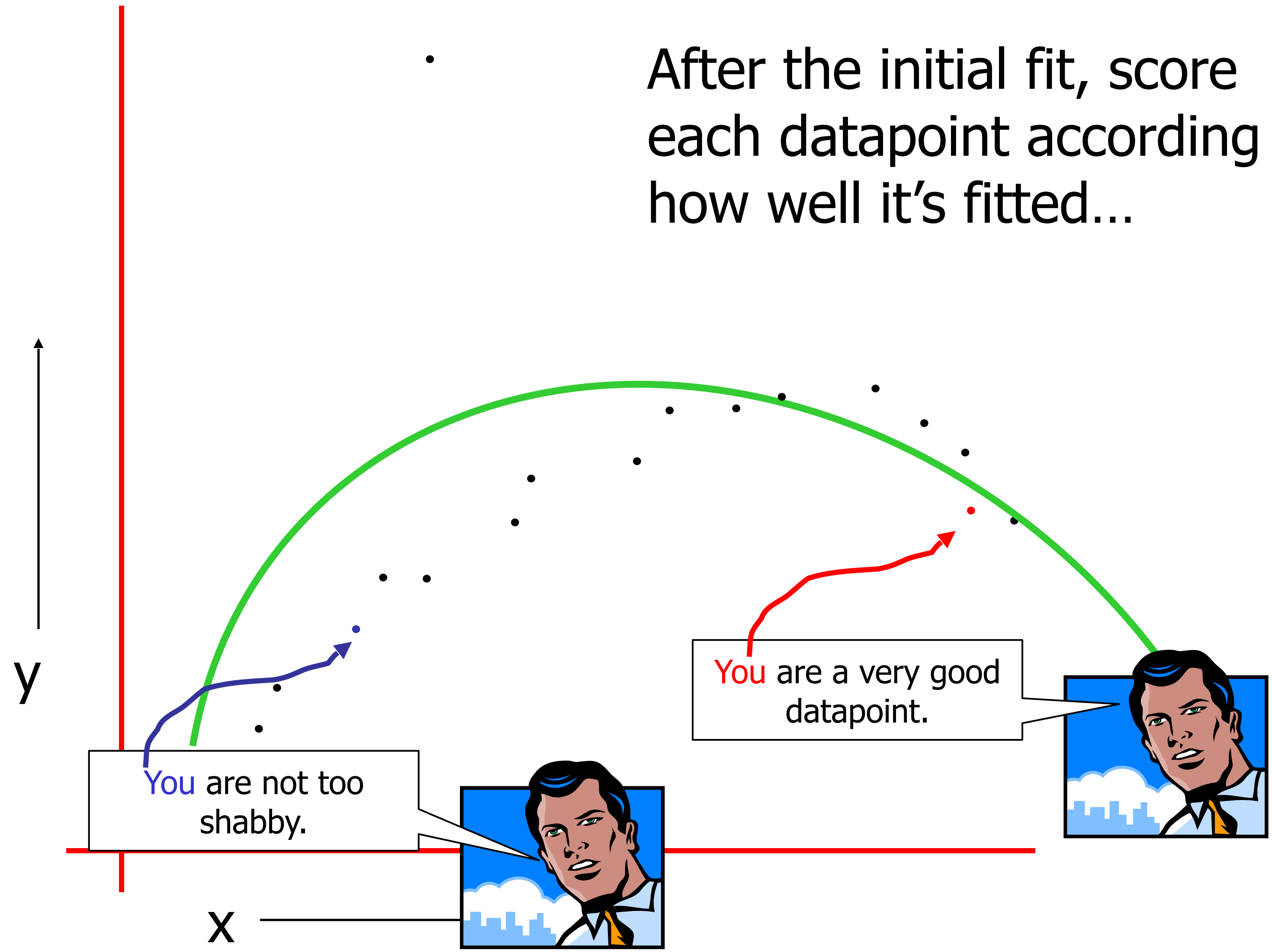
LOESS-based Robust Regression

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



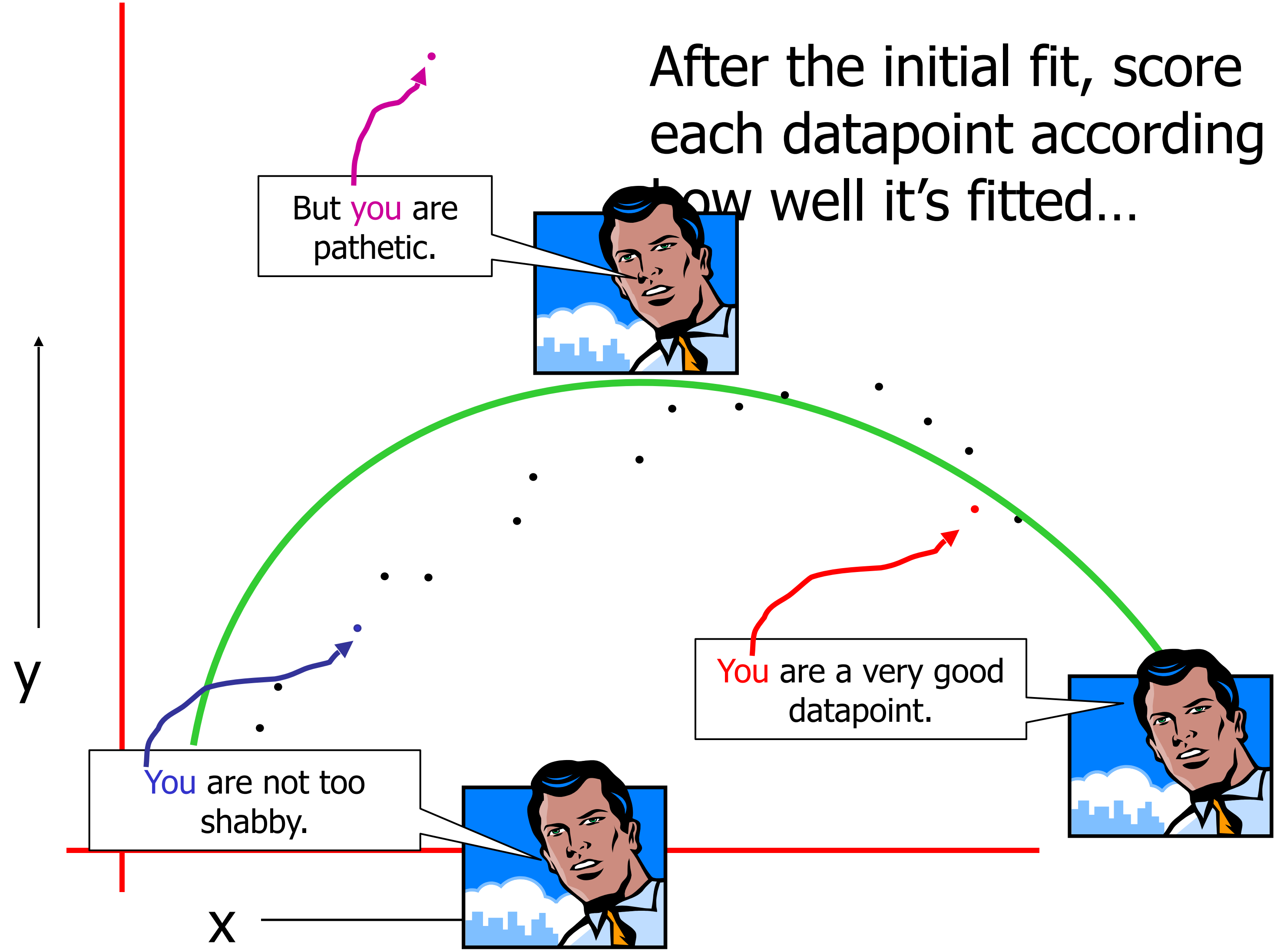
LOESS-based Robust Regression

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

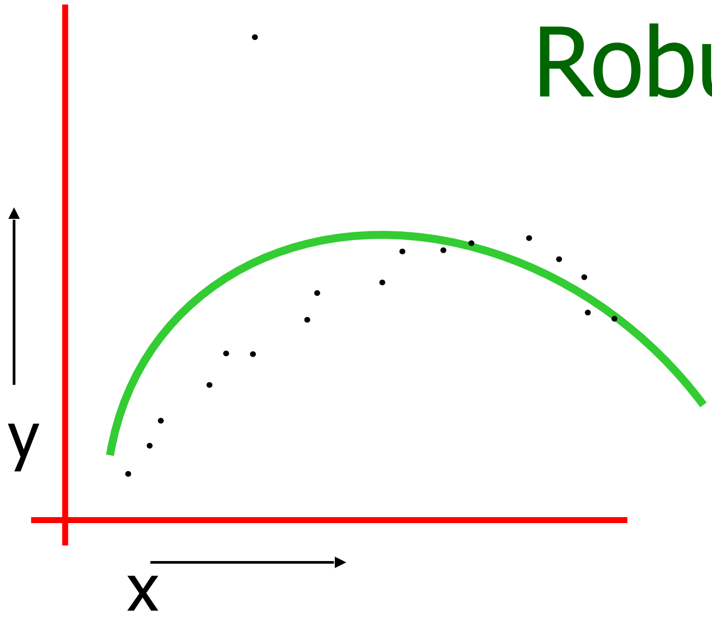


LOESS-based Robust Regression

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



Robust Regression

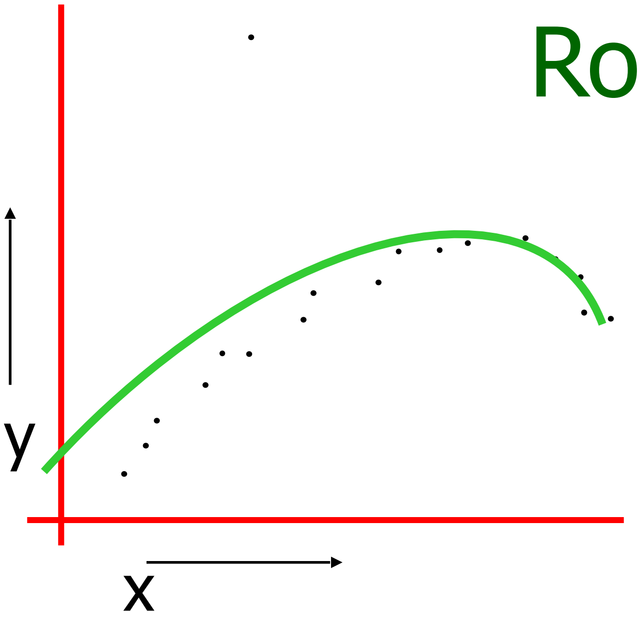


For k = 1 to R…

- 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:

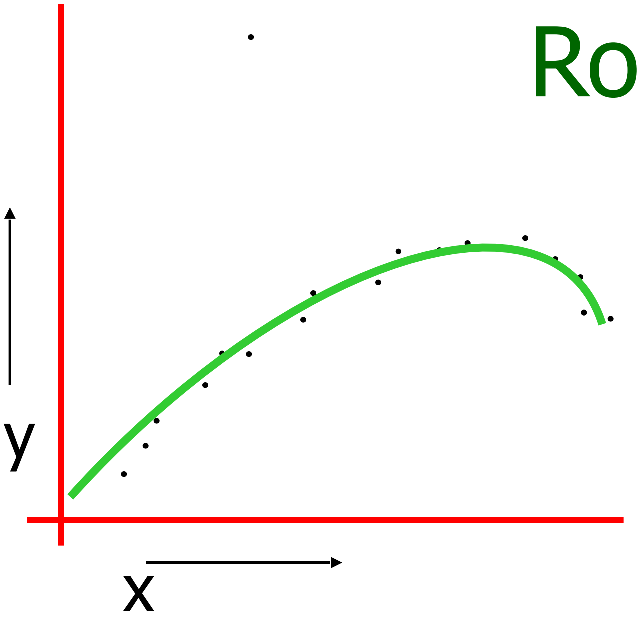
$$w_k = \text{KernelFn}([y_k - y^{est}_k]^2)$$

Then redo the regression using weighted datapoints.

Weighted regression was described earlier in the "vary noise" section, and is also discussed in the "Memory-based Learning" Lecture.

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.

Regression Trees

Regression Trees

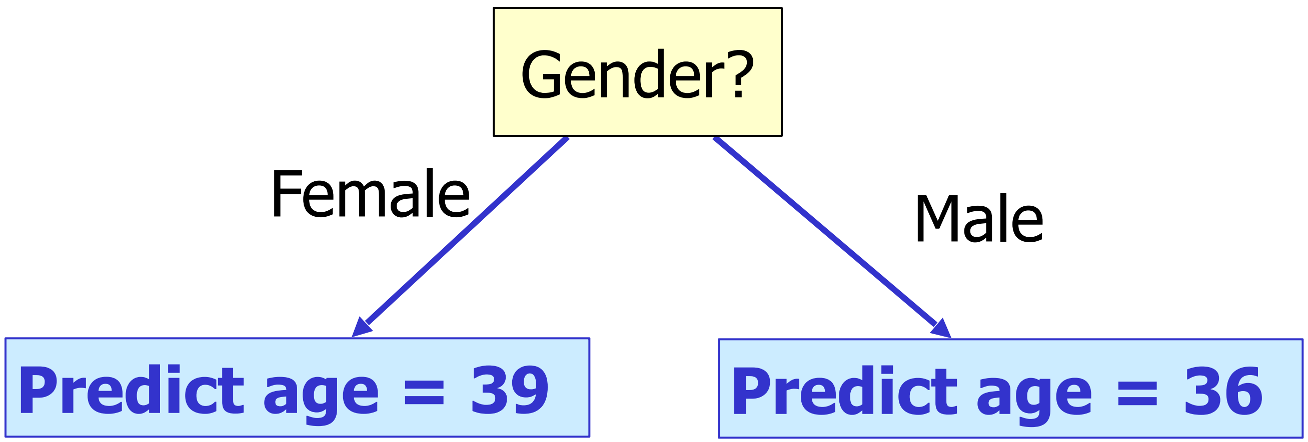
- "Decision trees for regression"

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

…property-owner = Yes

Also known as
“M l Trees”

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

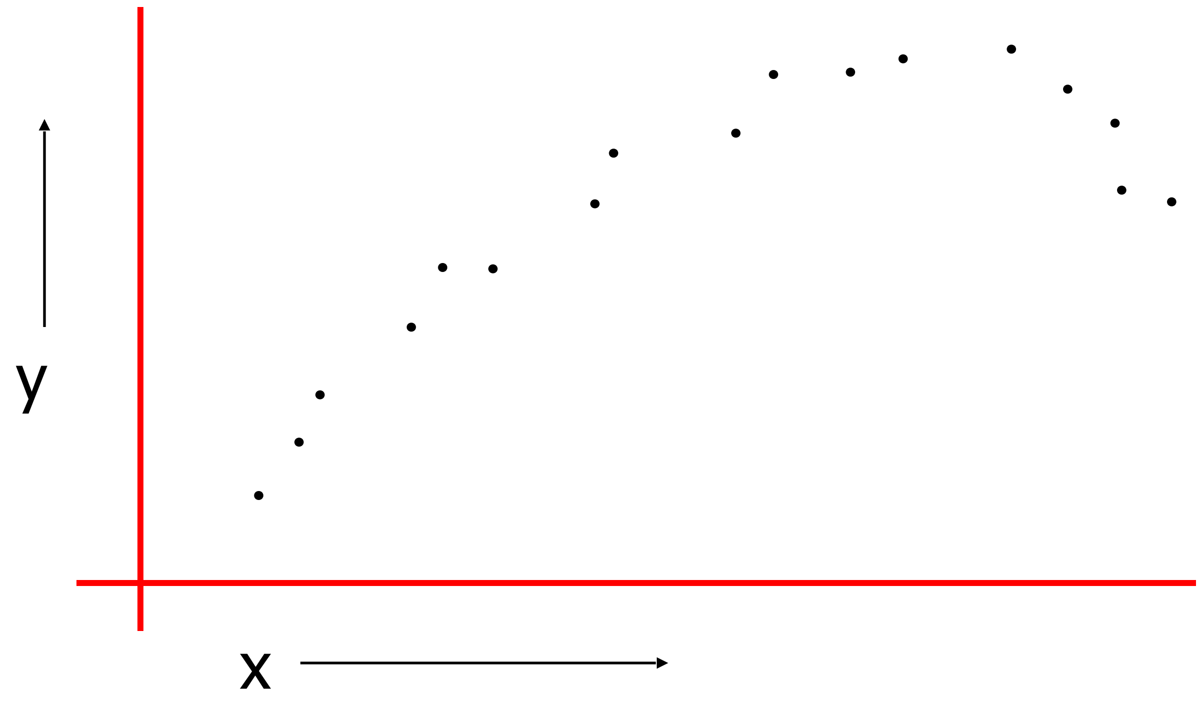
Leaves contain
functions (traine
linear regression
records matching t leaf)

tribute chosen to minimize
SE of regressed children.

Pruning with a different Chi-squared

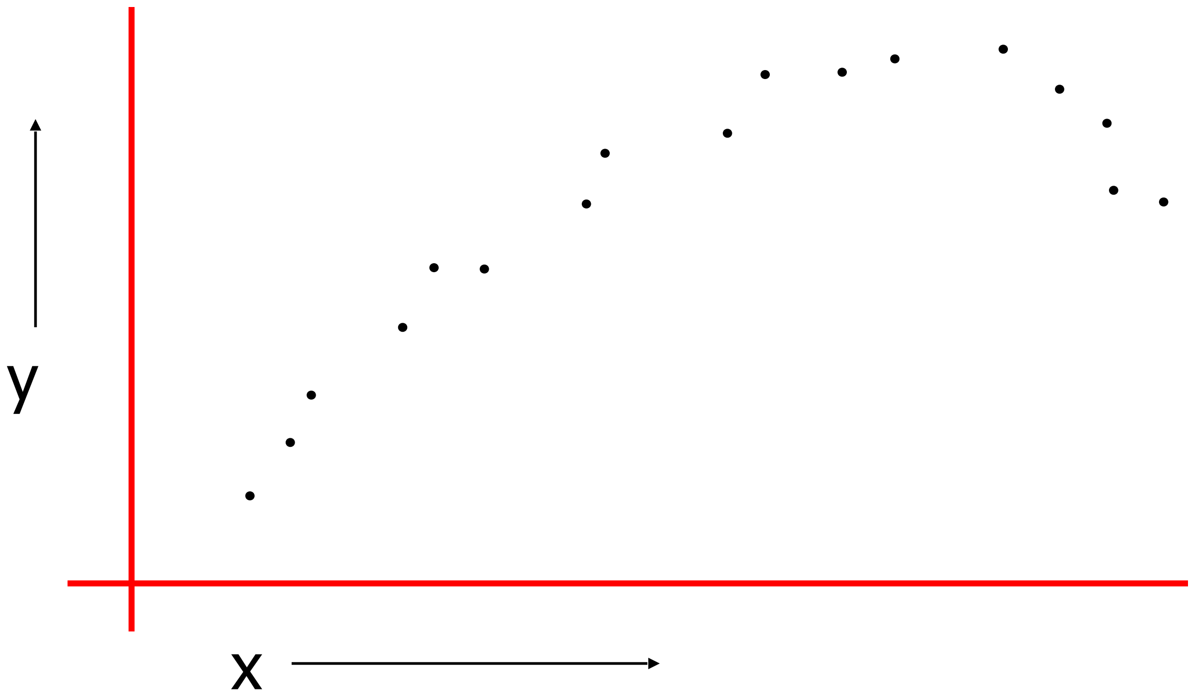
Test your understanding

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



Test your understanding

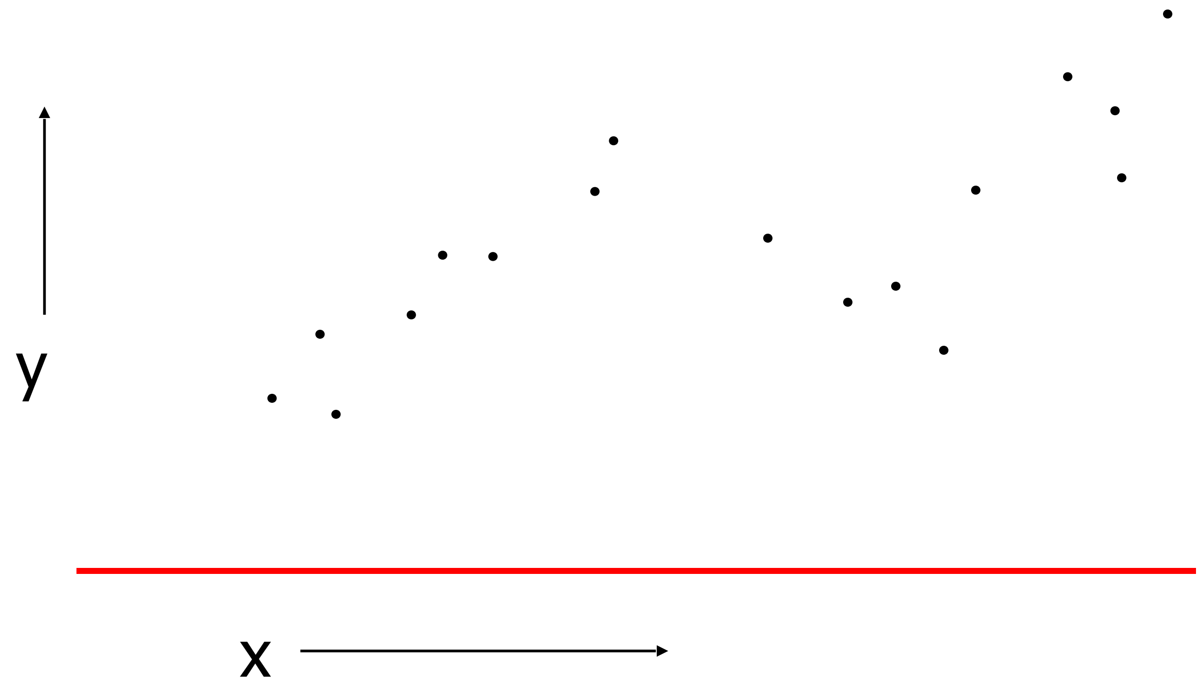
Assuming linear regression trees, can you sketch a graph of the fitted function $y^{est}(x)$ over this diagram?



Multilinear Interpolation

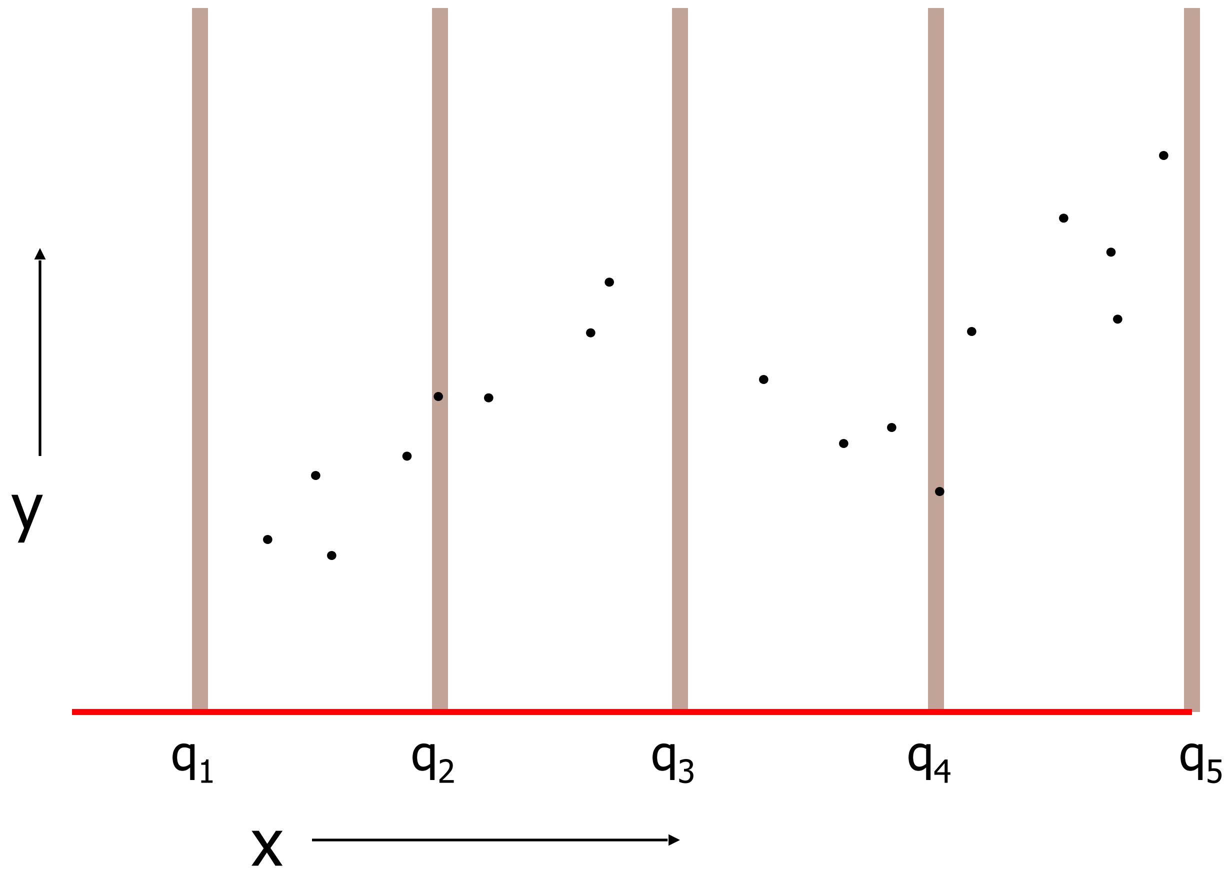
Multilinear Interpolation

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



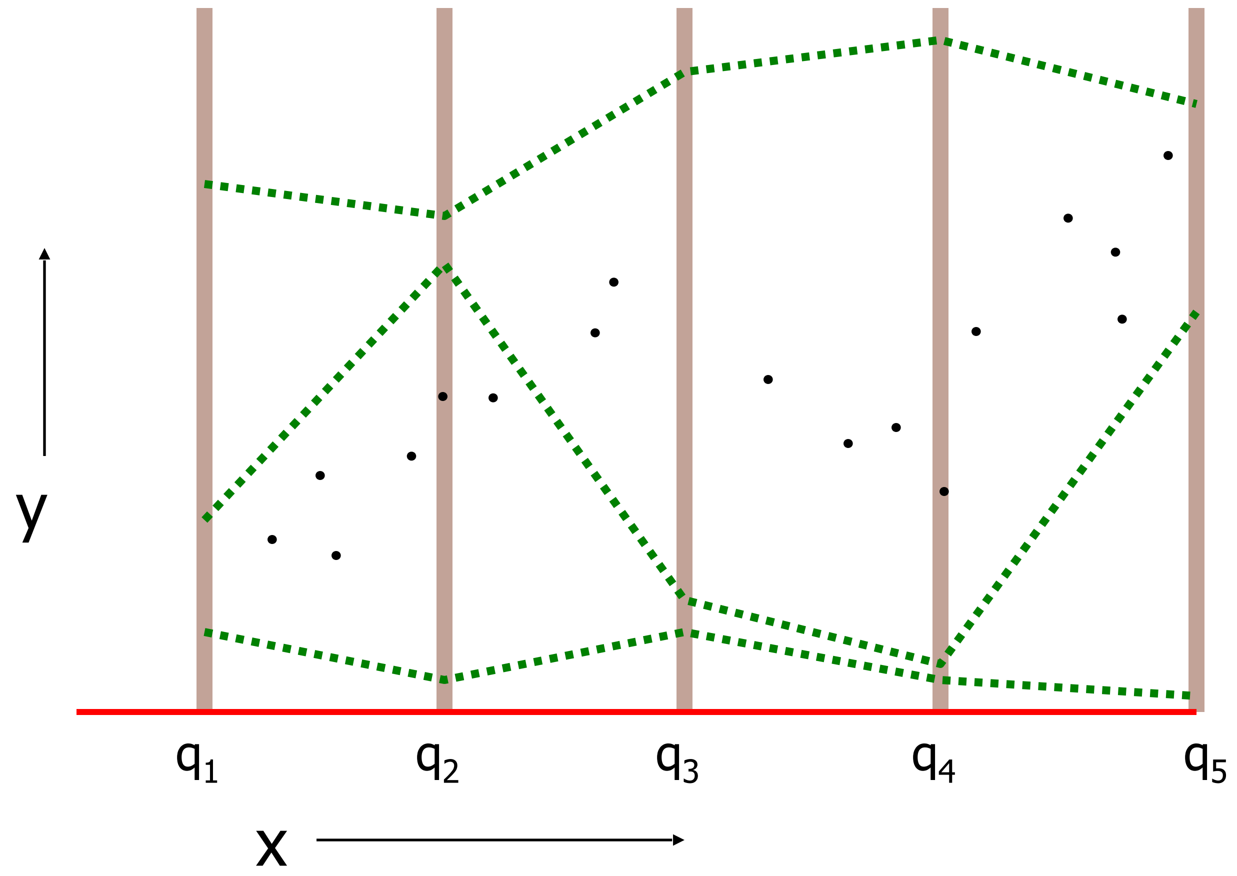
Multilinear Interpolation

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



Multilinear Interpolation

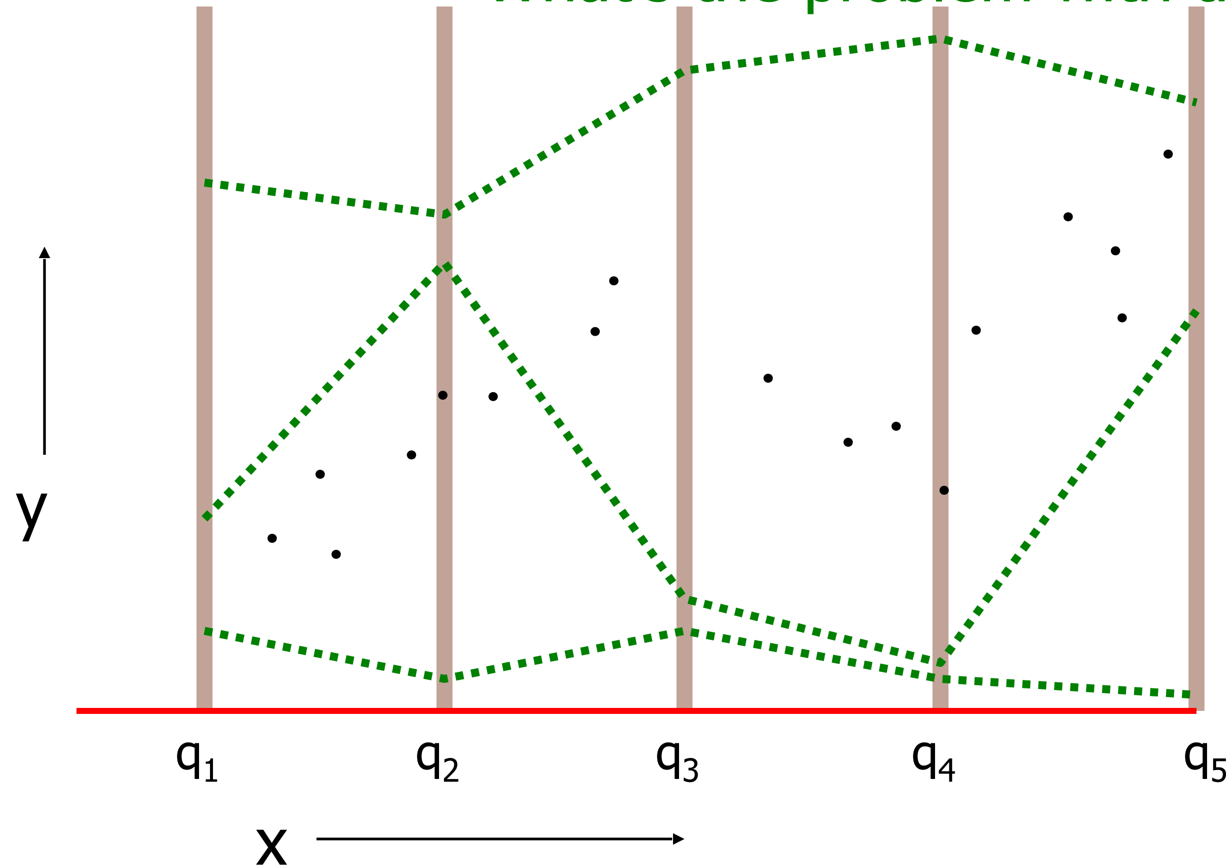
We are going to assume the data was generated by a noisy version of a function that can only bend at the knots. Here are 3 examples (none fits the data well)



How to find the best fit?

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

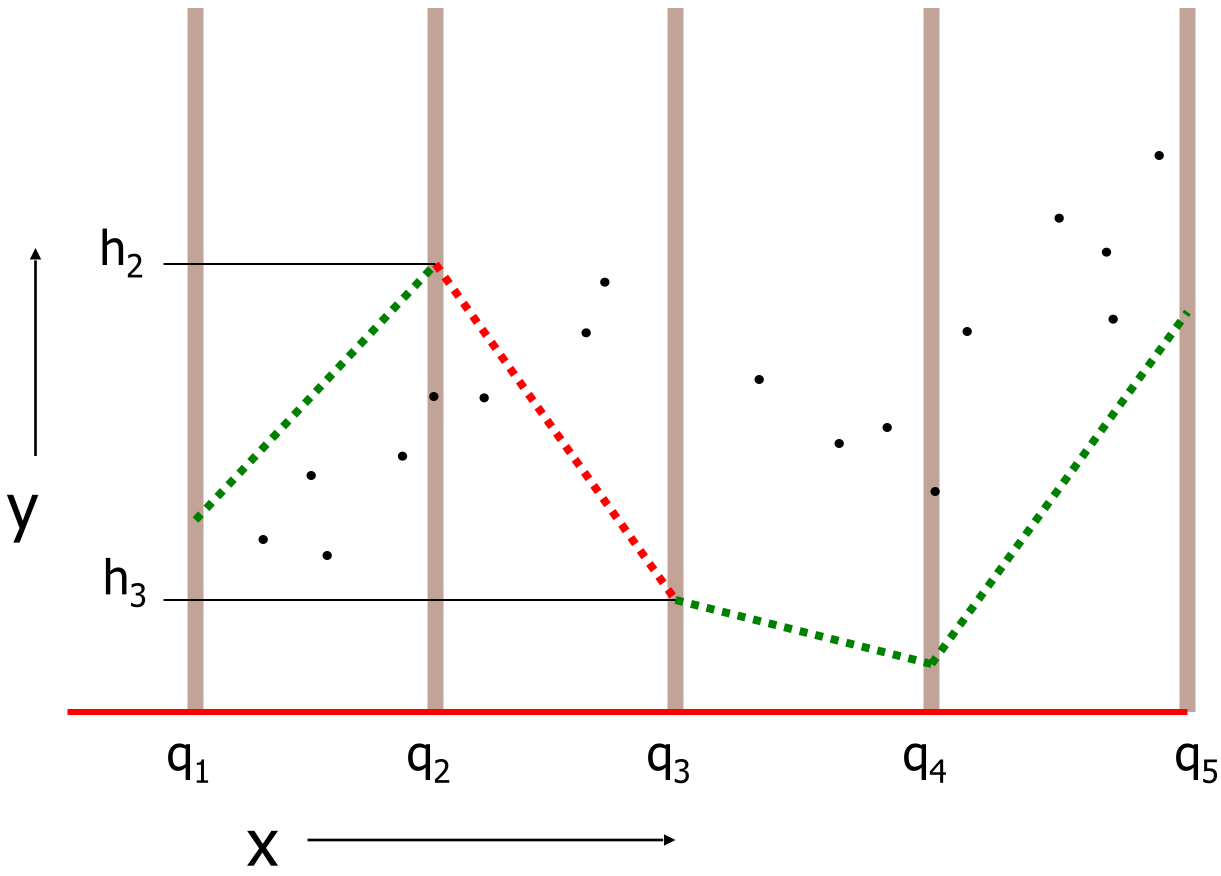
What's the problem with this idea?



How to find the best fit?

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

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

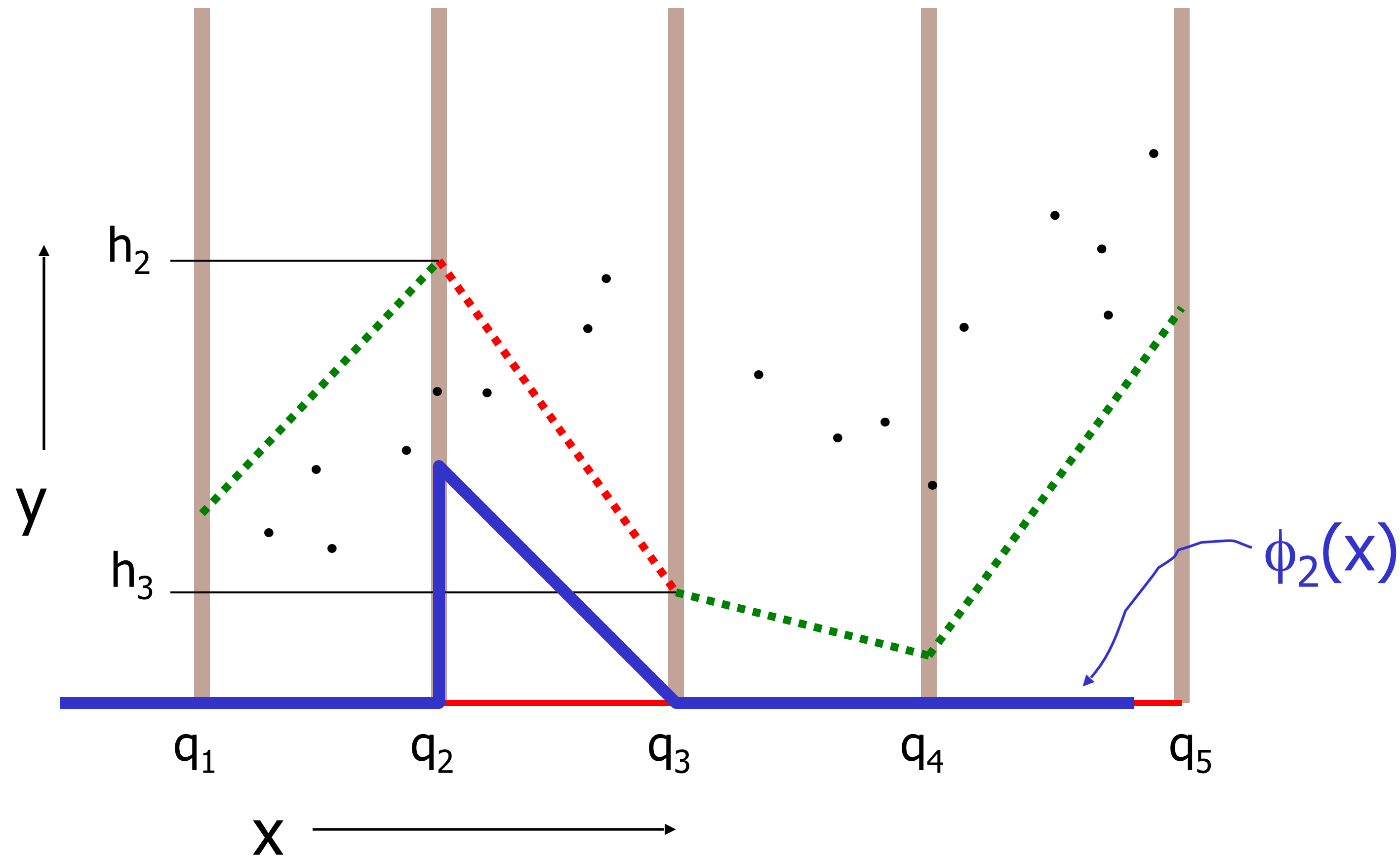


How to find the best fit?

In the red segment…

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

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

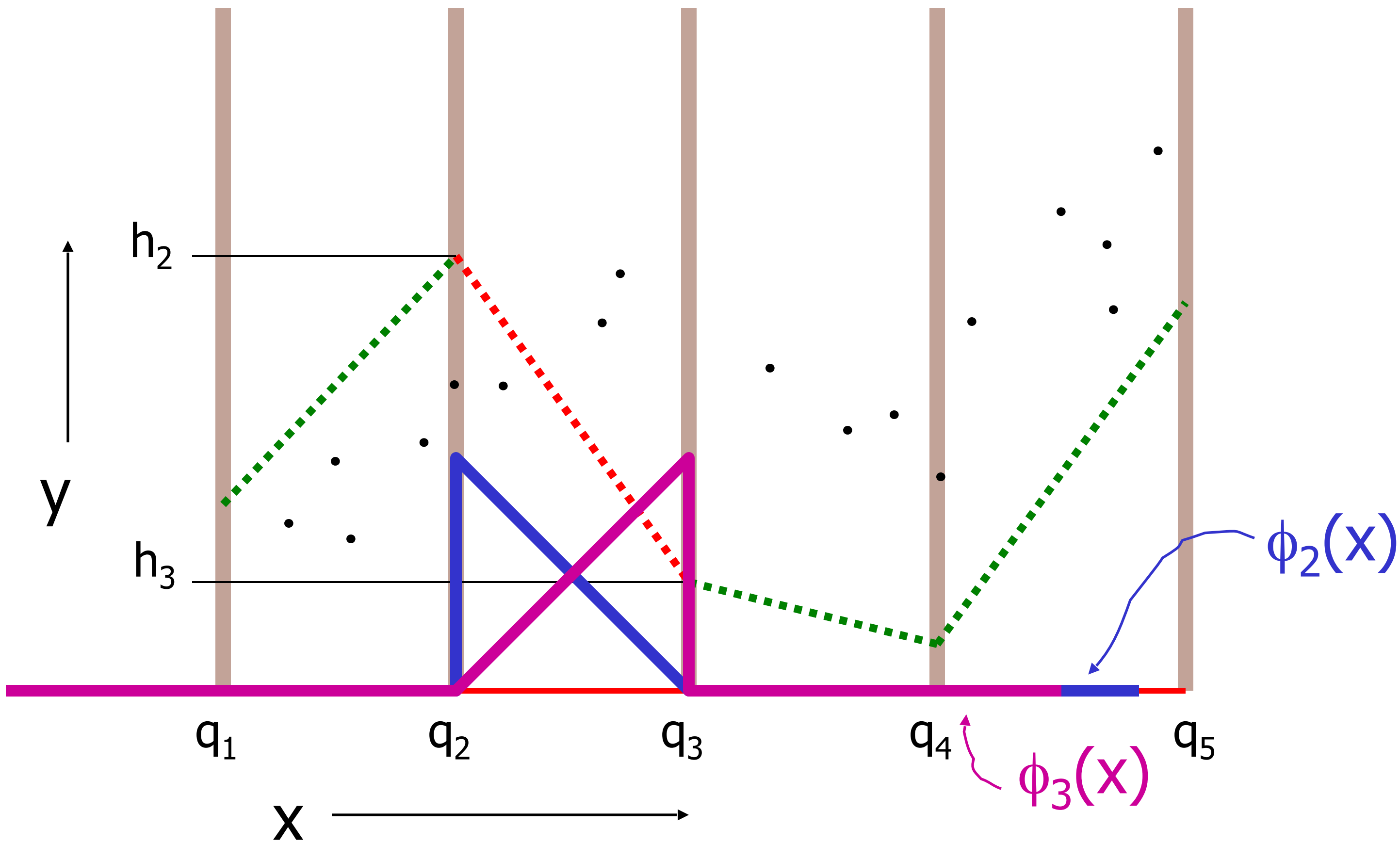


How to find the best fit?

In the red segment…

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

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

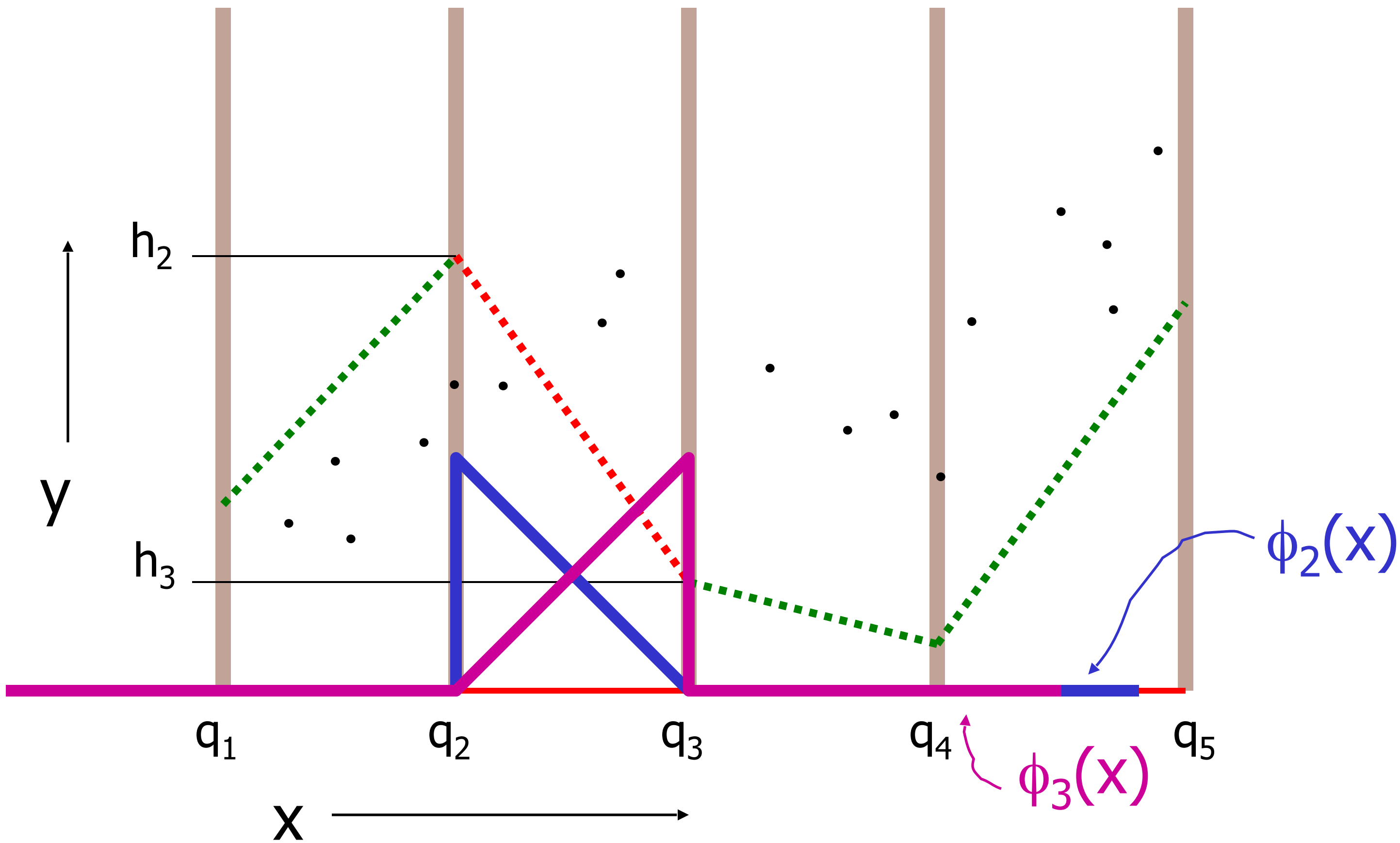


How to find the best fit?

In the red segment…

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

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

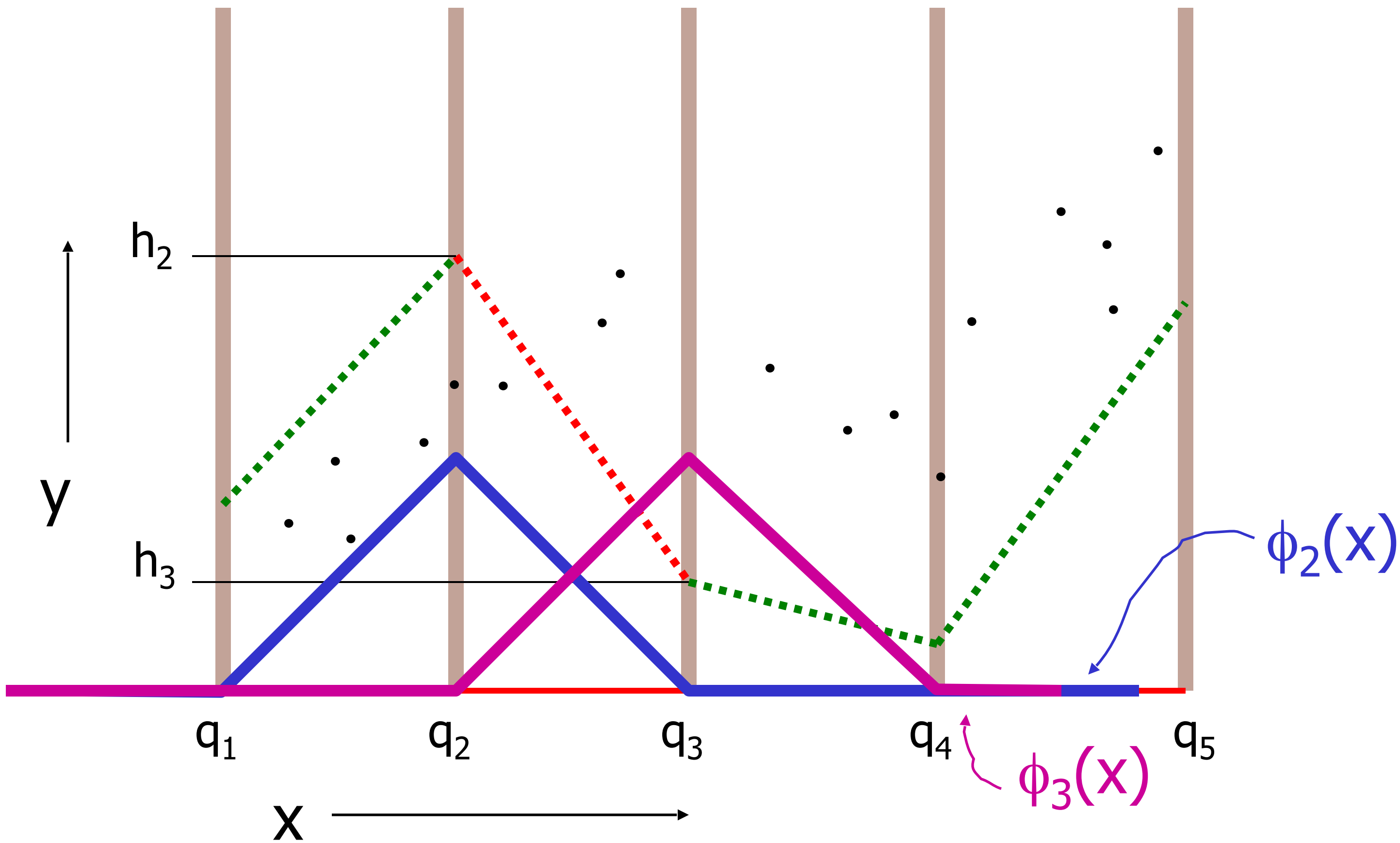


How to find the best fit?

In the red segment…

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

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

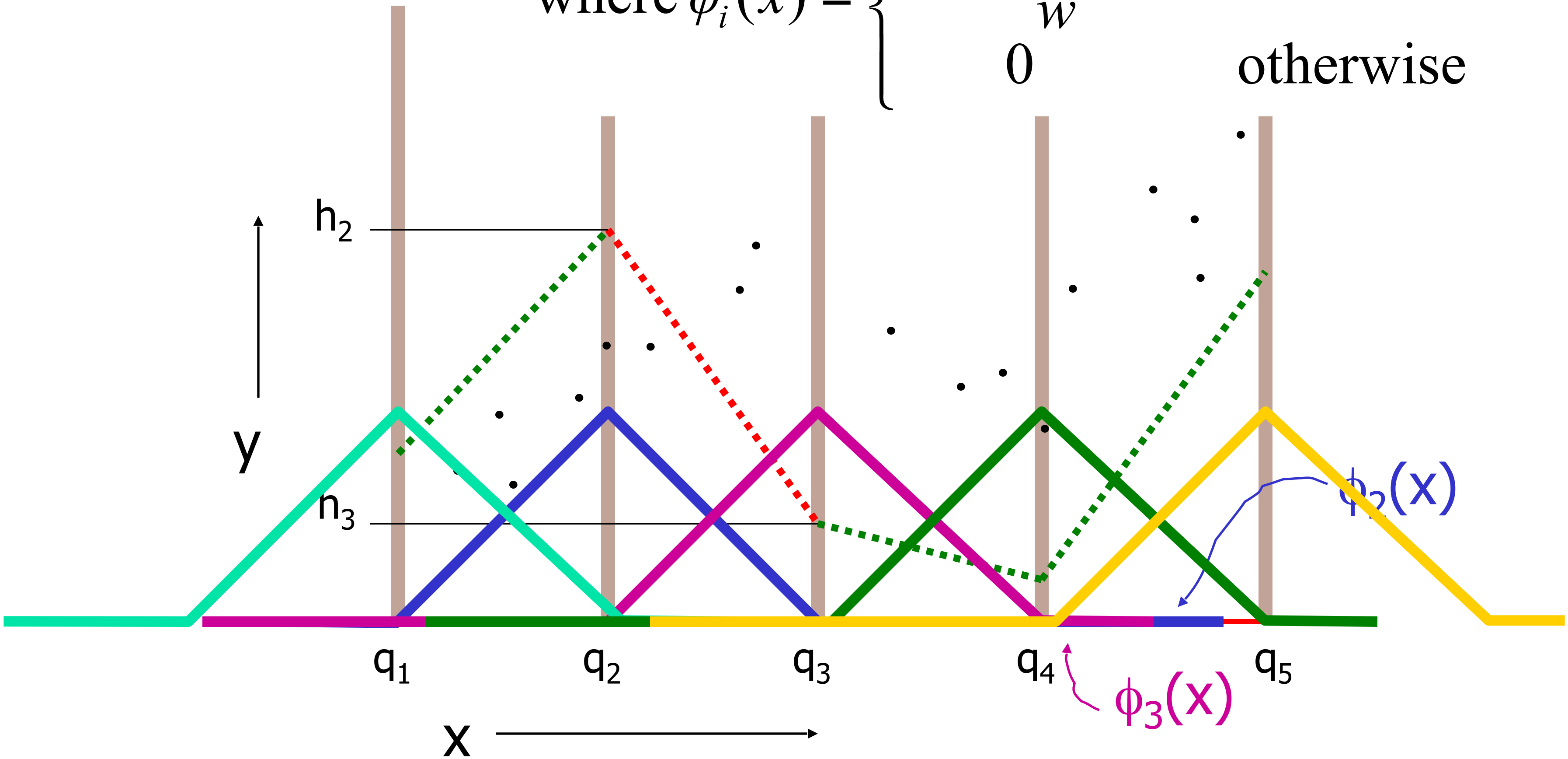


How to find the best fit?

In **general**

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

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

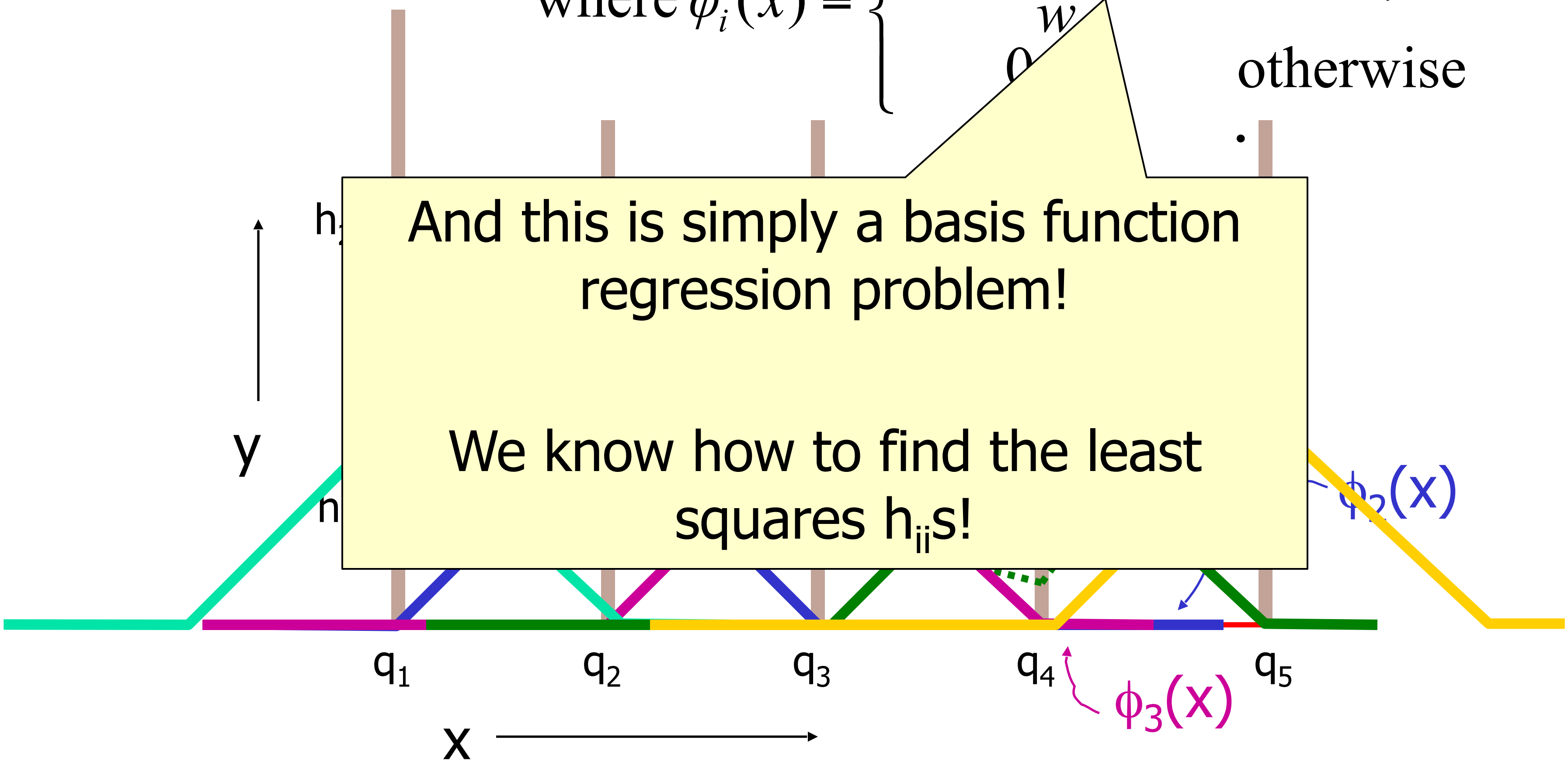


How to find the best fit?

In **general**

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

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

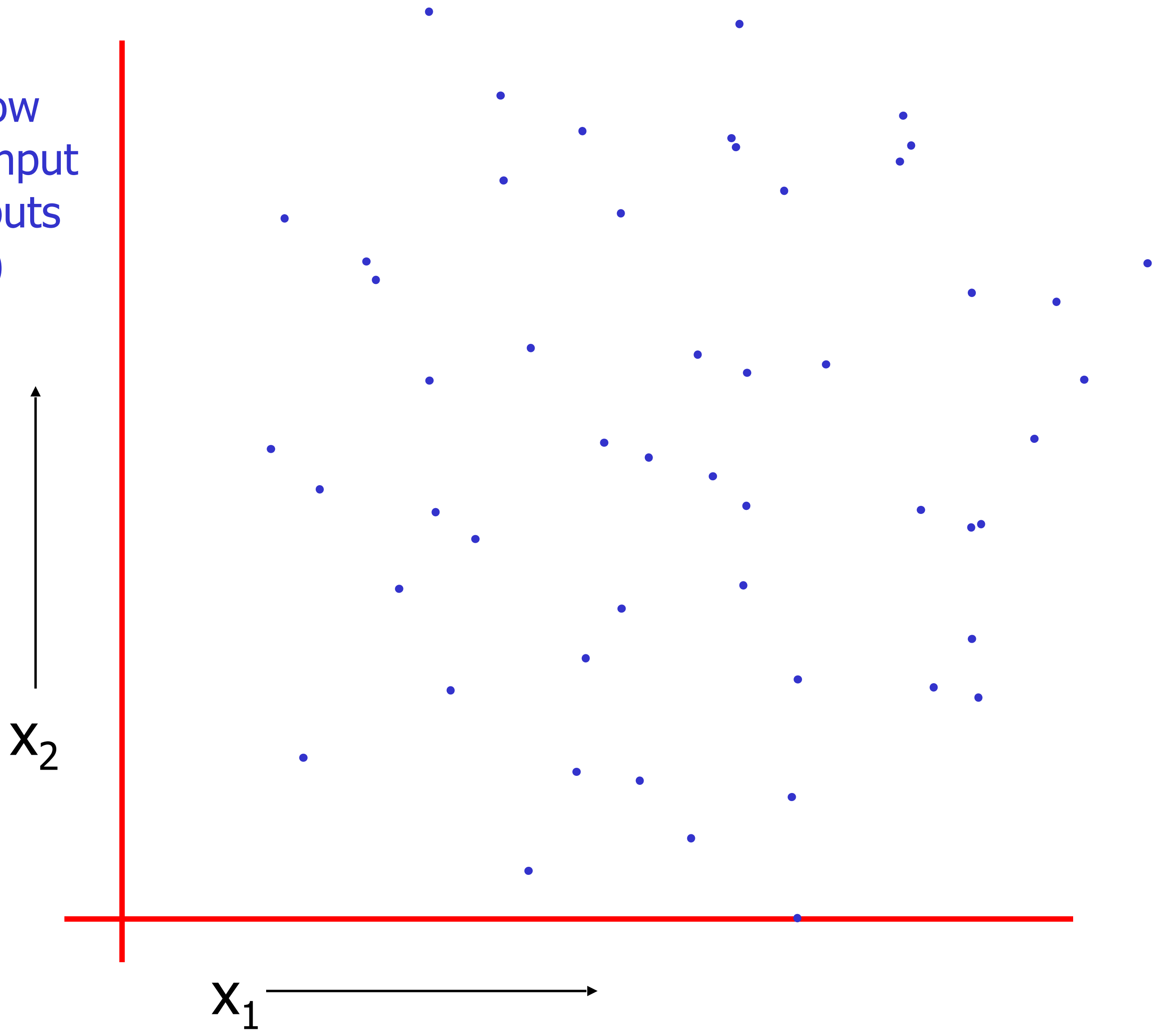


And this is simply a basis function regression problem!

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

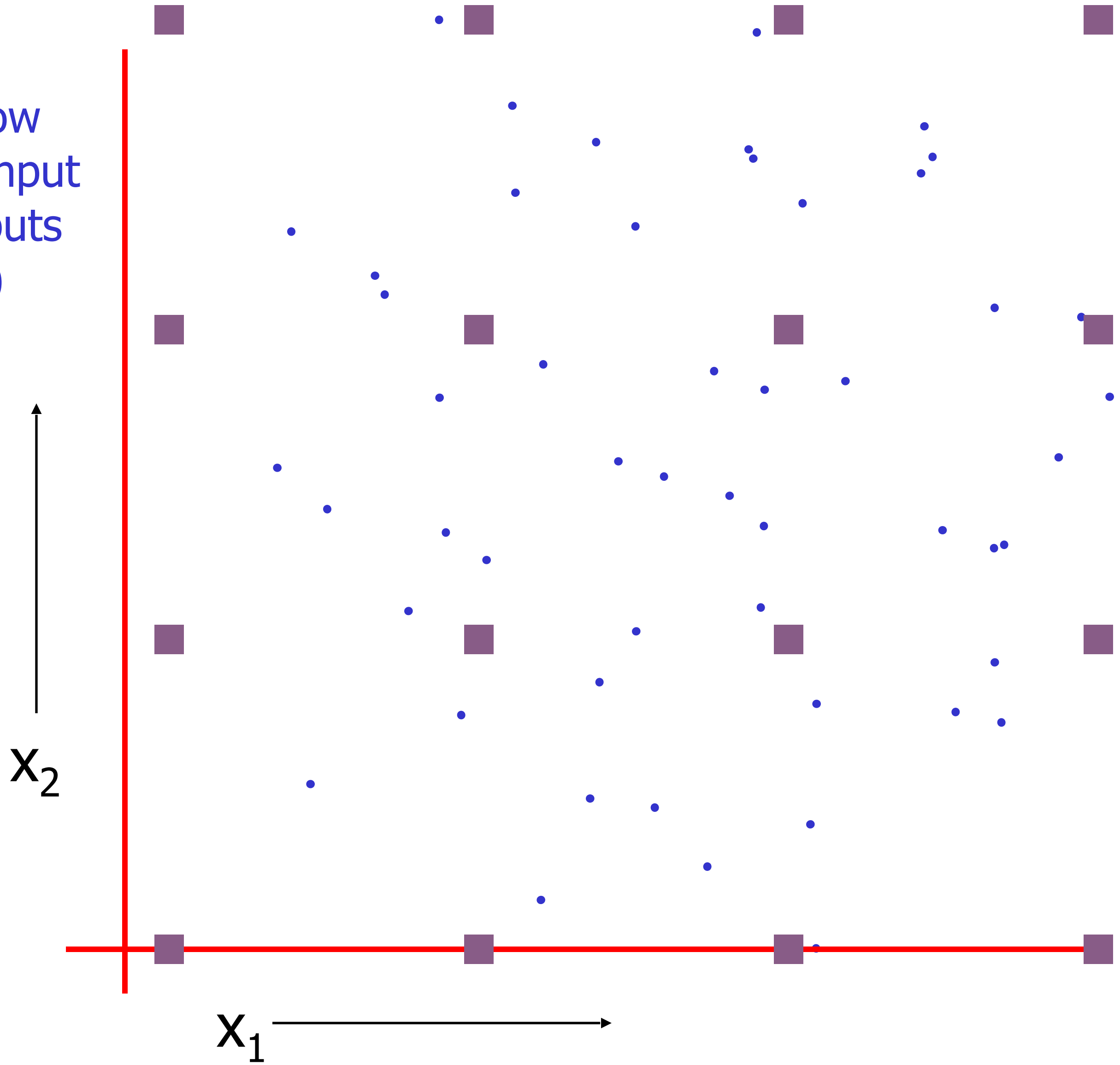
In two dimensions…

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



In two dimensions…

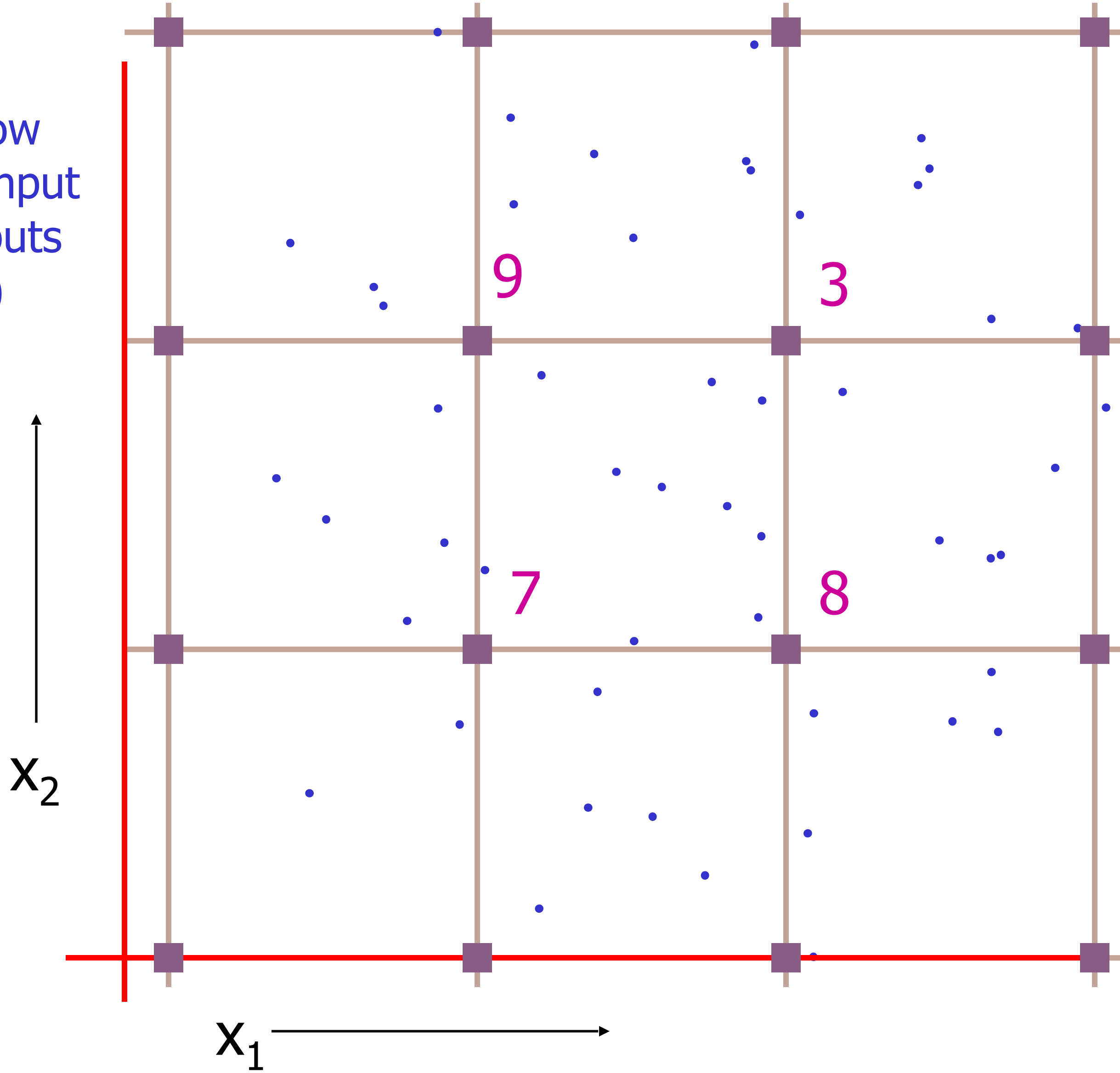
Blue dots show locations of input vectors (outputs not depicted)



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

In two dimensions…

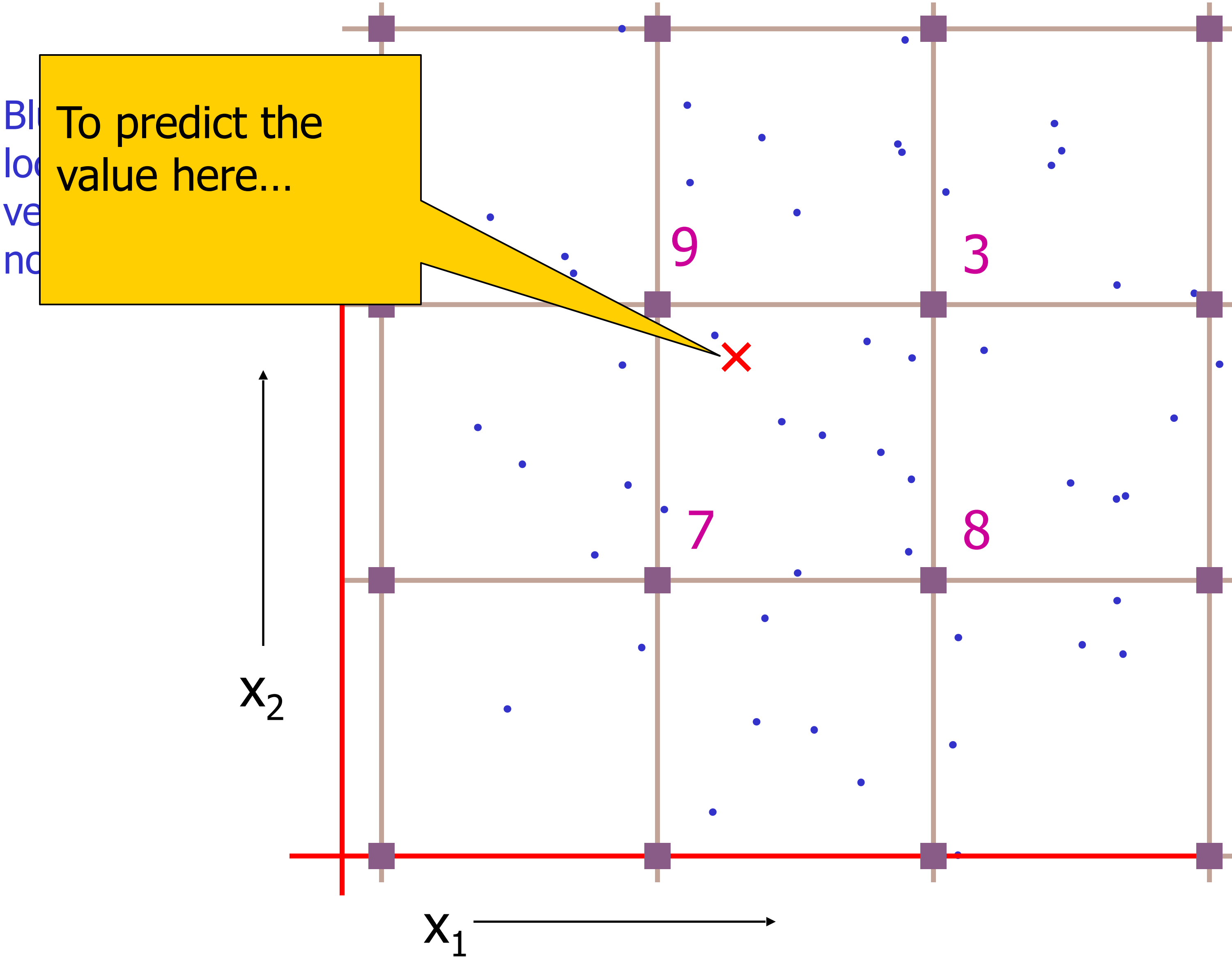
Blue dots show locations of input vectors (outputs not depicted)



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…

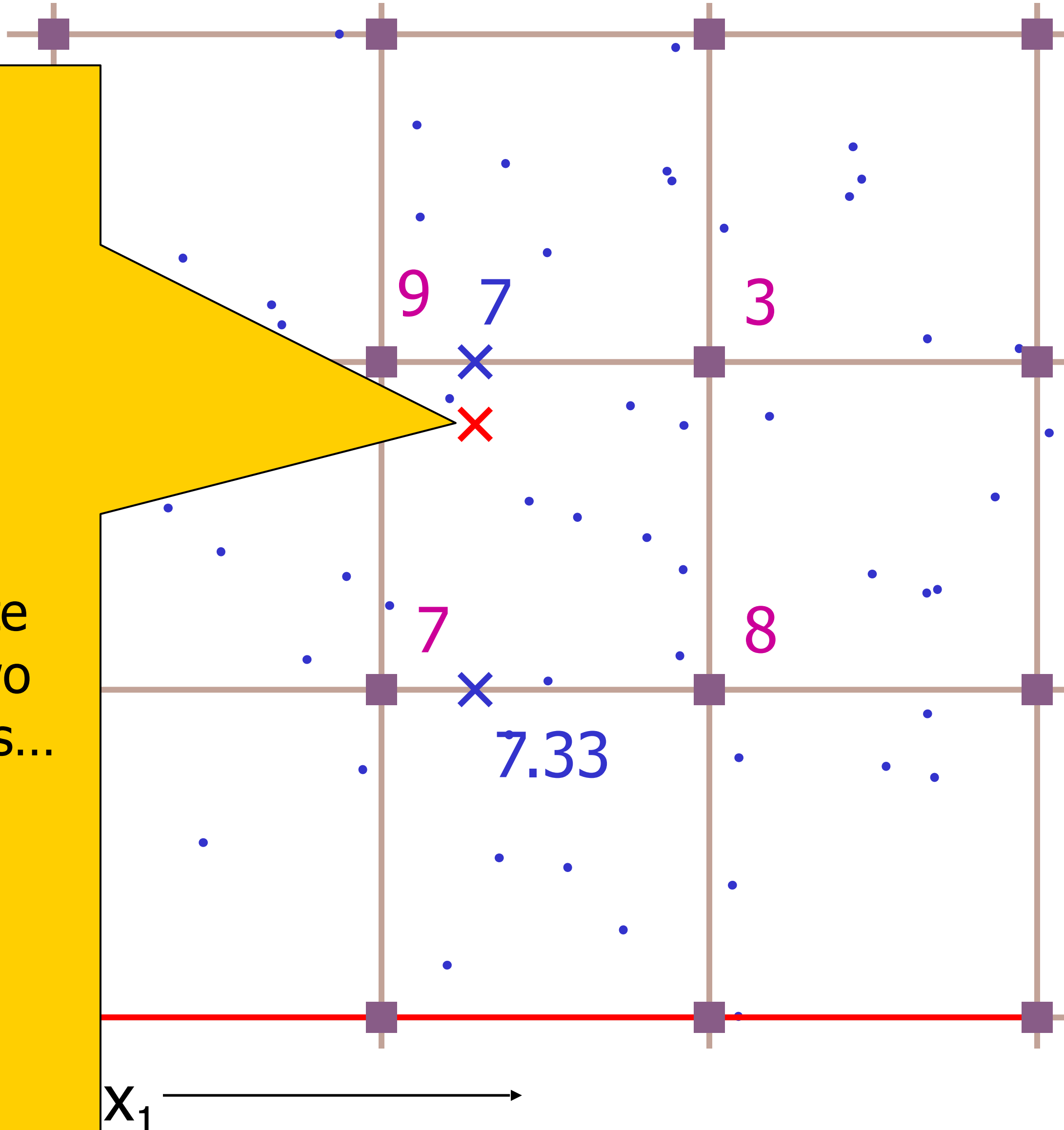


Each purple dot is a knot point. It will contain the height of the estimated surface

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…

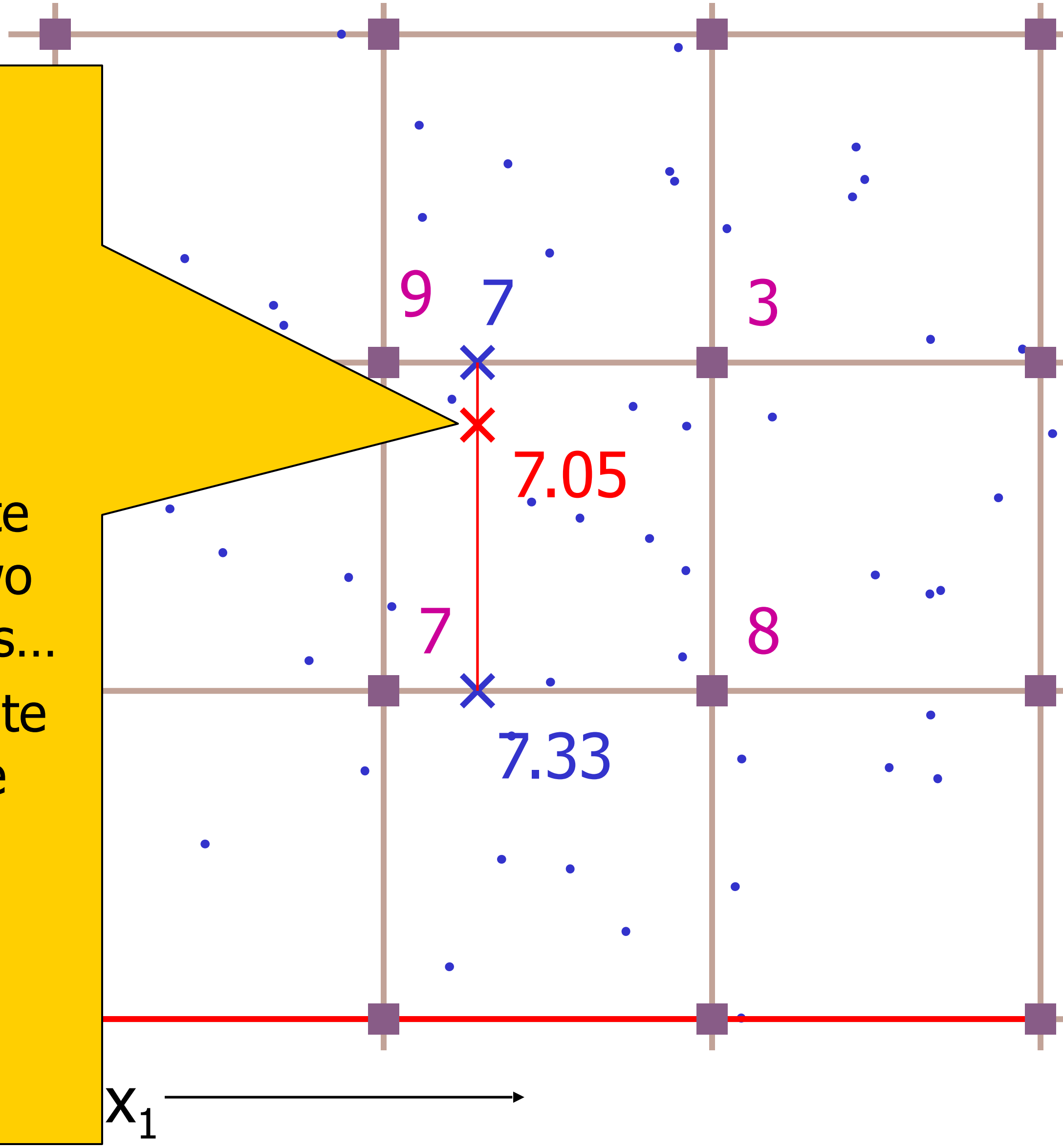
First interpolate its value on two opposite edges…

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…

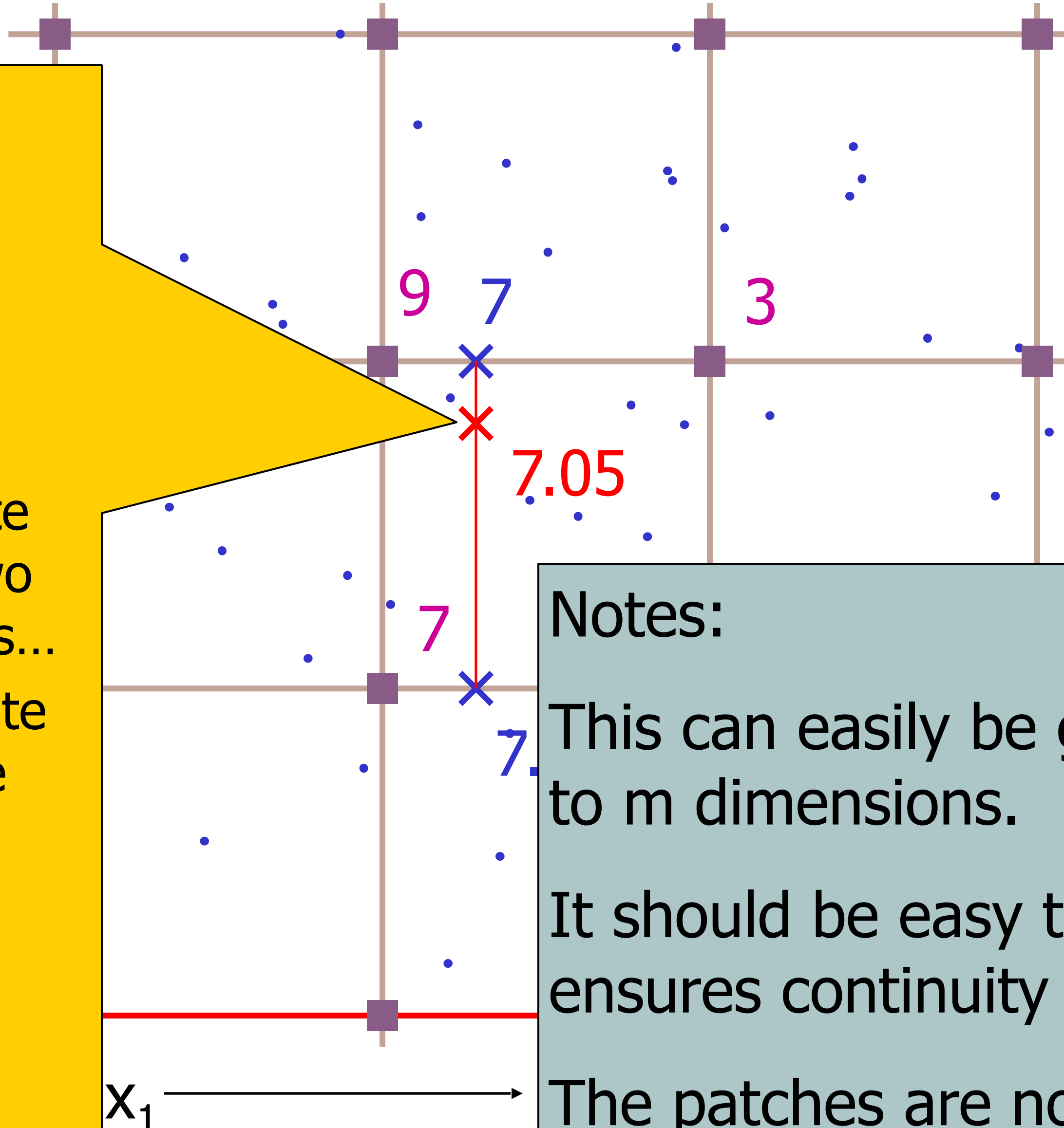
First interpolate its value on two opposite edges…

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

To predict the value here…

First interpolate its value on two opposite edges…

Then interpolate between those two values

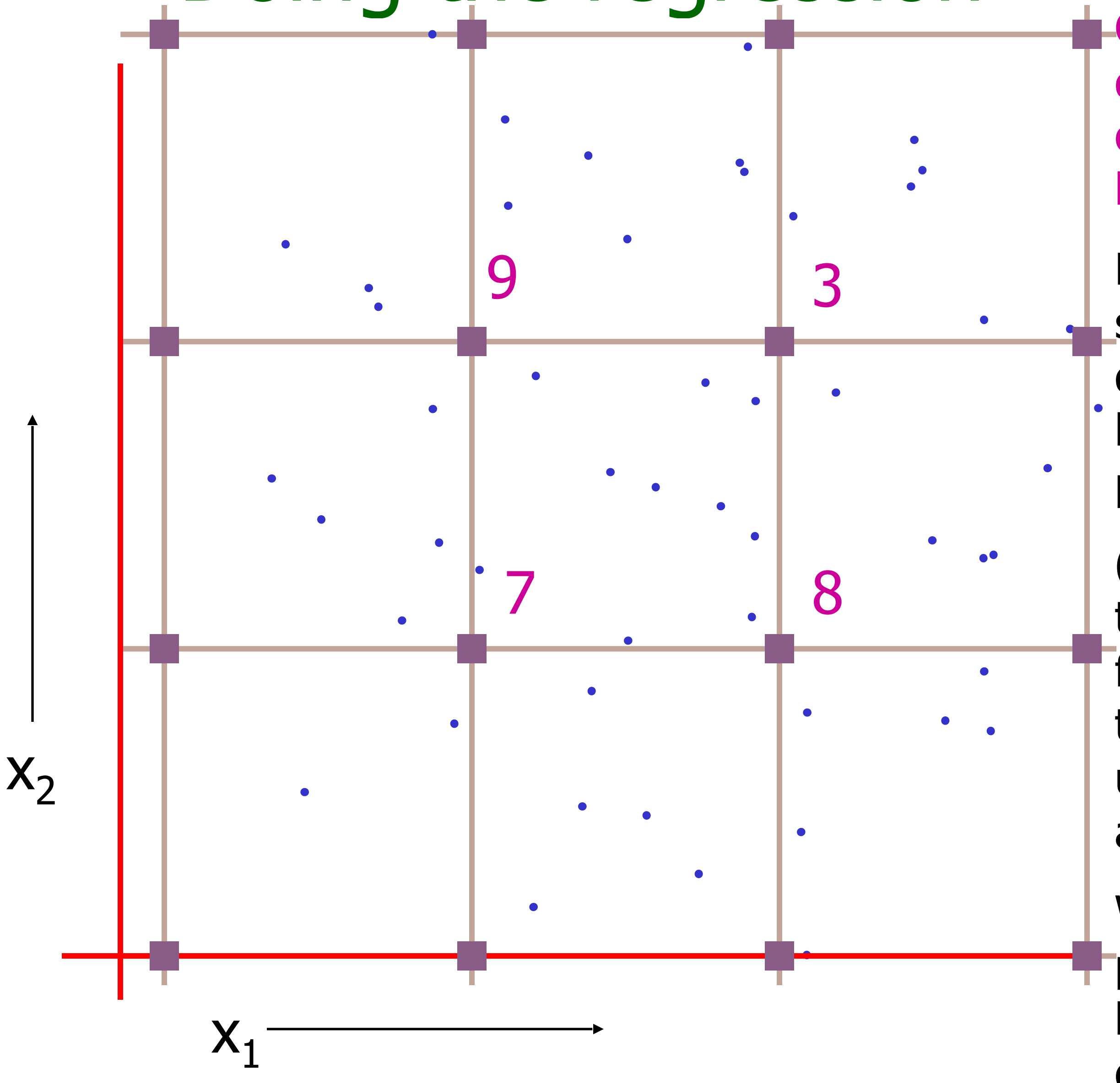
Notes:

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?

MARS: Multivariate Adaptive Regression Splines

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:

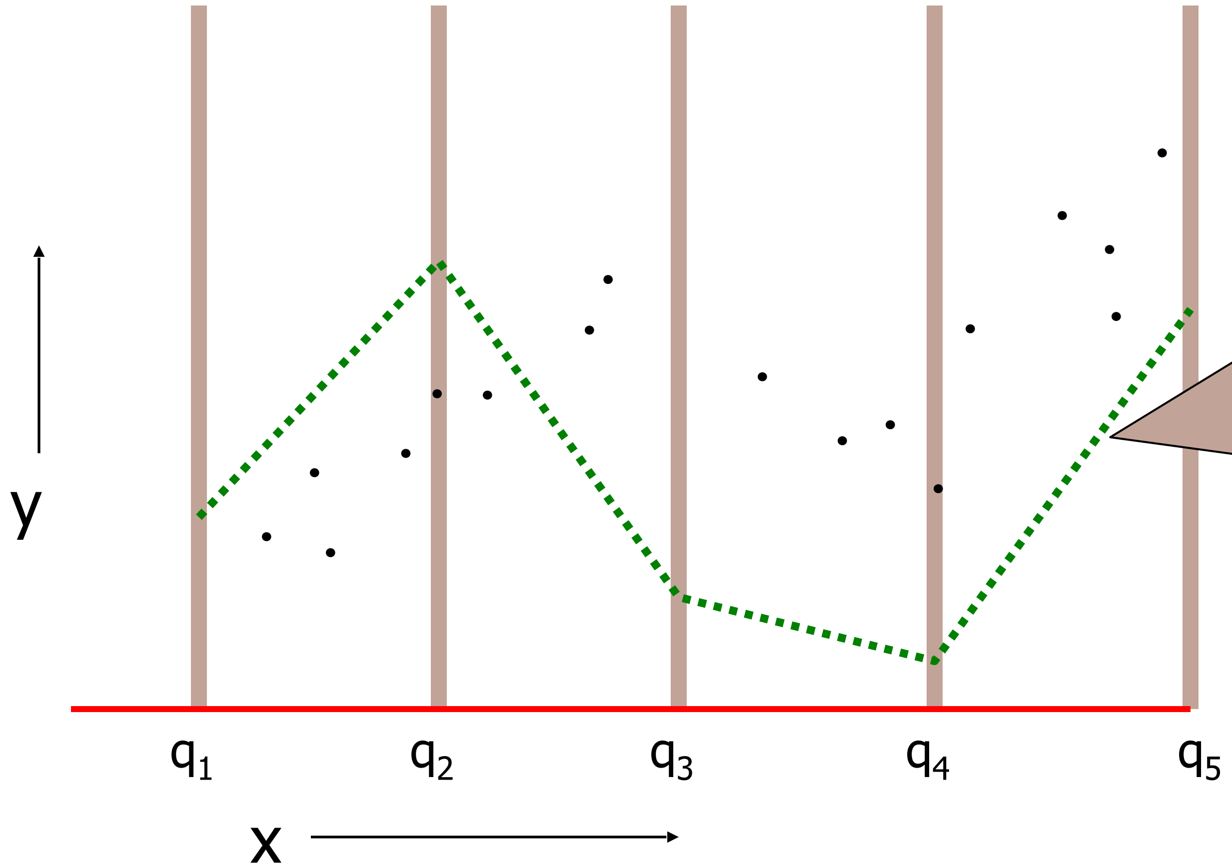
$$y^{est}(\mathbf{x}) = \sum_{k=1}^{m} g_k(x_k)$$

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



Idea: Each g_k is one of these

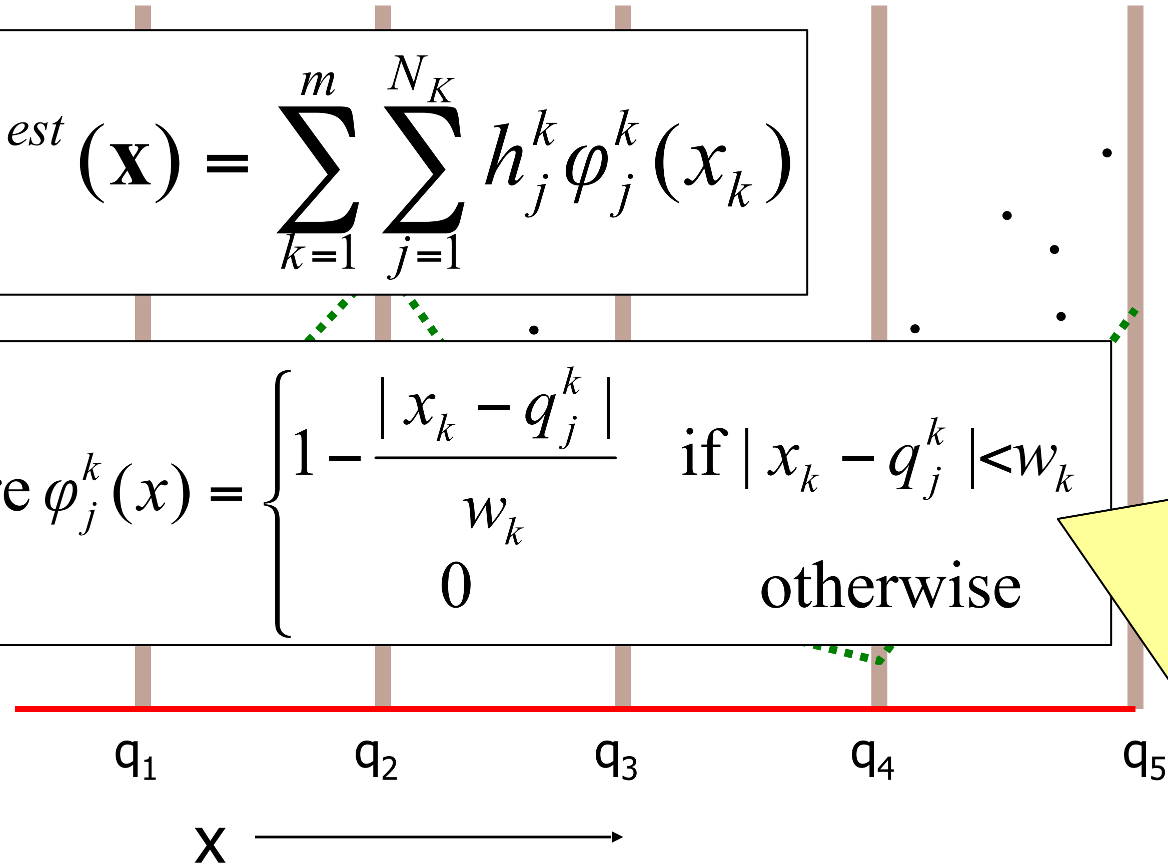
MARS

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

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
Inputs → Classifier → Predict category	Dec Tree, Gauss/Joint BC, Gauss Naïve BC,
Inputs → Density Estimator → Prob-ability	Joint DE, Naïve DE, Gauss/Joint DE, Gauss Naïve DE
Inputs → Regressor → Predict real no.	Linear Regression, Polynomial Regression, RBFs, Robust Regression Regression Trees, Multilinear Interp, MARS

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

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

MARS

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