

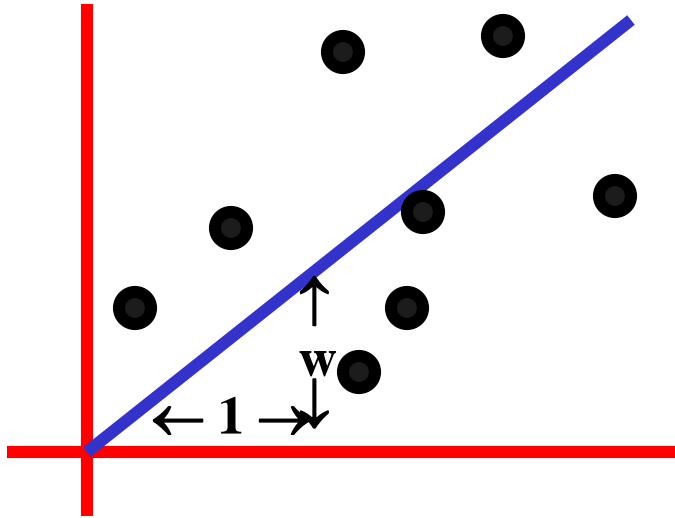
Lecture 4: An introduction to Regression

based on Andrew W. Moore Slides

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) = w x** for some unknown w .

Given the data, we can estimate w .

1-parameter linear regression

Assume that the data is formed by

$$y_i = wx_i + \text{noise}_i$$

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

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

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.

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

- 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?”

\Leftrightarrow

For what w is

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

\Leftrightarrow

For what w is

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

For what w is

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

For what w is

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

For what w is

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

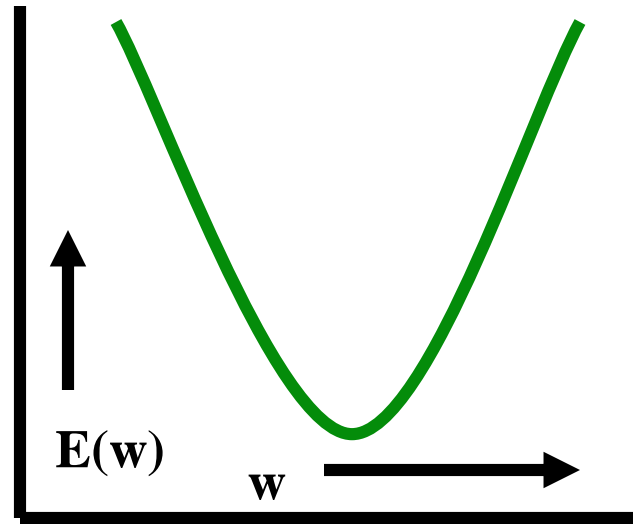
For what w is

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

Least Square

Linear Regression

The maximum likelihood w is the one that minimizes sum-of-squares of residuals



$$\begin{aligned} E &= \sum_i (y_i - wx_i)^2 \\ &= \sum_i y_i^2 - (2 \sum x_i y_i)w + \left(\sum x_i^2 \right) w^2 \end{aligned}$$

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

$$\text{Out}(x) = wx$$

We can use it for prediction

Linear Regression

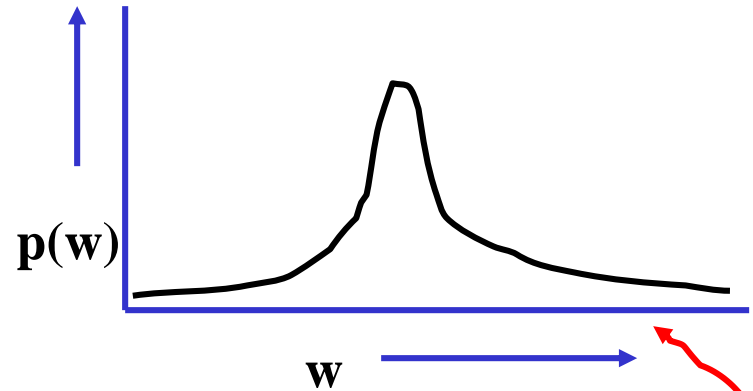
Easy to show the sum of squares is minimized when

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

The maximum likelihood mode is

$$\text{Out}(x) = wx$$

We can use it for prediction



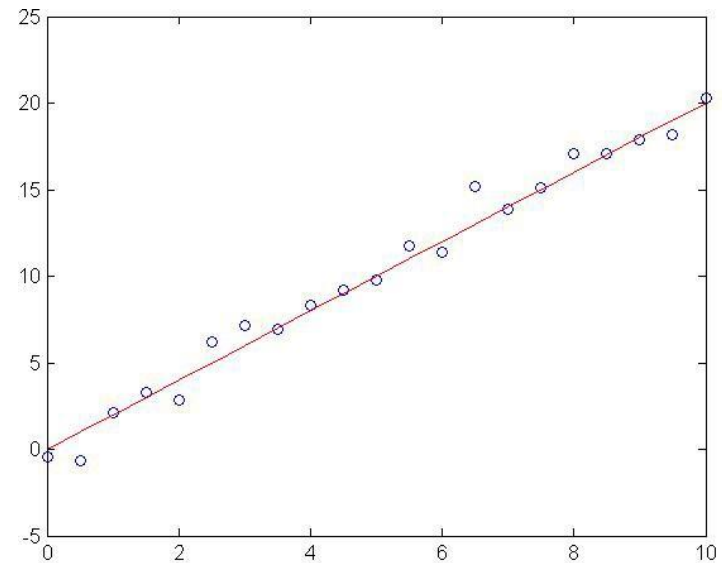
Note: In Bayesian stats you'd have ended up with a prob dist of w

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.

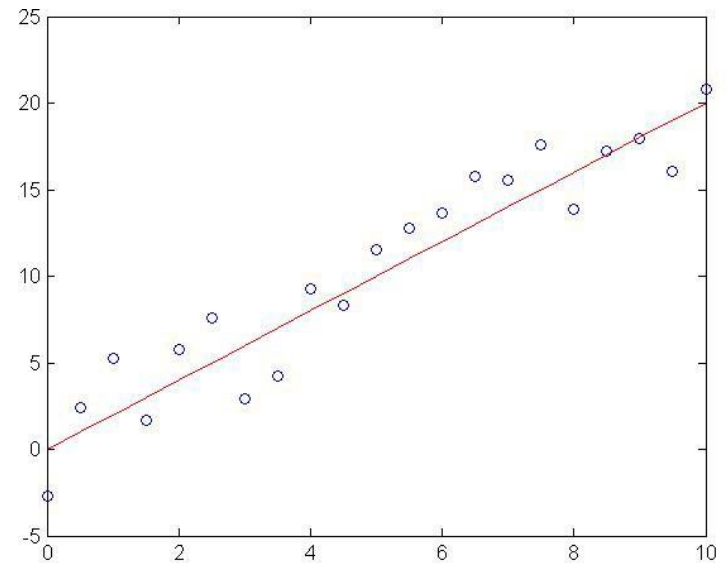
Regression example

- Generated: $w=2$
- Recovered: $w=2.03$
- Noise: $\text{std}=1$



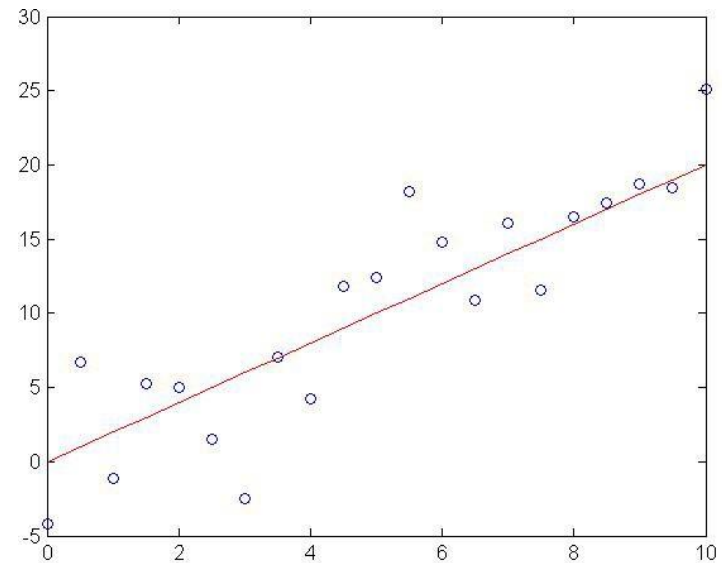
Regression example

- Generated: $w=2$
- Recovered: $w=2.05$
- Noise: $\text{std}=2$



Regression example

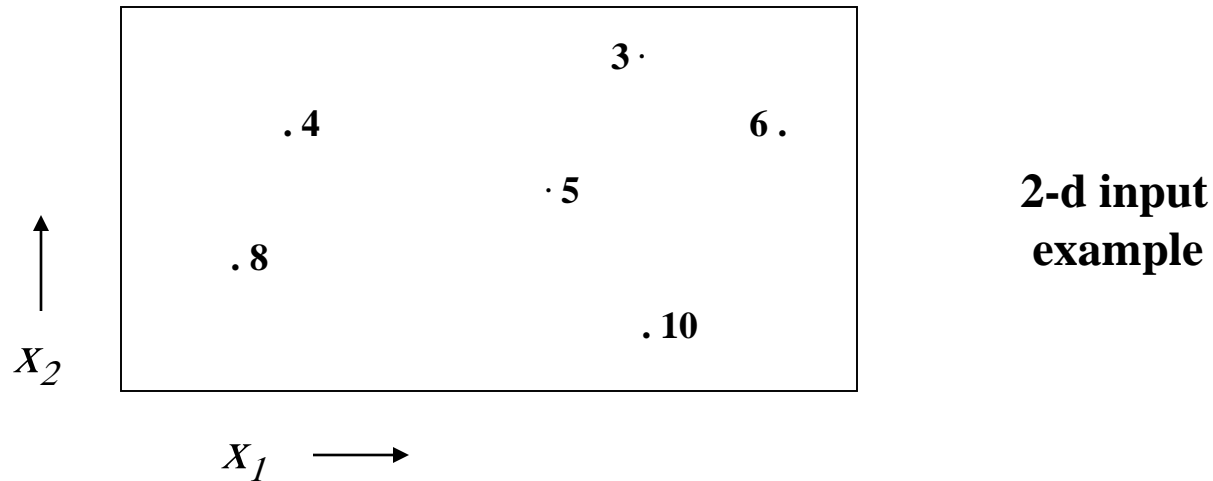
- Generated: $w=2$
- Recovered: $w=2.08$
- Noise: $\text{std}=4$



Multivariate Linear Regression

Multivariate Regression

What if the inputs are vectors?



Dataset has form

$$\begin{array}{cc} \mathbf{x}_1 & y_1 \\ \mathbf{x}_2 & y_2 \\ \mathbf{x}_3 & y_3 \\ \vdots & \vdots \\ \mathbf{x}_R & y_R \end{array}$$

Multivariate Regression

Write matrix \mathbf{X} and \mathbf{Y} thus:

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

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

The linear regression model assumes a vector \mathbf{w} such that

$$\text{Out}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = w_1 x[1] + w_2 x[2] + \dots w_m x[D]$$

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

Multivariate Regression

Write matrix \mathbf{X} and \mathbf{Y} thus:

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

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

The linear regression model assumes a vector \mathbf{w} such that

$$\text{Out}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = w_1 x[1] + w_2 x[2] + \dots w_m x[D]$$

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

Multivariate Regression

- $|Y - XW|^2 = (Y - XW)^T(Y - XW)$

$$A = (Y^T - W^T X^T)(Y - XW) = Y^T Y - Y^T XW - W^T X^T Y + W^T X^T XW =$$

$$A = Y^T Y - 2W^T X^T Y + W^T X^T XW$$

$$\frac{\partial A}{\partial W} = -2X^T Y + 2X^T XW = 0 \rightarrow$$

$$W = (X^T X)^{-1} X^T Y$$

Multivariate Regression (con't)

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

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

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

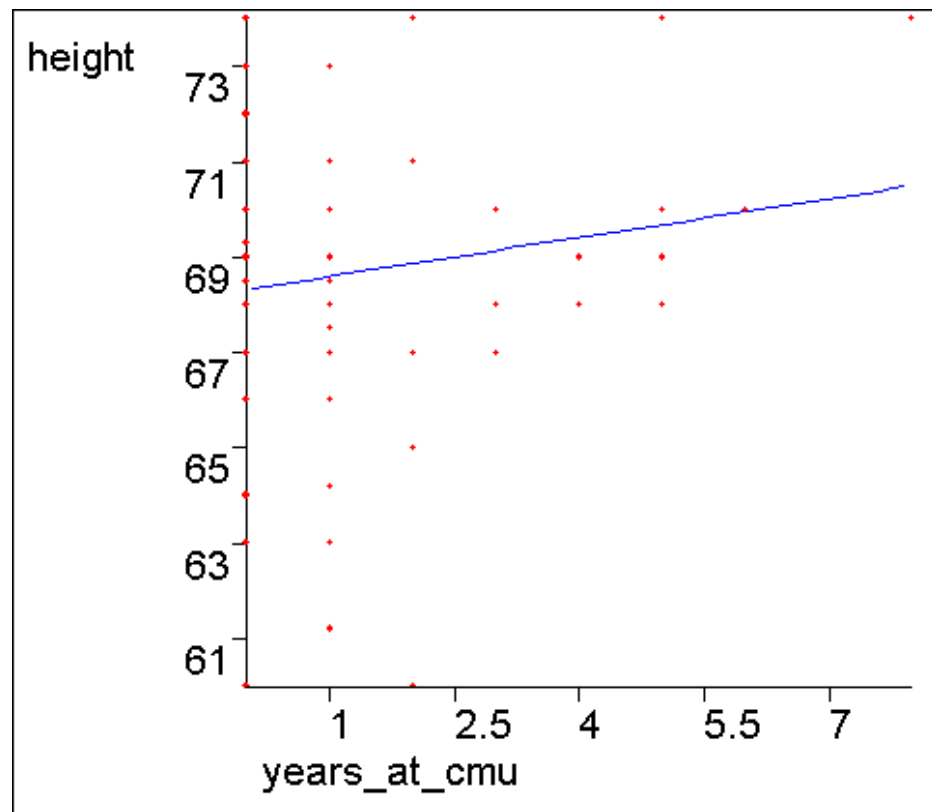
Constant Term in Linear Regression

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:

$$Y = w_1 X_1 + w_2 X_2$$

...has to be a poor model

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

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

After:

$$Y = w_0 X_0 + w_1 X_1 + w_2 X_2$$

$$= w_0 + w_1 X_1 + w_2 X_2$$

...has a fine constant term

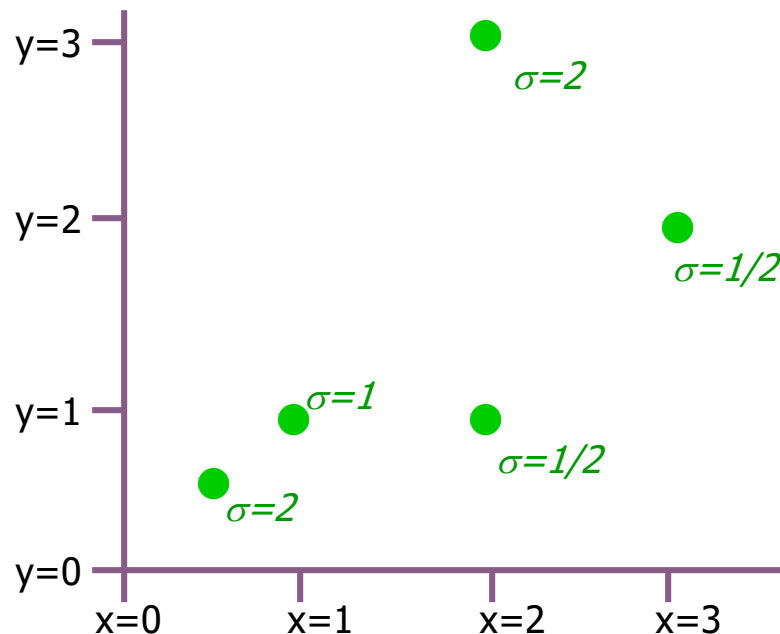
Heteroscedasticity...

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
$1/2$	$1/2$	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

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

w

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

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

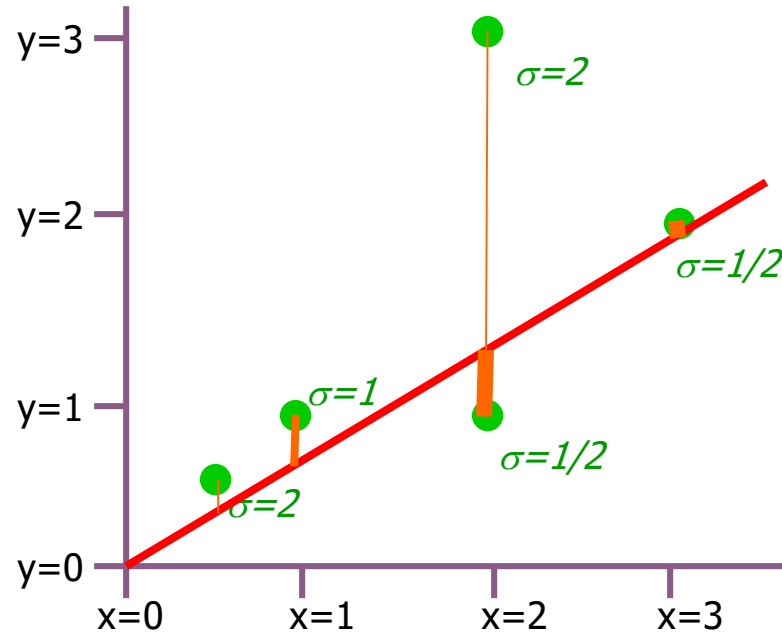
$$\frac{\left(\sum_{i=1}^R \frac{x_i y_i}{\sigma_i^2} \right)}{\left(\sum_{i=1}^R \frac{x_i^2}{\sigma_i^2} \right)}$$

Trivial algebra

This is Weighted Regression

- We are asking to minimize the weighted sum of squares

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



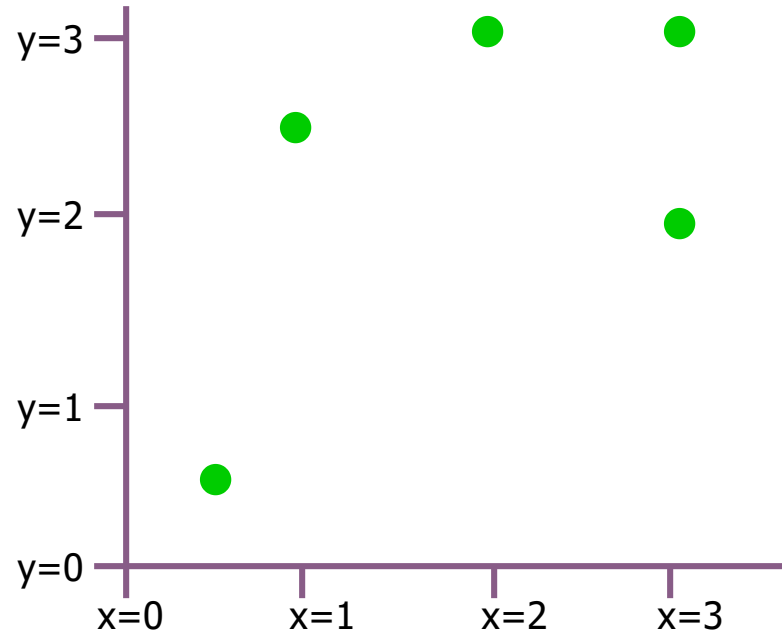
where weight for i'th datapoint is $\frac{1}{\sigma_i^2}$

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
$1/2$	$1/2$
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

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

w

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

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

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

w

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

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

algebraic
solution???

So guess what
we do?

Non-linear MLE estimation

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

Common (but not only) approach:
Numerical Solutions:

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

$$w + x_i)^2 =$$

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

$$\frac{d}{dw} (w + x_i)^2 = 0 \Bigg) =$$

Setting dLL/dw equal to zero

algebraic
solution???

*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
\vdots	\vdots	\vdots

$\mathbf{X} =$	3	2
	1	1
	\vdots	\vdots

$\mathbf{y} =$	7
	3
	\vdots

$y_1 = 7..$

$\mathbf{z} =$	1	3	2
	1	1	1
	\vdots		\vdots

$\mathbf{y} =$	7
	3
	\vdots

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

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

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

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

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
\vdots	\vdots	\vdots

$\mathbf{X} =$	3	2
	1	1
	\vdots	\vdots

$\mathbf{y} =$	7
	3
	\vdots

$y_1 = 7..$

$\mathbf{Z} =$	1	3	2	9	6	4
	1	1	1	1	1	1
	\vdots					\vdots

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

$\mathbf{y} =$	7
	3
	\vdots

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

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

Quadratic Regression

It's trivial

X_1	X_2
3	2
1	1
\vdots	\vdots

Z=

1
1
\vdots

z=(1

Each component of a \mathbf{z} vector is called a term.

Each column of the \mathbf{Z} matrix is called a term column

How many terms in a quadratic regression with m inputs?


- 1 constant term
 - m linear terms
 - $(m+1)\text{-choose-}2 = m(m+1)/2$ quadratic terms
- $(m+2)\text{-choose-}2$ 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
\vdots	\vdots	\vdots

X =	3	2	y =	7
	1	1		3
	:	:		:

 $\mathbf{Z} =$

1	3	2	9	6	...
1	1	1	1	1	...
:					...

 $\mathbf{y} =$

7
3
:

$\mathbf{z} = (\text{all products of powers of inputs in which sum of powers is } q \text{ or less})$

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

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

m inputs, degree Q: how many terms?

= the number of unique terms of the form

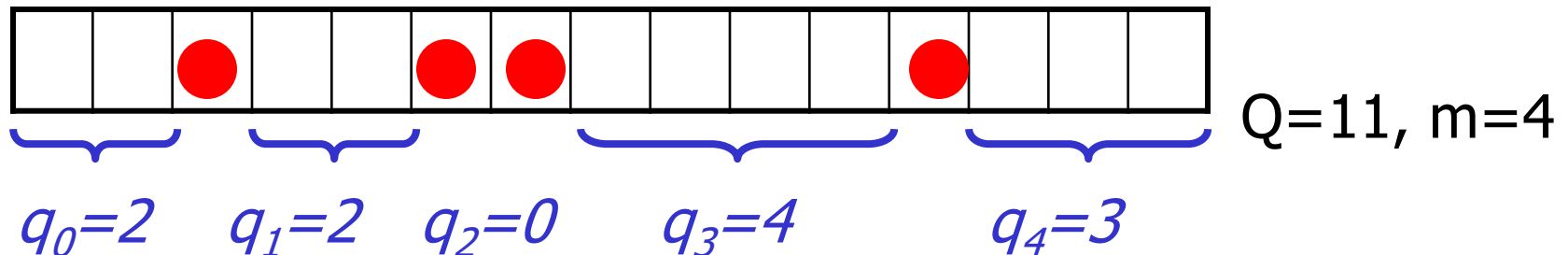
$$x_1^{q_1} x_2^{q_2} \dots x_m^{q_m} \text{ where } \sum_{i=1}^m q_i \leq Q$$

= 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 $[q_0, q_1, q_2, \dots, q_m]$ in which $\sum q_i = Q$

= 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
\vdots	\vdots	\vdots

$\mathbf{X} =$	3	2	$\mathbf{y} =$	7
	1	1		3
	\vdots	\vdots		\vdots

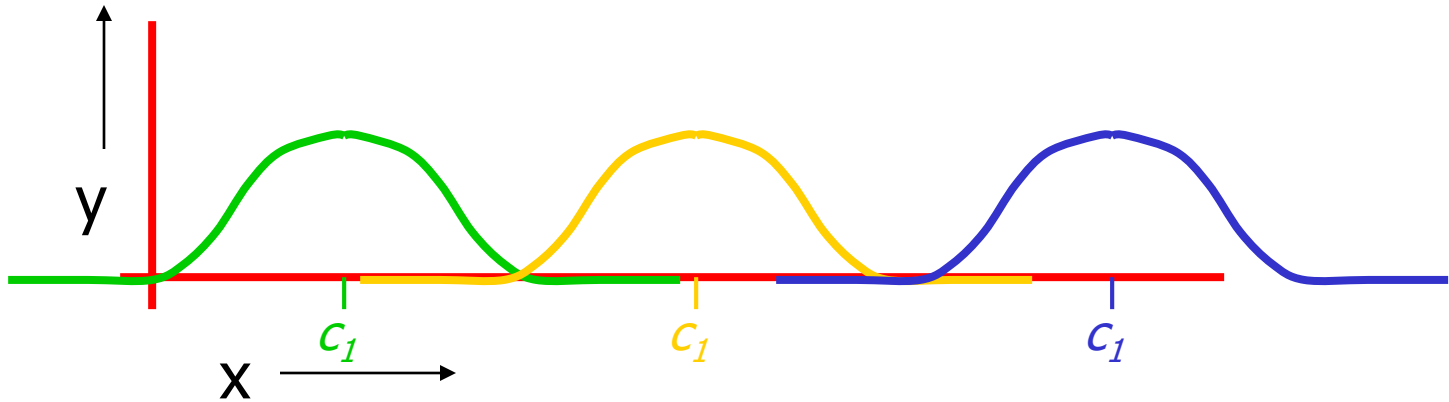
$\mathbf{Z} =$	$\mathbf{y} =$	7
		3
		\vdots

$\mathbf{z} = (\text{list of radial basis function evaluations})$

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

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

1-d RBFs

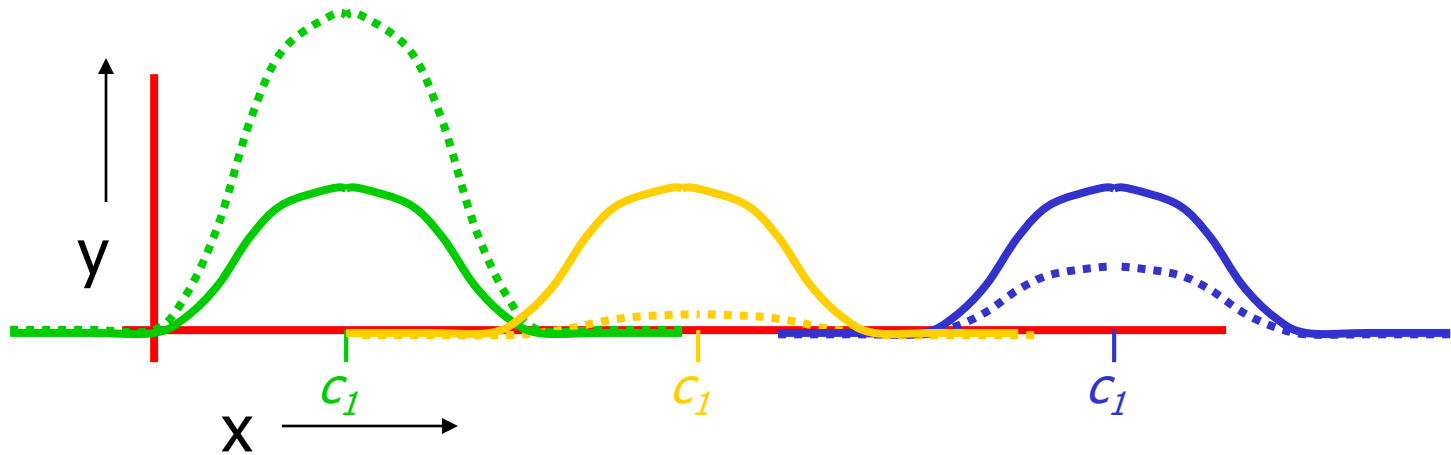


$$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$$

where

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

Example

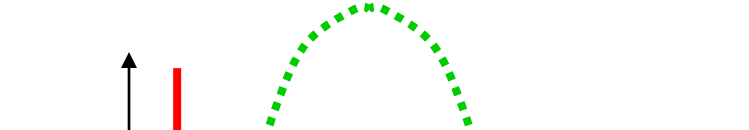


$$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$$

where

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

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*)

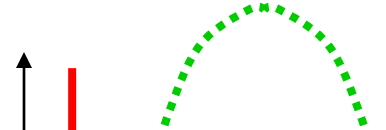
*Usually much better than the crappy
overlap on my diagram

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

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

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

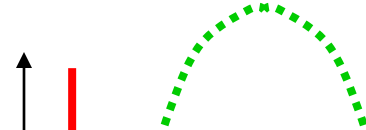
where

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

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

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

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)

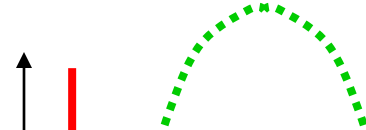
$$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 5\phi_3(x)$$

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 ?

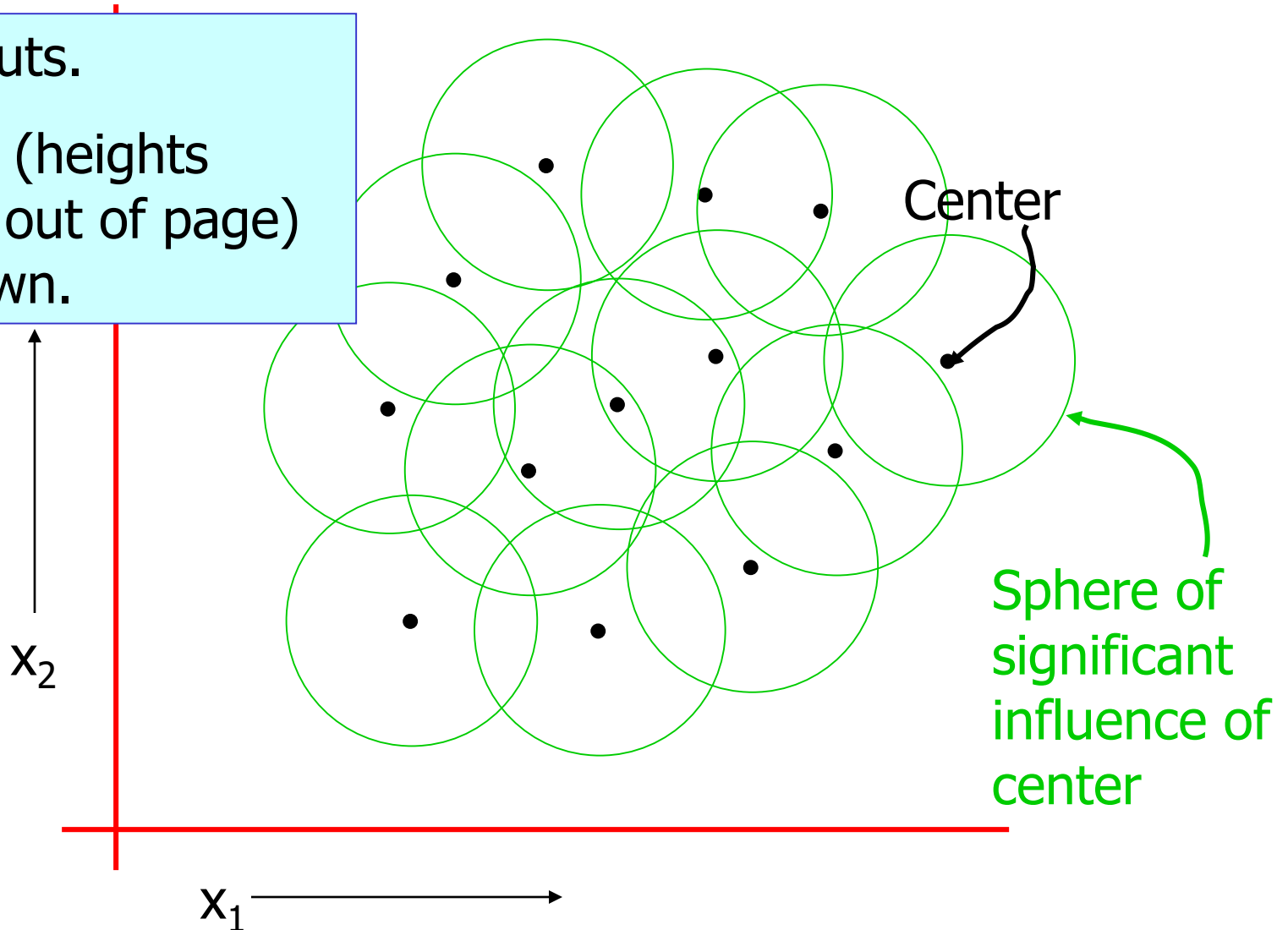
(But I'd like to see, or hope someone's already done, a hybrid, where the c_i 's and KW are updated with gradient descent while the β_j 's use matrix inversion)

Answer: Gradient Descent

Radial Basis Functions in 2-d

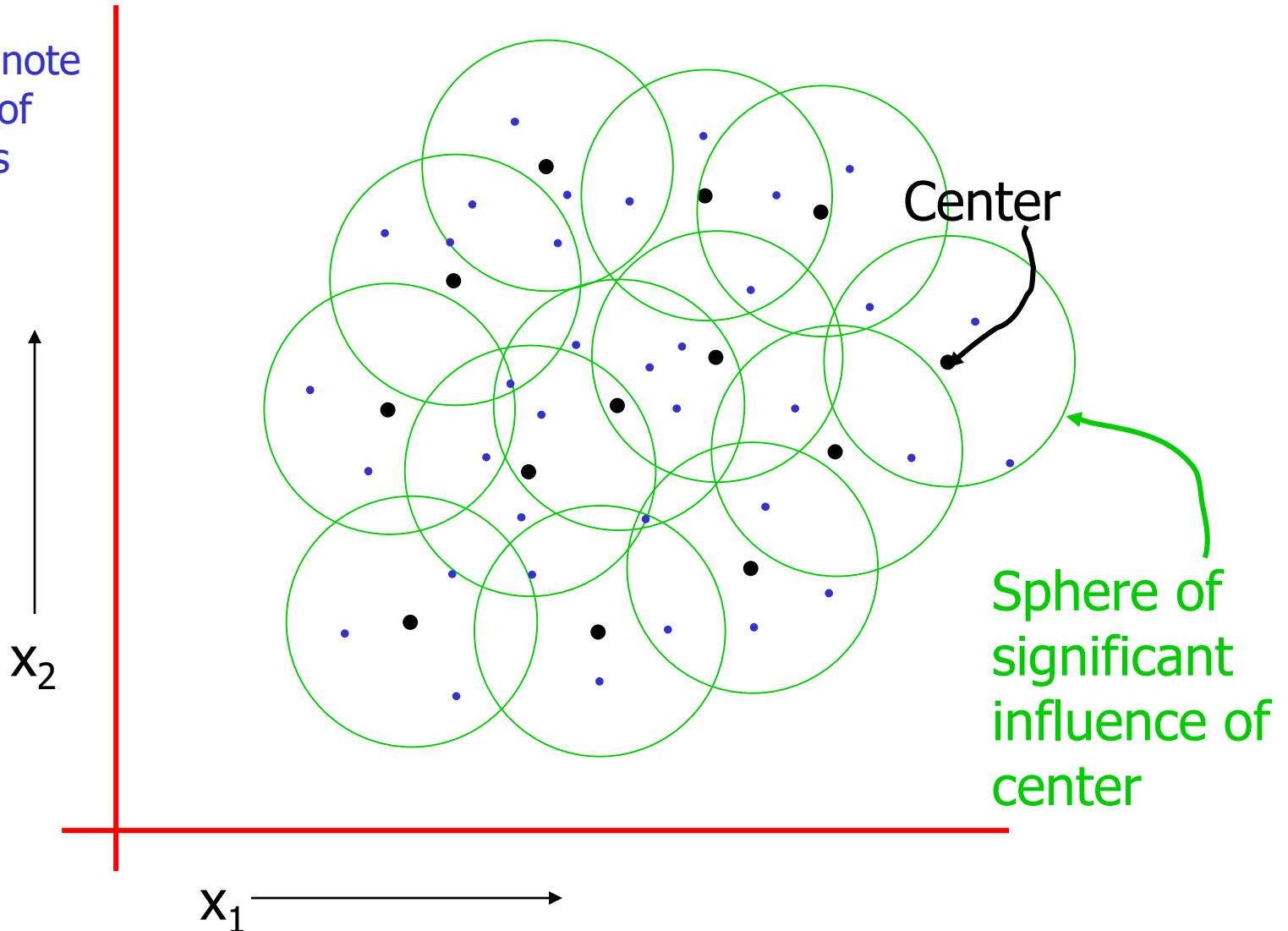
Two inputs.

Outputs (heights sticking out of page) not shown.



Happy RBFs in 2-d

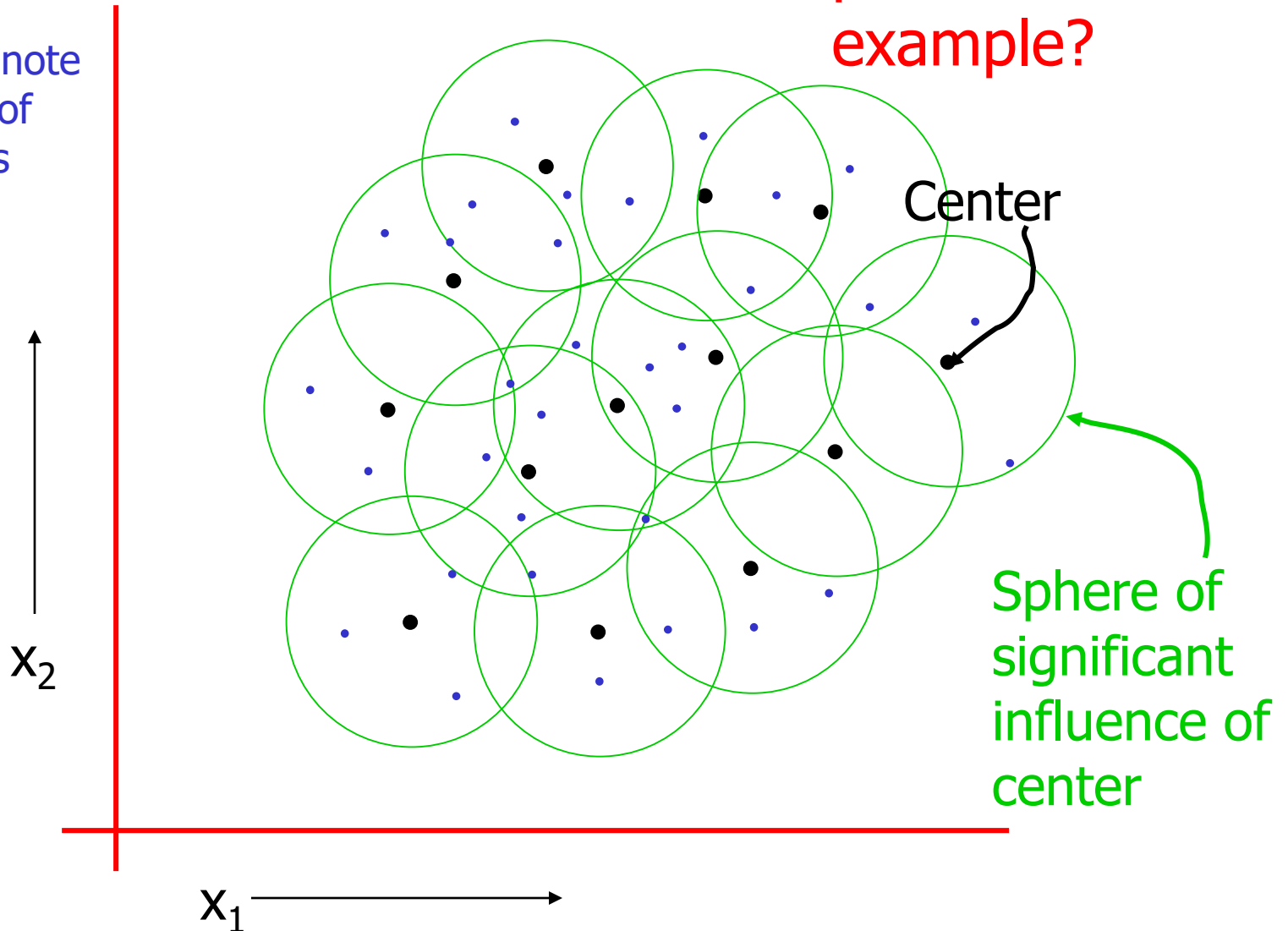
Blue dots denote
coordinates of
input vectors



Crabby RBFs in 2-d

What's the problem in this example?

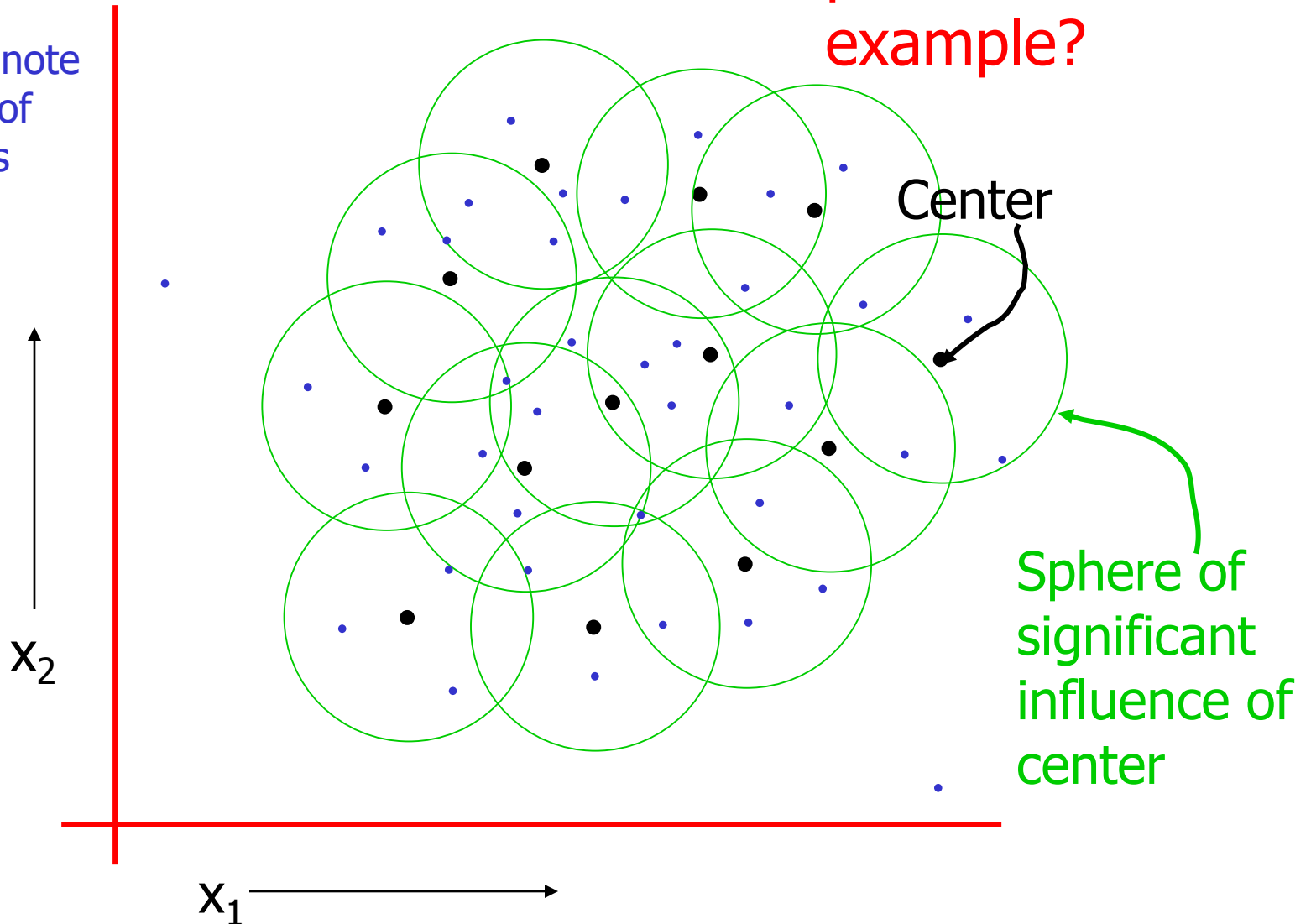
Blue dots denote coordinates of input vectors



More crabby RBFs

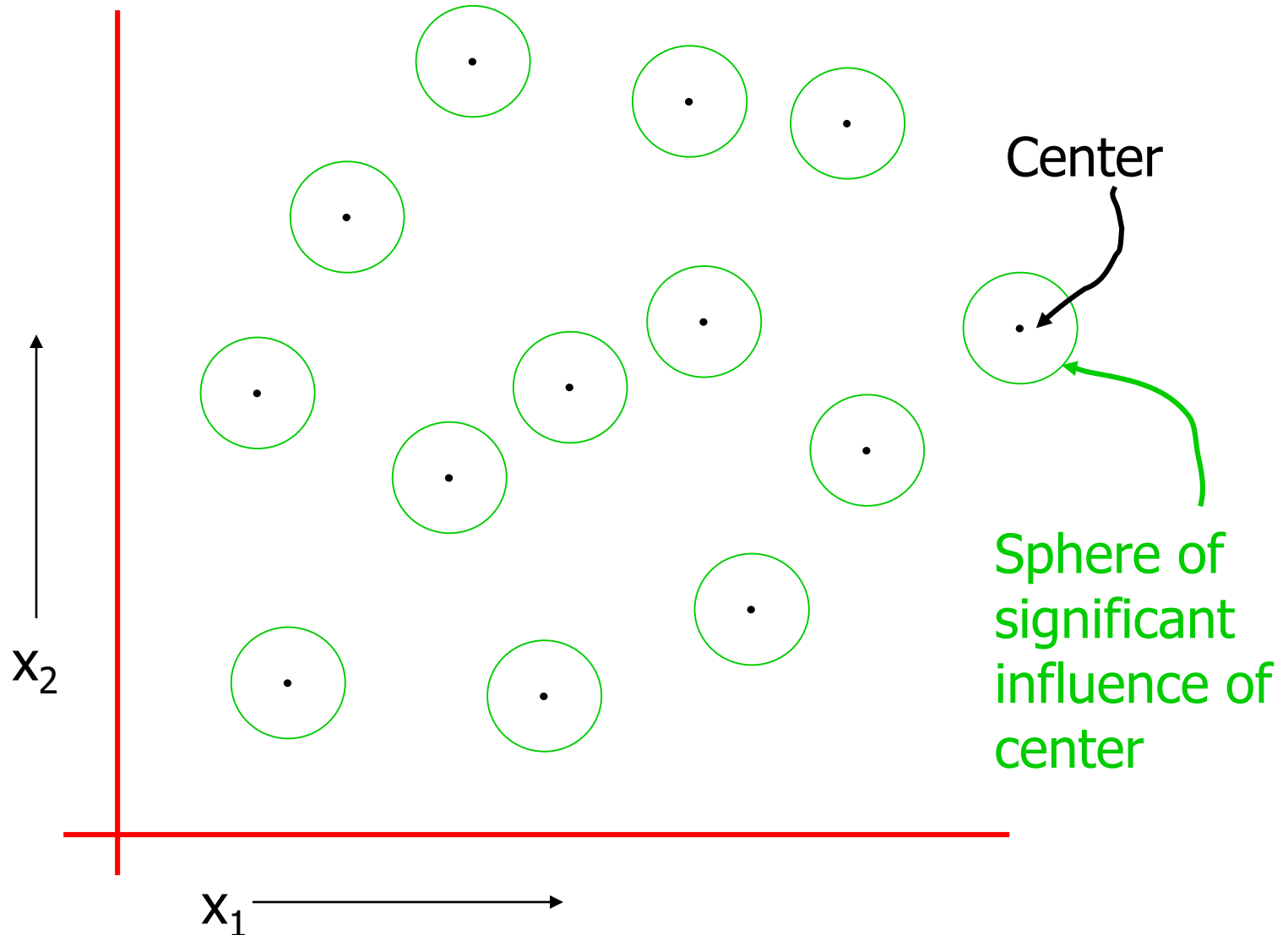
And what's the problem in this example?

Blue dots denote coordinates of input vectors



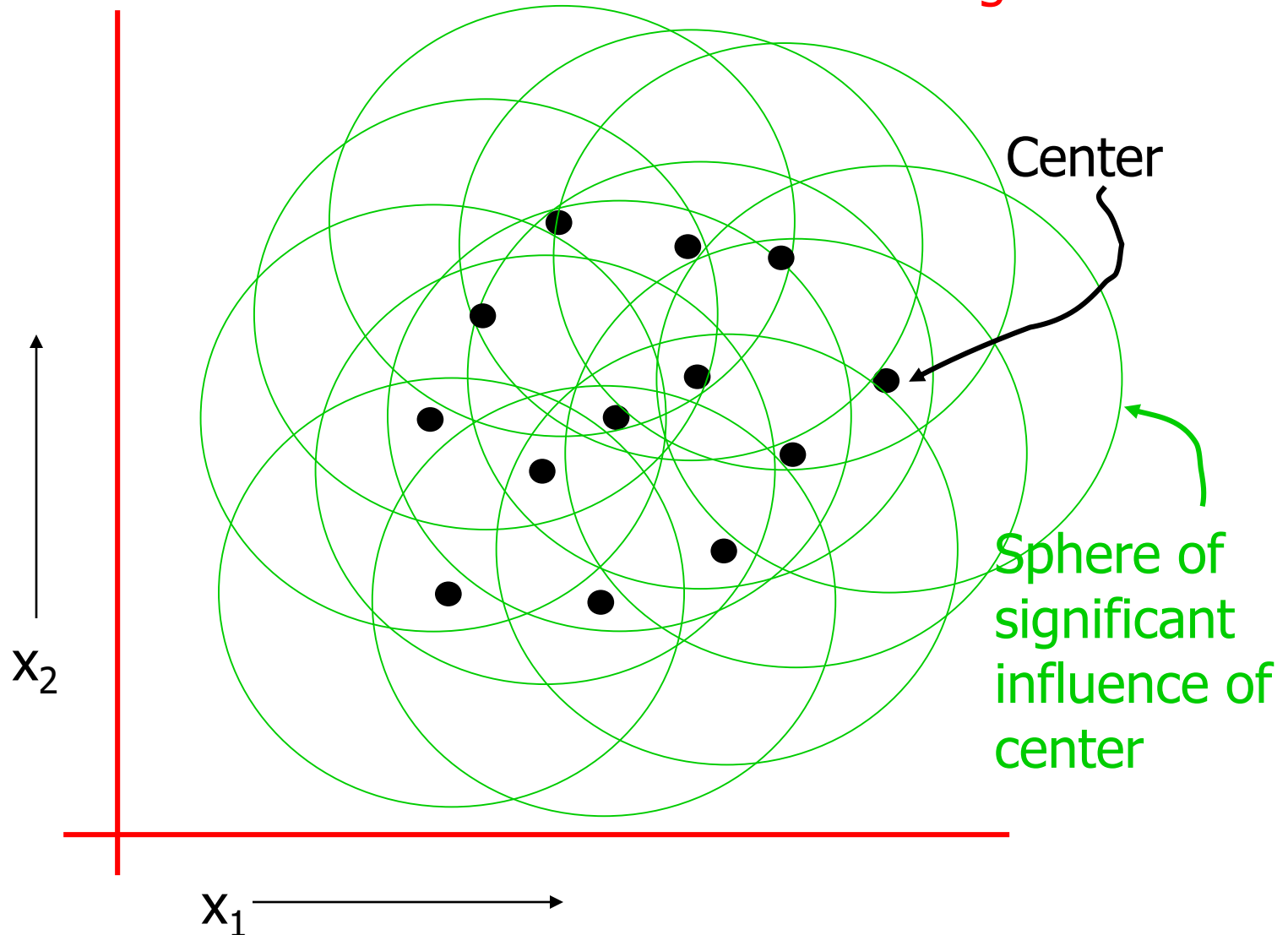
Hopeless!

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



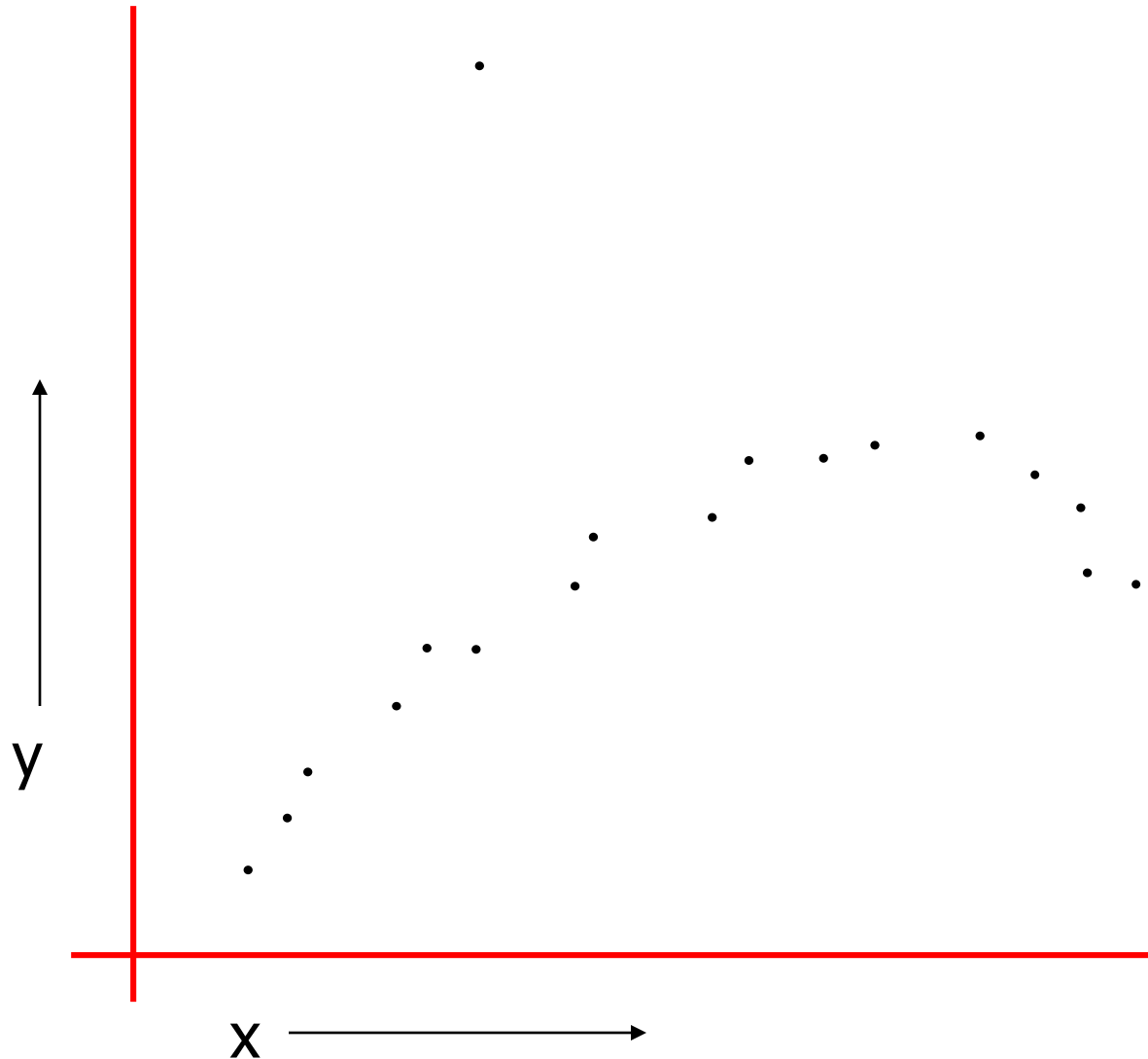
Unhappy

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

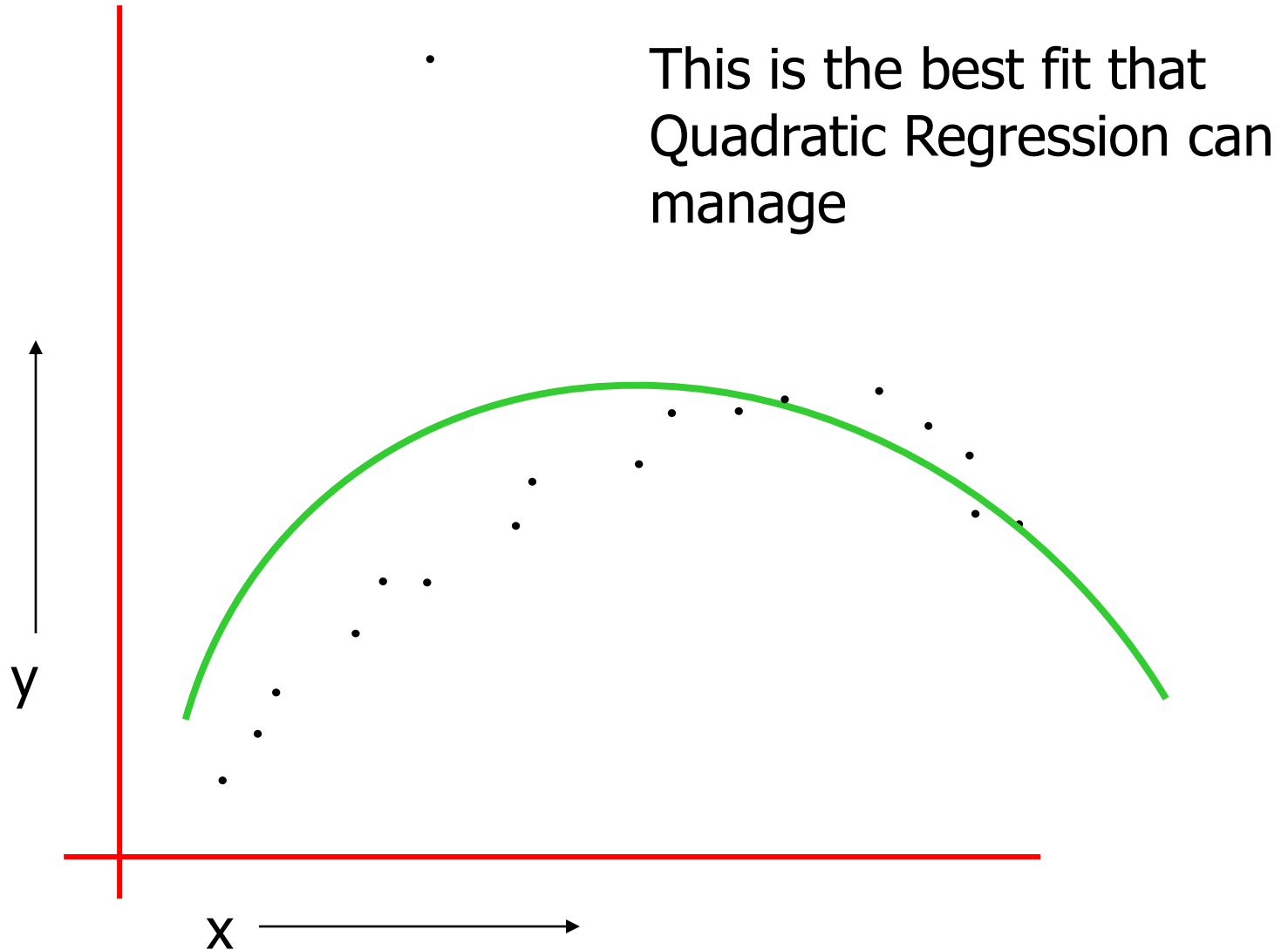


Robust Regression

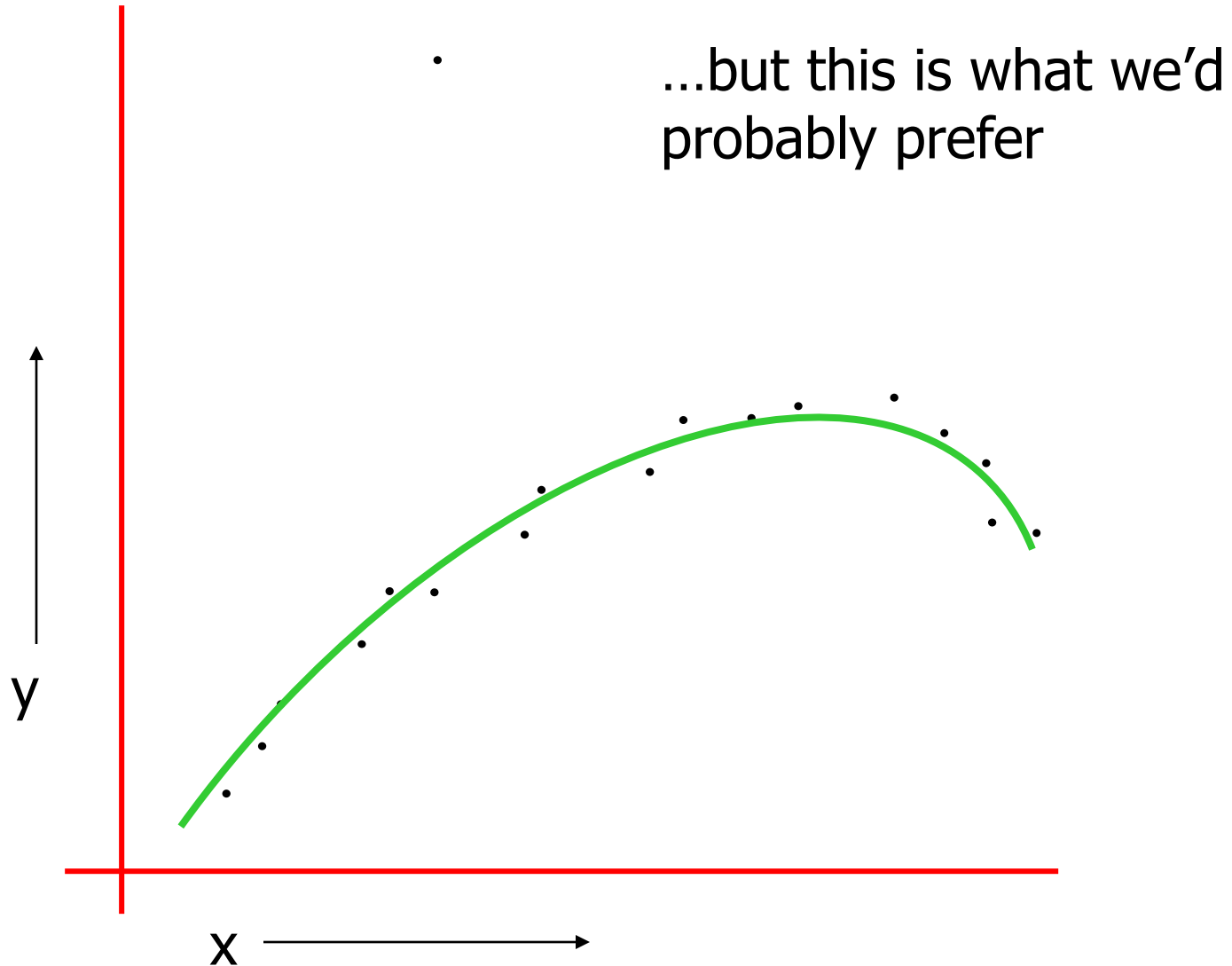
Robust Regression



Robust Regression

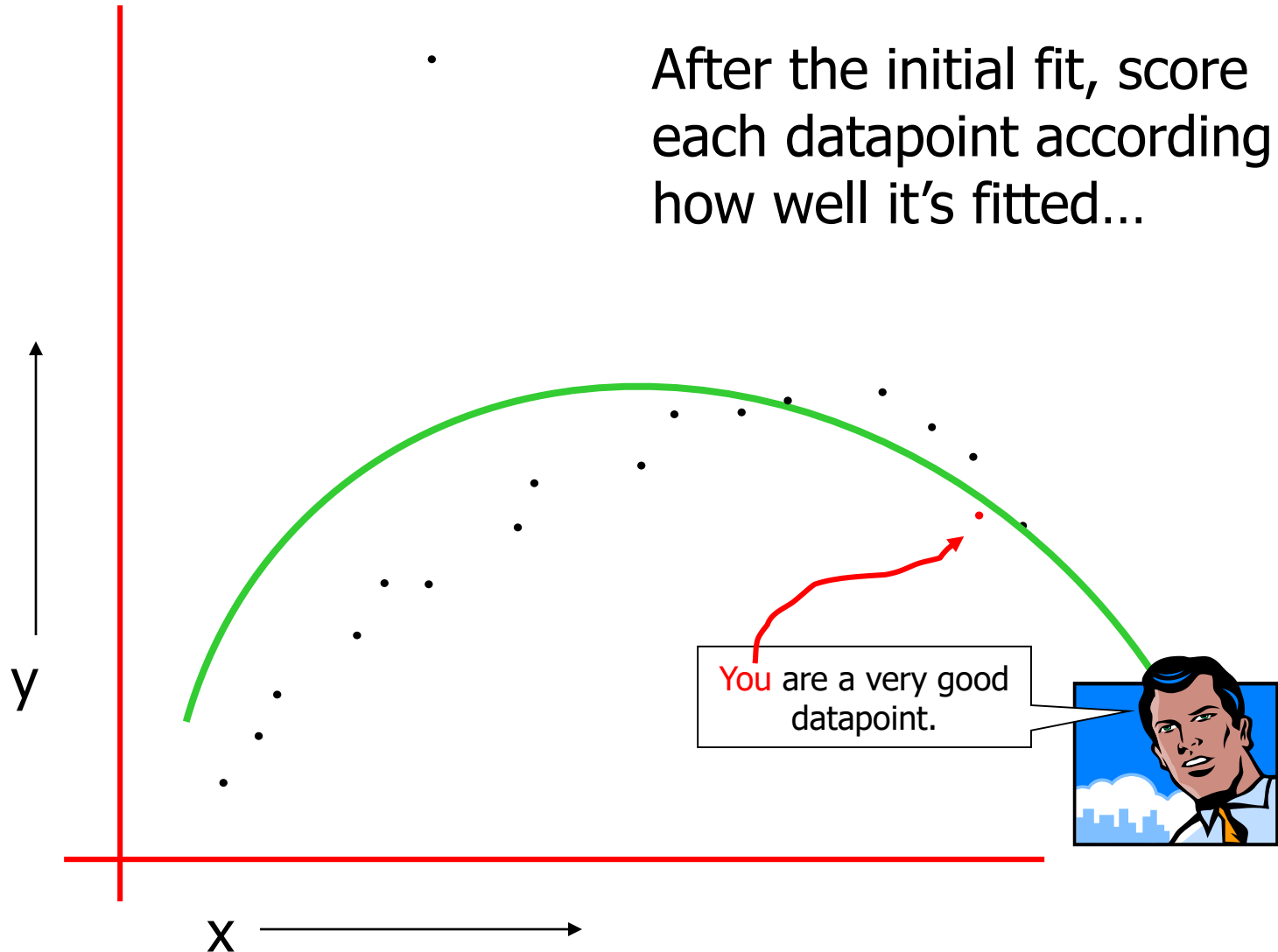


Robust Regression



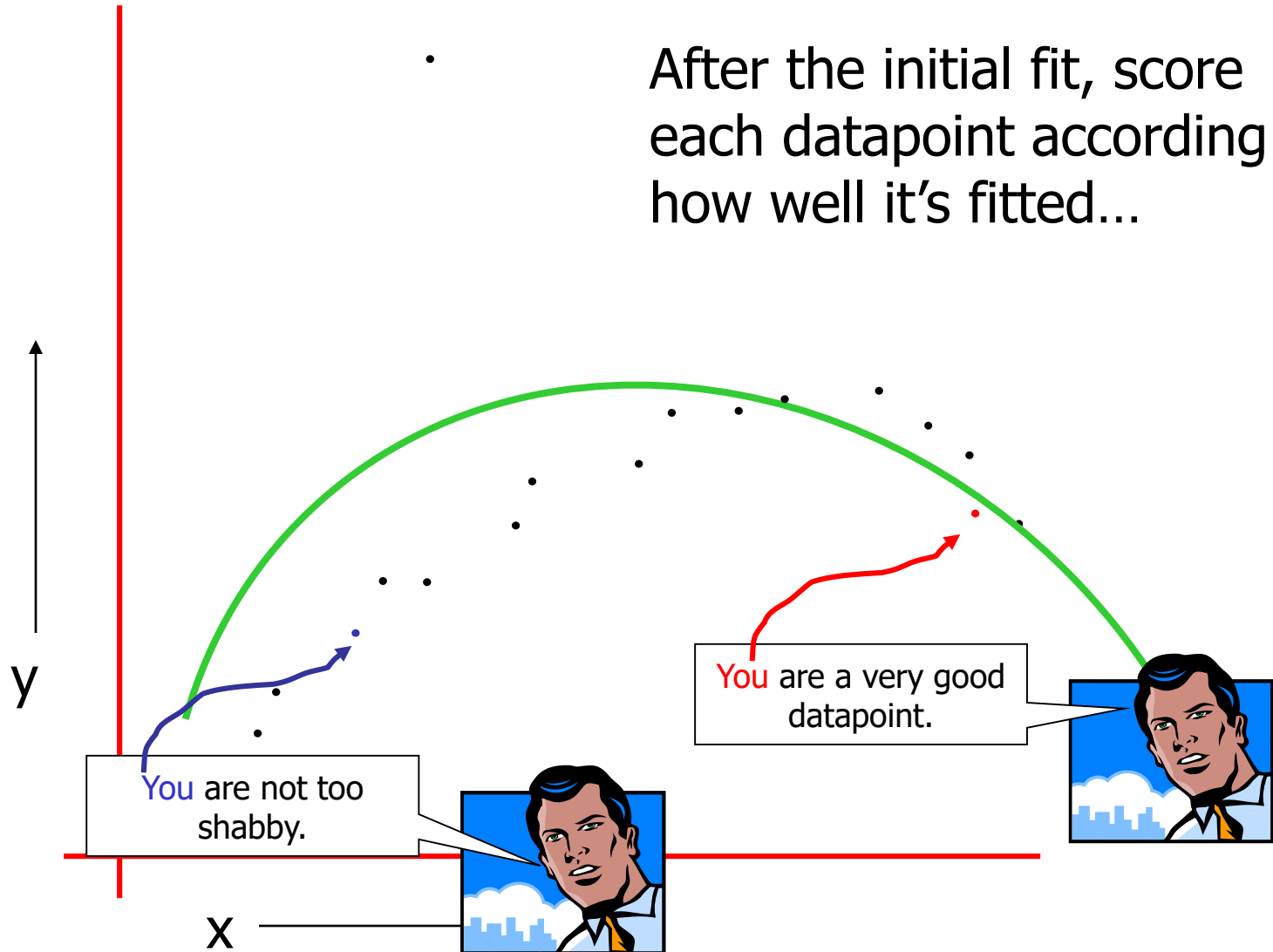
LOESS-based Robust Regression

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

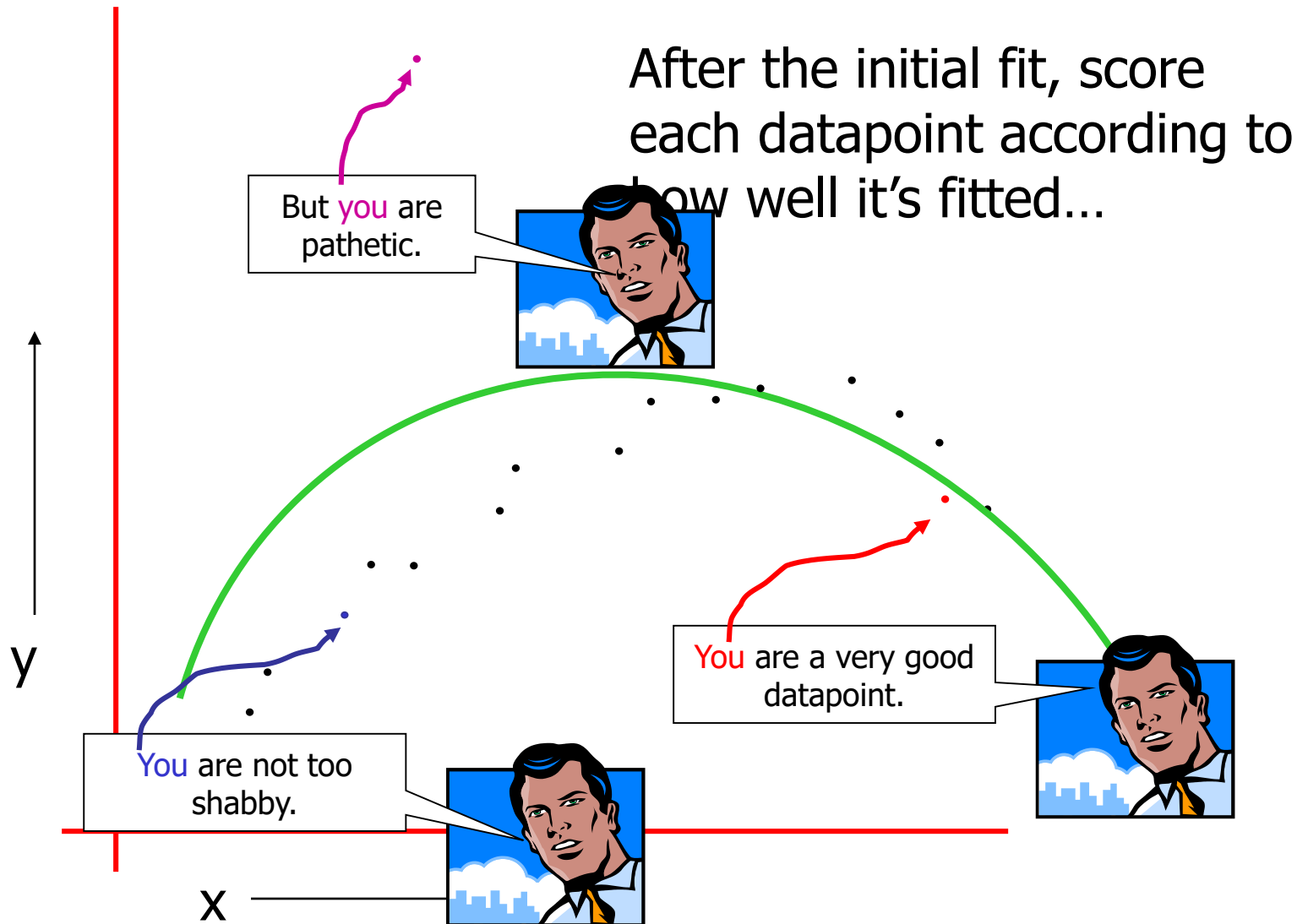


LOESS-based Robust Regression

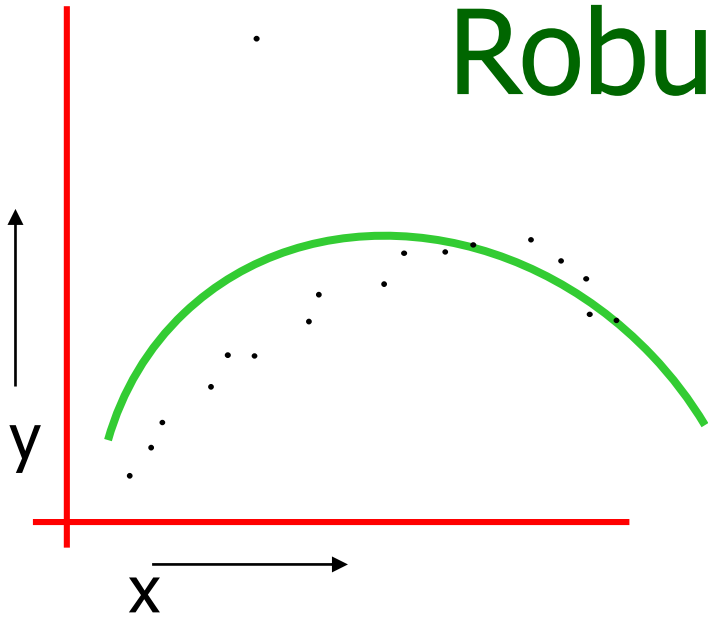
After the initial fit, score each datapoint according to how well it's fitted...



LOESS-based Robust Regression



Robust Regression

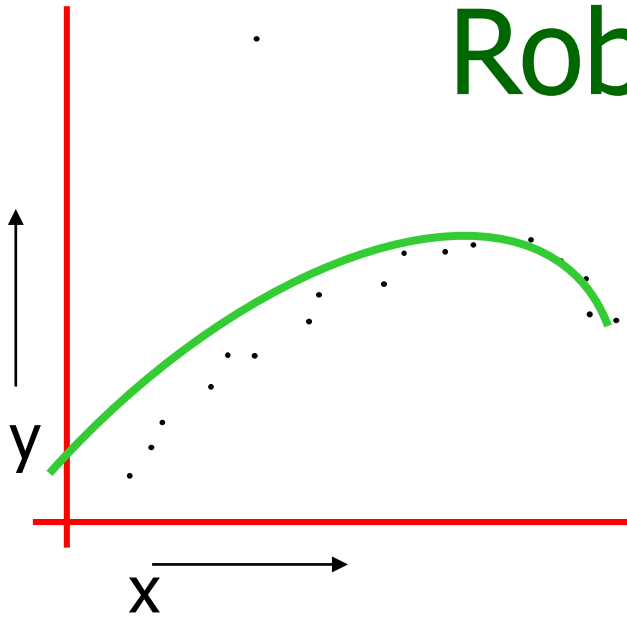


For $k = 1$ to $R...$

- Let (x_k, y_k) be the k th datapoint
- Let y_k^{est} 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:

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

Robust Regression



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.

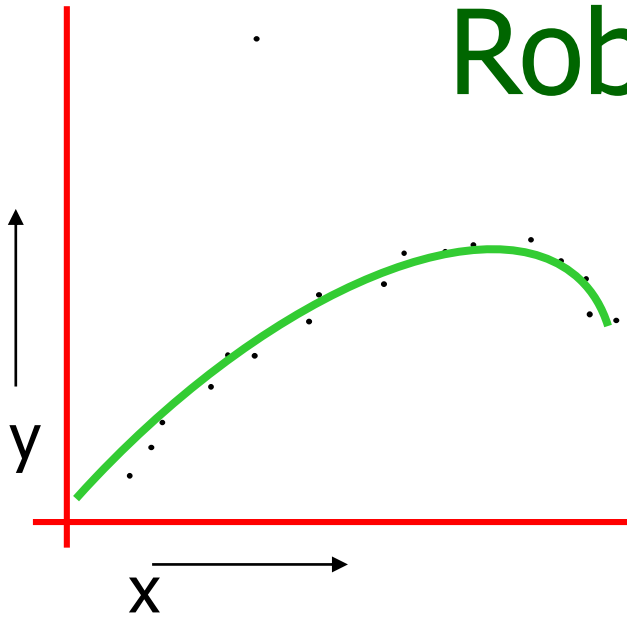
Guess what happens next?

For $k = 1$ to $R...$

- Let (x_k, y_k) be the k th datapoint
- Let y_k^{est} 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:

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

Robust Regression



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!

For $k = 1$ to $R...$

- Let (x_k, y_k) be the k th datapoint
- Let y_k^{est} 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:

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

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 $\beta_0, \beta_1, \beta_2, p, \mu$ 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

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

Computational task is to find the Maximum Likelihood $\beta_0, \beta_1, \beta_2, p, \mu$ and σ_{huge}

Mysteriously, the reweighting procedure does this computation for us.

Your first glimpse of two spectacular letters:

E.M.