Regression

Paul R. Schrater

CSCI 5521, Fall 2007

Note: Several slides taken from

http://www.cs.cmu.edu/~awm/tutorials

Classification

X

Anything:

- continuous ($\Re$, $\Re^d$, …)
- discrete ({0,1}, {1,…k}, …)
- structured (tree, string, …)
- …

Y

- discrete:
  - {0,1} *binary*
  - {1,…k} *multi-class*
  - tree, etc. *structured*

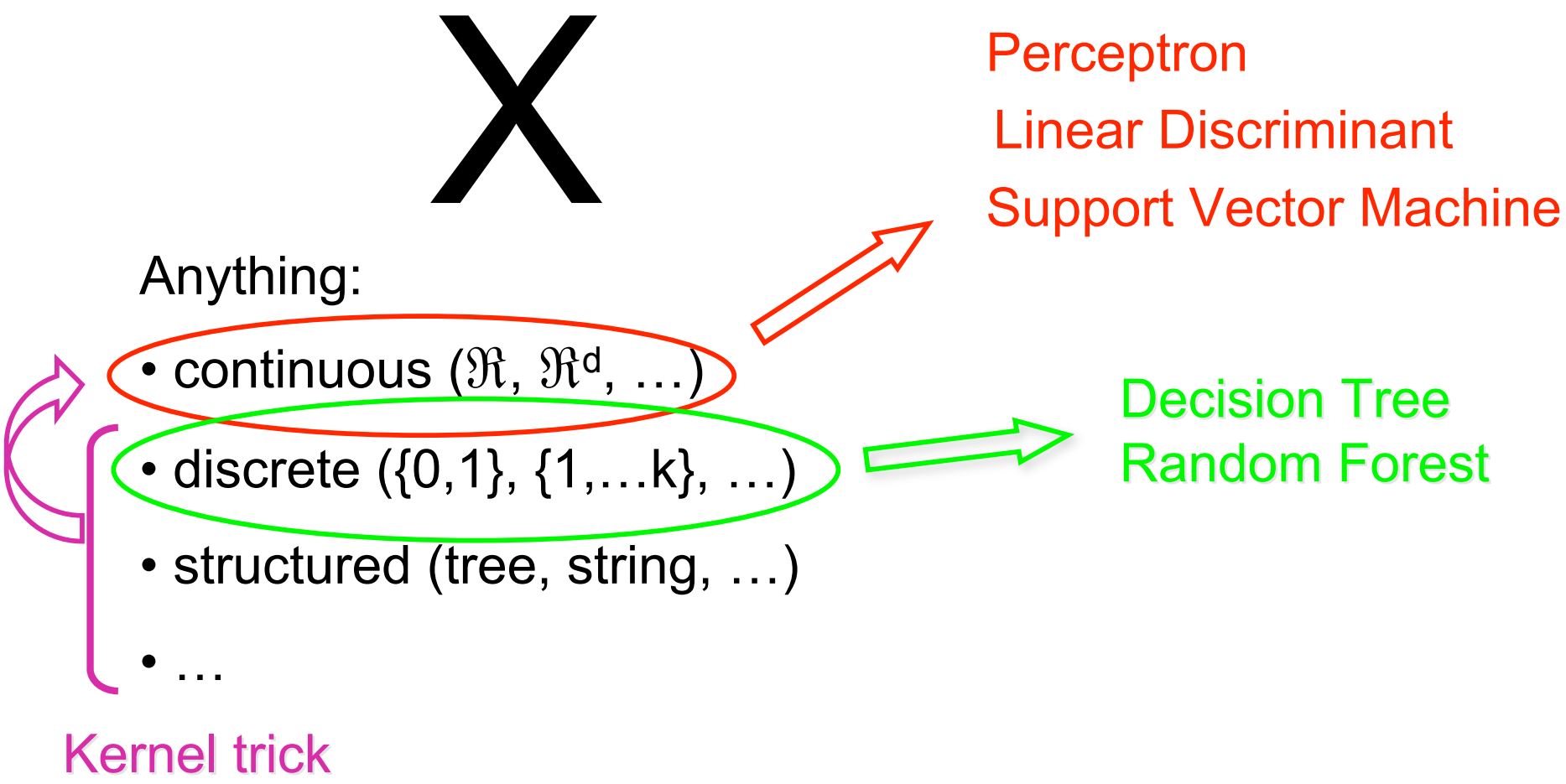
Classification (reminder)

X

Anything:

- continuous ($\Re$, $\Re^d$, …)
- discrete ({0,1}, {1,…k}, …)
- structured (tree, string, …)
- …

Classification (reminder)



Regression

X

Anything:

1 • continuous ($\Re$, $\Re^d$, …)

• discrete ({0,1}, {1,…k}, …)

• structured (tree, string, …)

• …

Y

• continuous:
  – $\Re$, $\Re^d$

Examples

- Voltage vs. Temperature
- Processes, memory vs. Power consumption
- Protein structure vs. Energy
- Robot arm controls vs. Torque at effector
- Location, industry, past losses vs. Premium

Preliminaries

Data $(x_1,y_1),\ldots (x_N,y_N)$. x_i is the predictor (regressor, covariate, feature, independent variable) y_i is the response (dependent variable, outcome)

We denote the *regression function* by

$$f(x)=E(Y|x)$$

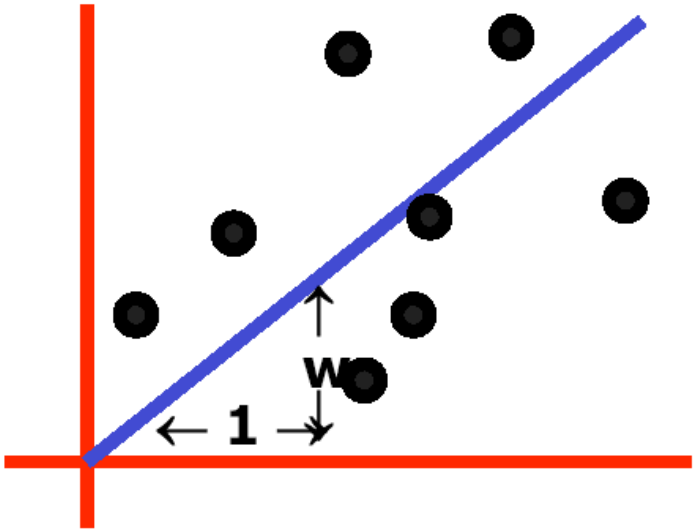
This is the conditional expectation of Y given x.

The linear regression model assumes a specific linear form for f

$$y=f(x)=\alpha+\beta x$$

which is usually thought of as an approximation to the truth.

Linear Regression



DATASET

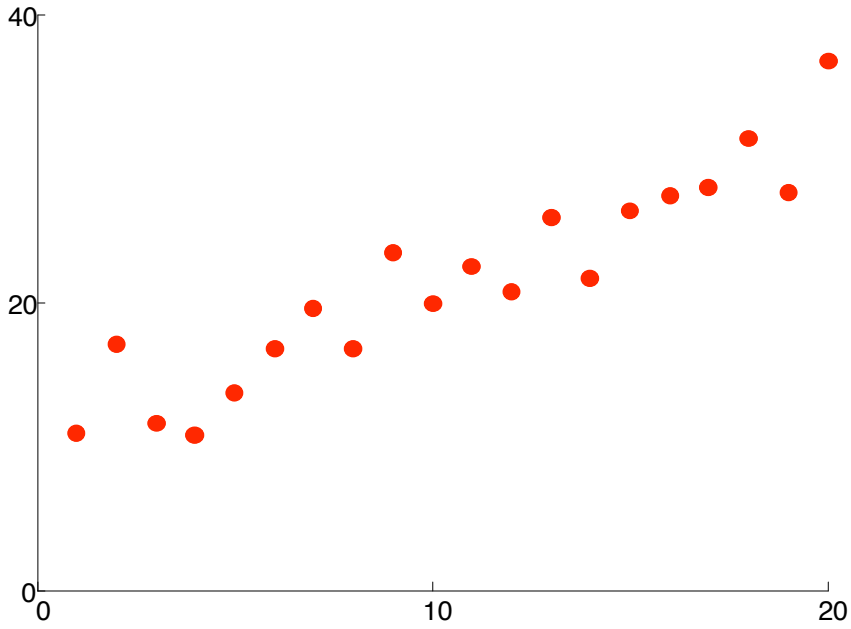
inputs	outputs
$x_1 = 1$	$y_1 = 1$
$x_2 = 3$	$y_2 = 2.2$
$x_3 = 2$	$y_3 = 2$
$x_4 = 1.5$	$y_4 = 1.9$
$x_5 = 4$	$y_5 = 3.1$

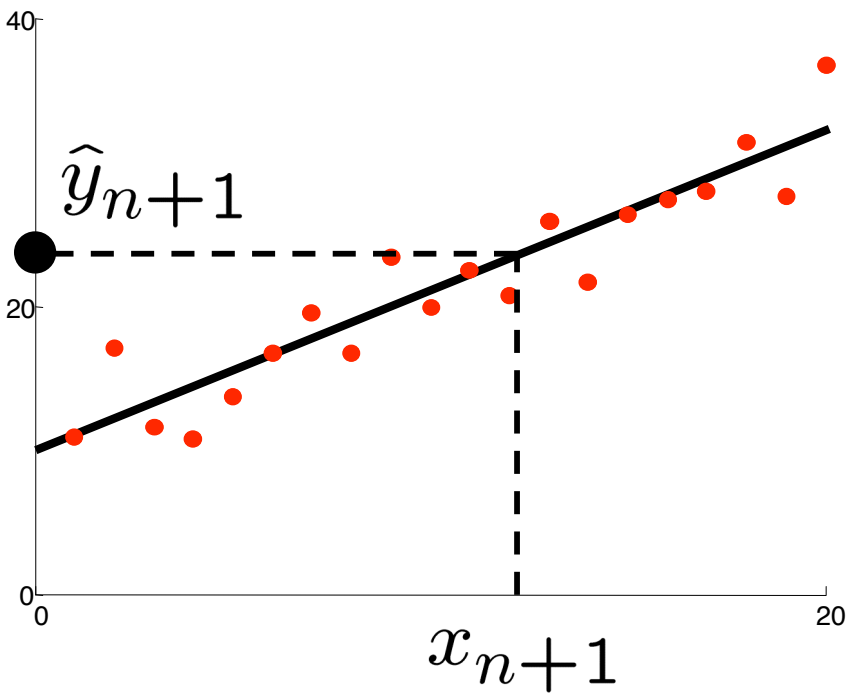
Linear regression assumes that the expected value of the output given an input, *E[y/x]*, is linear.

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

Given the data, we can estimate *w*.

Linear prediction





Given examples $(x_i, y_i)_{i=1...n}$

Predict y_{n+1} given a new point x_{n+1}

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

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

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

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

For what *w* is

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

For what w is

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

For what w is

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

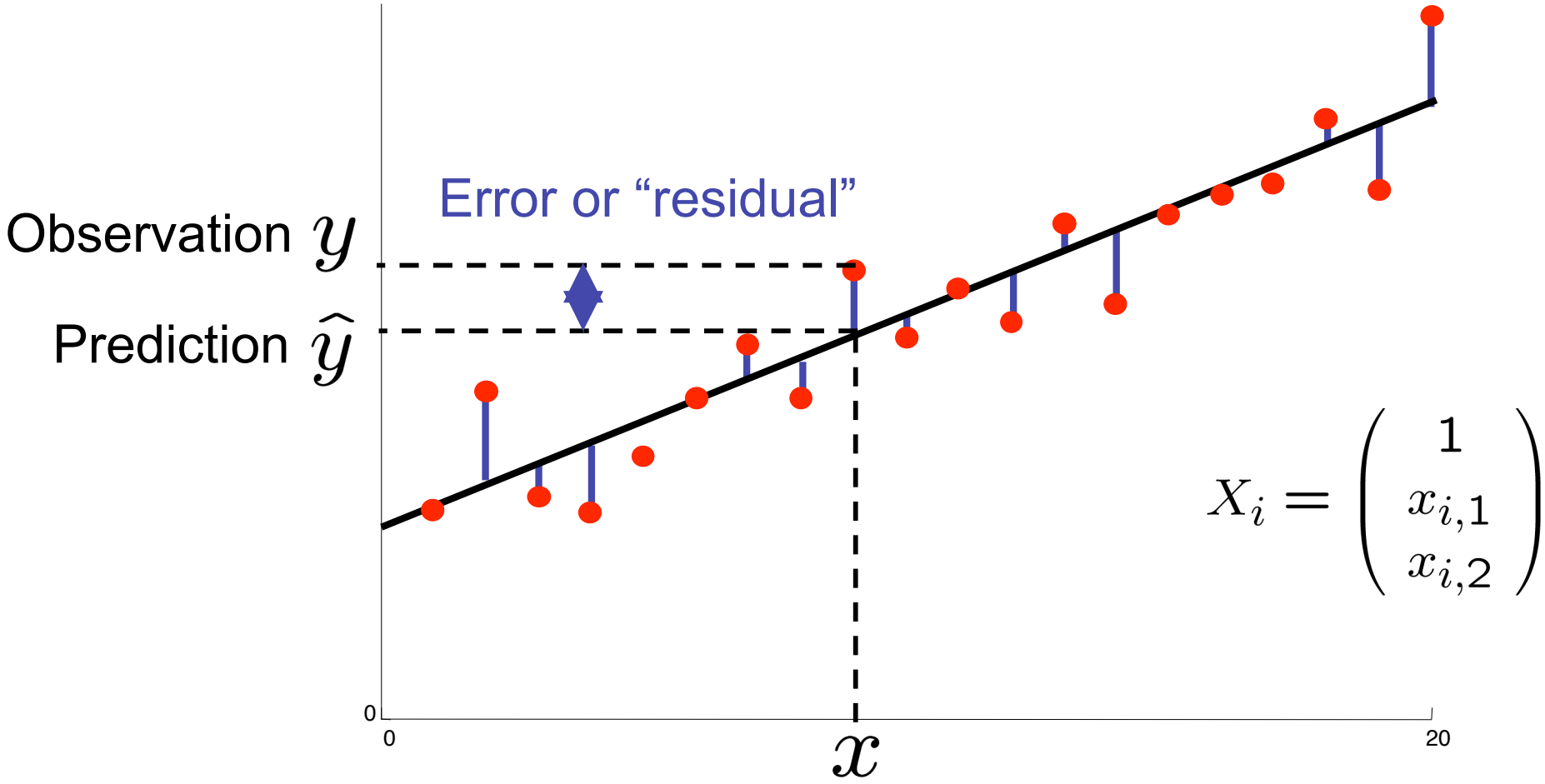
For what w is

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

For what w is

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

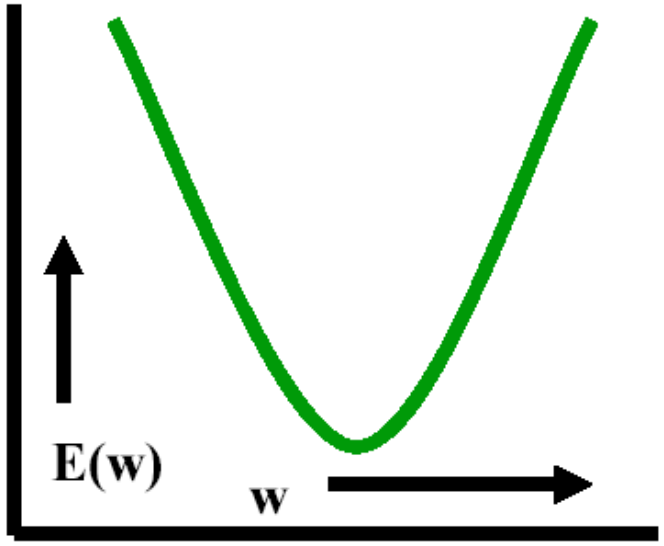
Ordinary Least Squares (OLS)



Sum squared error $\sum_i (X_i^\top w - y_i)^2$

Linear Regression

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



$$E = \sum_i (y_i - w x_i)^2$$

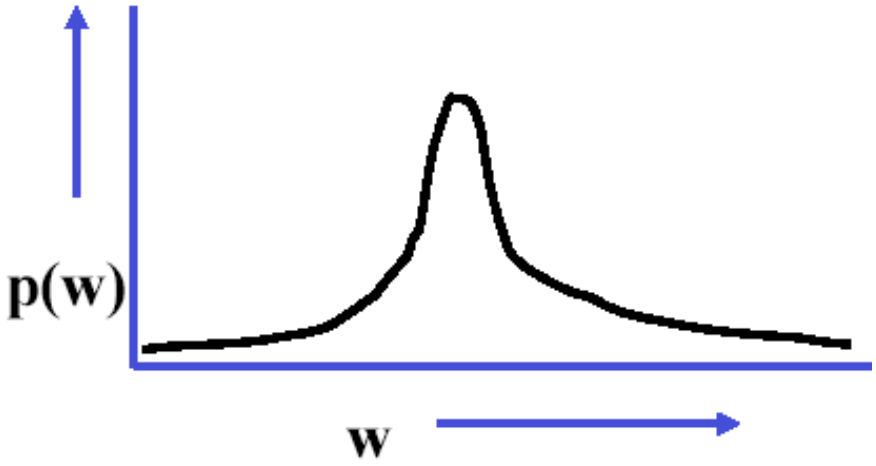
$$= \sum_i y_i^2 - \left(2 \sum x_i y_i\right) w + \left(\sum x_i^2\right) w^2$$

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

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 (error).

Max likelihood can give some kinds of confidence too.

Fitted Lines and coefficient errors

$$y_{pred}(x) = \hat{\beta}_0 + \hat{\beta}x$$

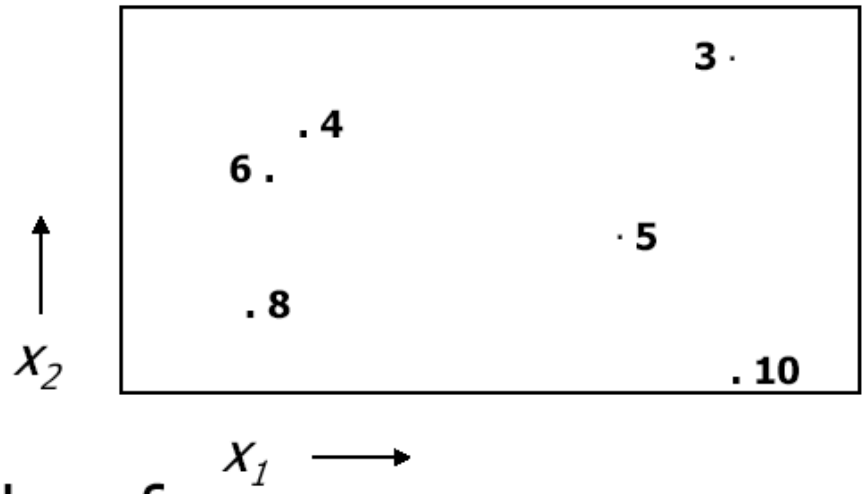
$$= mean(y) + \hat{\beta}\big(x - mean(x)\big)$$

$$err(y) = \Big[\mathrm{var}(mean(y)) + \mathrm{var}(\hat{\beta})(x - mean(x))^2\Big]$$

$$= \left[\frac{\sigma^2}{n} + \frac{\sigma^2(x-\bar{x})^2}{\sum(x_i-\bar{x})^2}\right]^{\frac{1}{2}}$$

Multivariate Regression

What if the inputs are vectors?



2-d input example

Dataset has form

$\boldsymbol{x}_1$	y_1
$\boldsymbol{x}_2$	y_2
$\boldsymbol{x}_3$	y_3
.:	:
.	
$\boldsymbol{x}_R$	y_R

Multivariate Regression

Write matrix X and Y thus:

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

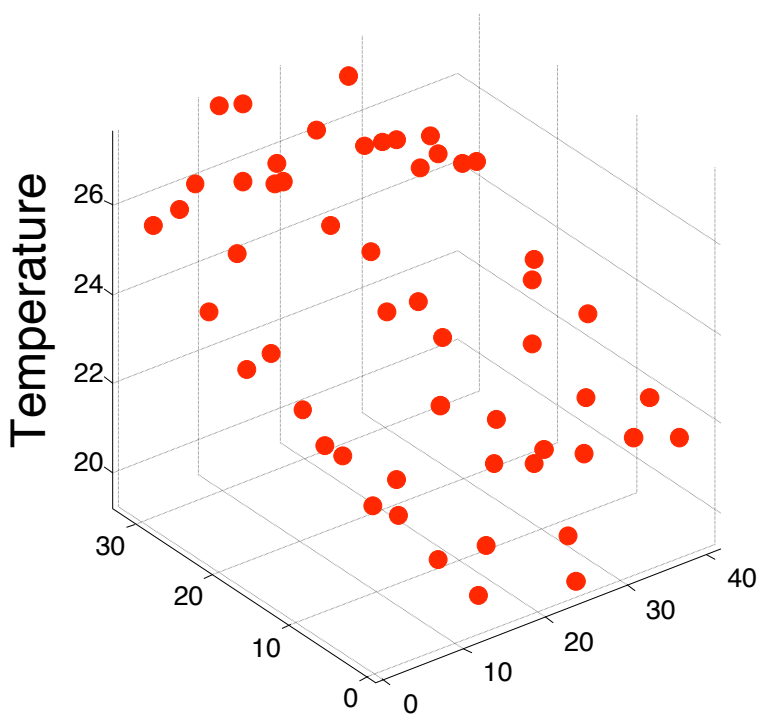
(there are *R* datapoints. Each input has *m* components)

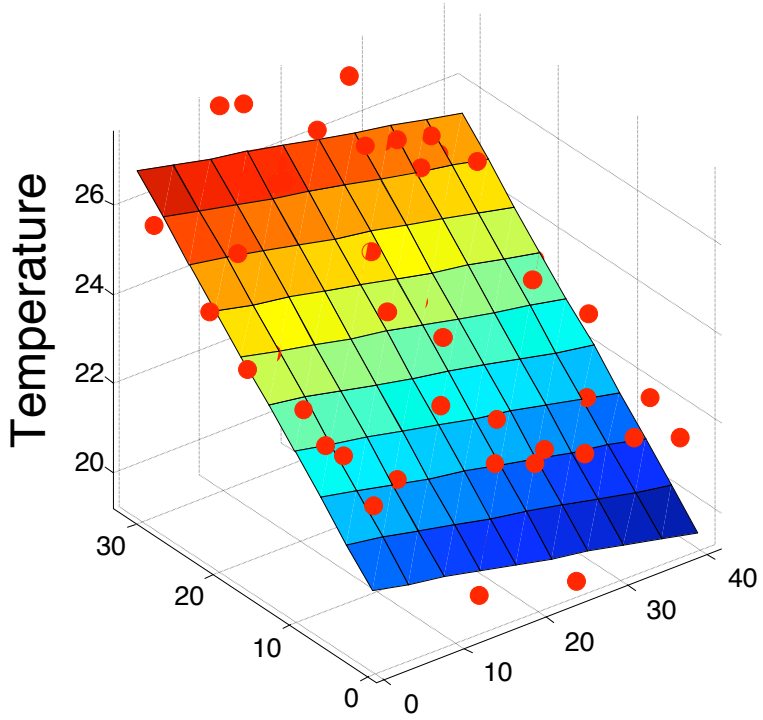
The linear regression model assumes a vector $\boldsymbol{w}$ such that

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

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

Linear regression





Prediction

$$\hat{y}_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$$

$$= \begin{pmatrix} 1 & x_{i,1} & x_{i,2} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$= X_i^\top w$$

Multivariate Regression (con't)

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

$X^T X$ is an m x 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 $\sum_{k=1}^{R} x_{ki} y_k$

Minimize the sum squared error

Sum squared error $E = \sum_i (X_i^\top w - y_i)^2$

$$\frac{\partial E}{\partial w_j} = \sum_i \frac{\partial}{\partial w_j}(X_i^\top w - y_i)^2$$

$$= \sum_i 2X_{i,j}(X_i^\top w - y_i)$$

$$\frac{\partial E}{\partial w_j} = 0 \iff \left(\sum_i X_{i,j} X_i^\top\right) w = \sum_i y_i X_{i,j}$$ Linear equation

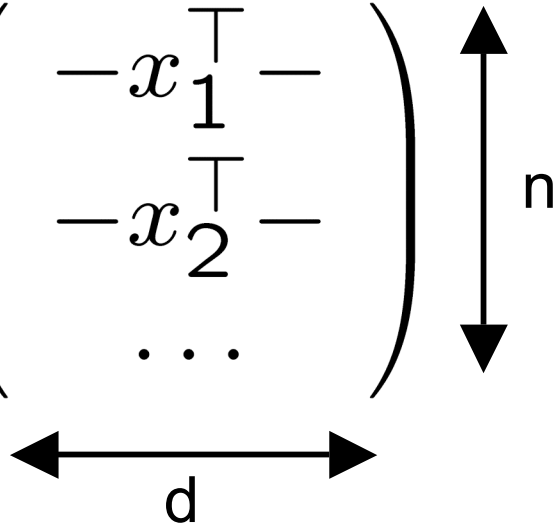
$$\frac{\partial E}{\partial w} = 0 \iff \underbrace{\left(\sum_i X_i X_i^\top\right)}_{A} w = \underbrace{\sum_i y_i X_i}_{b}$$ Linear system

$$\boxed{Aw = b}$$

Alternative derivation

$$
\begin{aligned}
E &= \sum_i (X_i^\top w - y_i)^2 = \|Xw - y\|_2^2 \\
&= w^\top \underbrace{X^\top X}_{A} w - 2 \underbrace{y^\top X}_{b^\top} w + \|y\|_2^2
\end{aligned}
$$

$$
X = \begin{pmatrix} -x_1^\top - \\ -x_2^\top - \\ \cdots \end{pmatrix}
$$



$$
\frac{\partial E}{\partial w} = 2Aw - 2b
$$

$$
Aw = b
$$

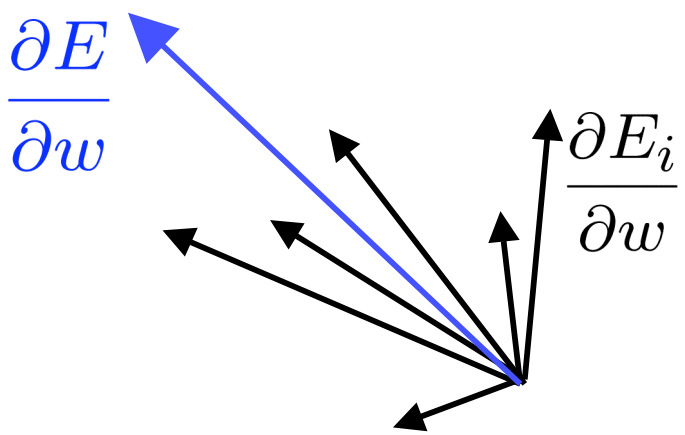
Solve the system (it's better not to invert the matrix)

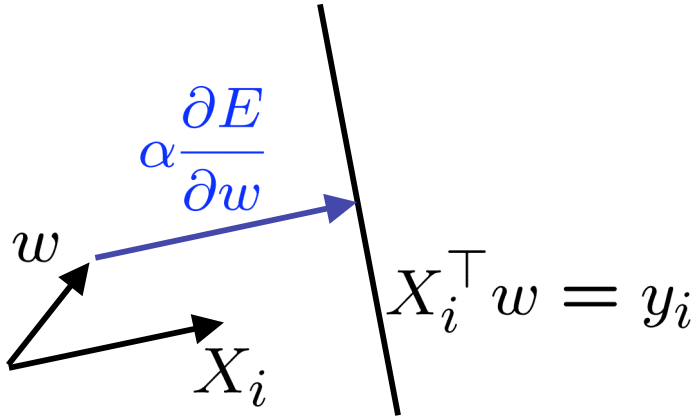
LMS Algorithm

(Least Mean Squares)

$$E = \sum_i (X_i^\top w - y_i)^2 = \sum_i E_i$$

$$\frac{\partial E}{\partial w} = \sum_i \frac{\partial E_i}{\partial w} \qquad \text{where} \qquad \begin{aligned} \frac{\partial E_i}{\partial w} &= \frac{\partial}{\partial w}(X_i^\top w - y_i)^2 \\ &= 2X_i(X_i^\top w - y_i) \end{aligned}$$





Online algorithm $w^{t+1} = w^t + \alpha X_i(y_i - X_i^\top w^t)$

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

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:

$Y= w_0X_0+w_1X_1+ w_2X_2$

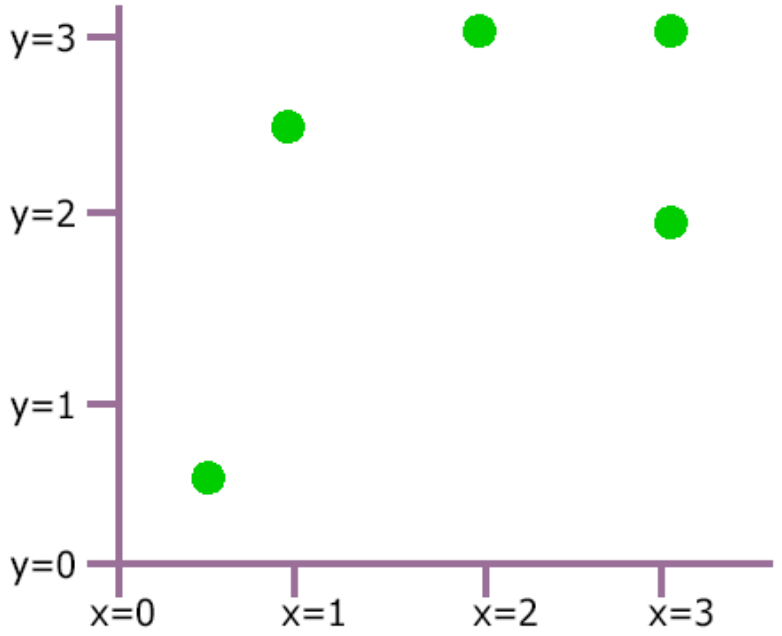
$= w_0+w_1X_1+ w_2X_2$

...has a fine constant term

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

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

$$\left(\overline{v + x_i} \right)^2 =$$

$$\left(\overline{+ x_i} = 0 \right) =$$

Common (but not only) approach:

Numerical Solutions:

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

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

Setting dLL/dw equal to zero

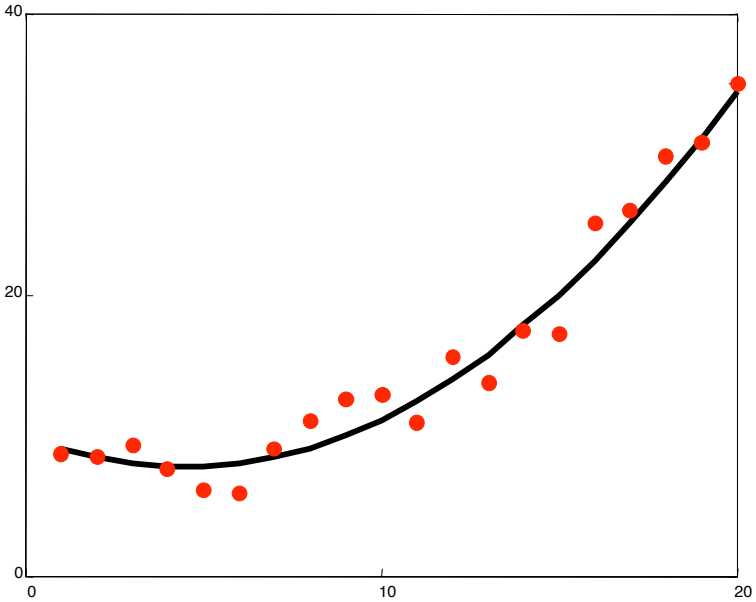
We're down the algebraic toilet

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

So guess what we do?

What to do?

- Same solution as classification
  - Change the feature space
  - For Example - ***Quadratic regression***



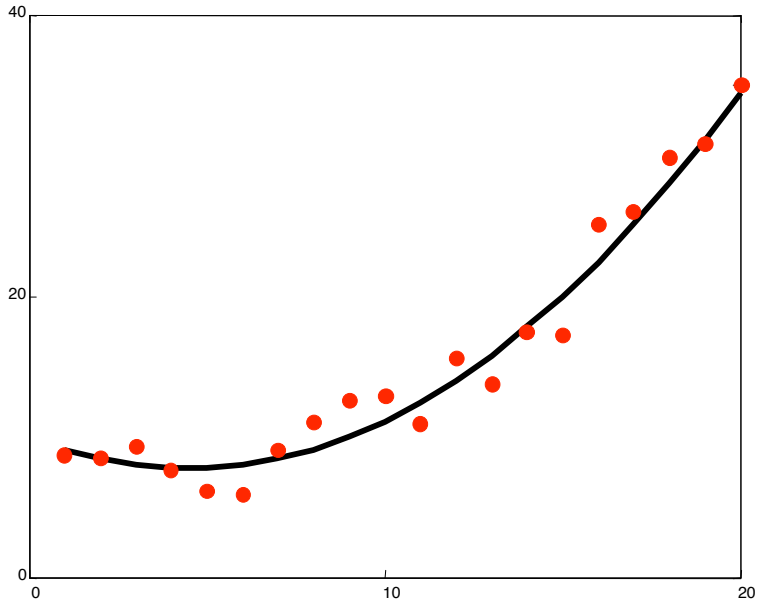
$$\hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2$$

still linear in w

everything is the same with $X_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$

Beyond lines and planes

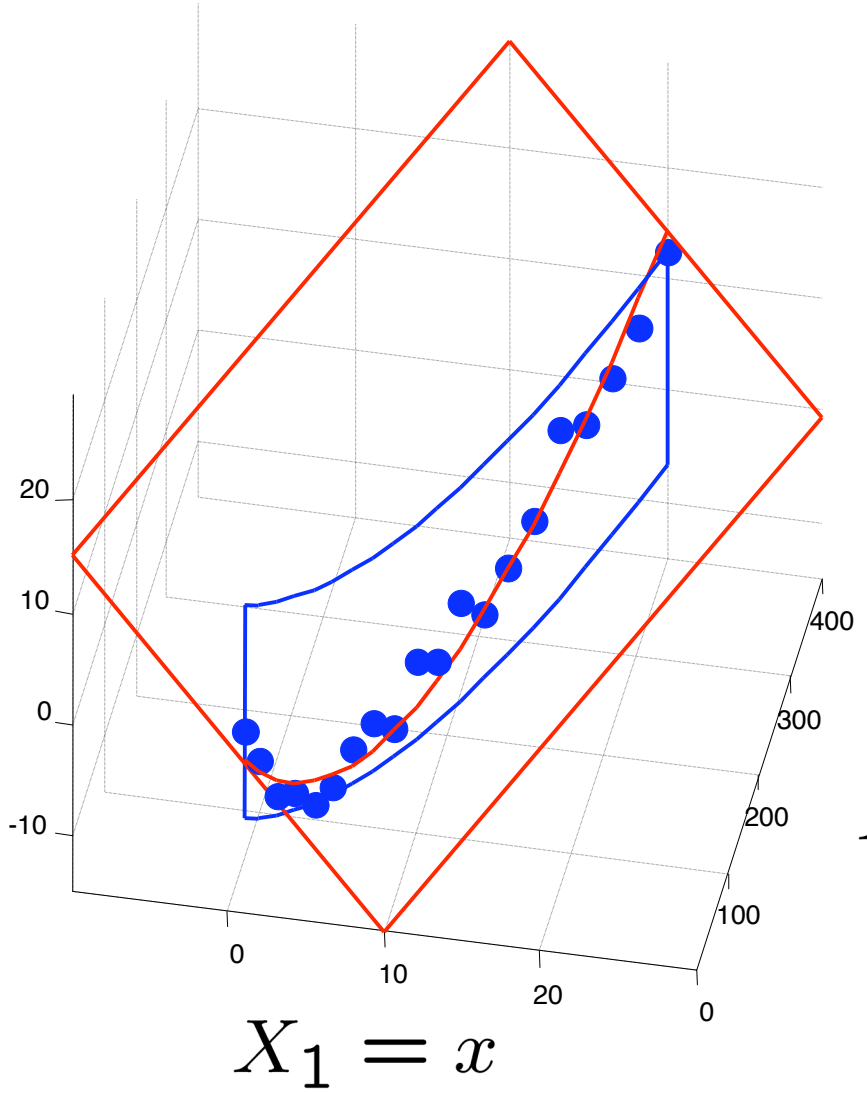
$$\hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2$$



still linear in w

everything is the same with $X_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$

Geometric interpretation



$$\hat{y} = w_1 x + w_2 x^2$$

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
:	:	:

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

$y_1=7..$

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

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

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

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

Quadratic Regression

It's triv

X_1	X_2
3	2
1	1
:	:

$\mathbf{z}=$

1
1
:

$\mathbf{z}=(1$

Each component of a z vector is called a term.

Each column of the Z matrix is called a term column

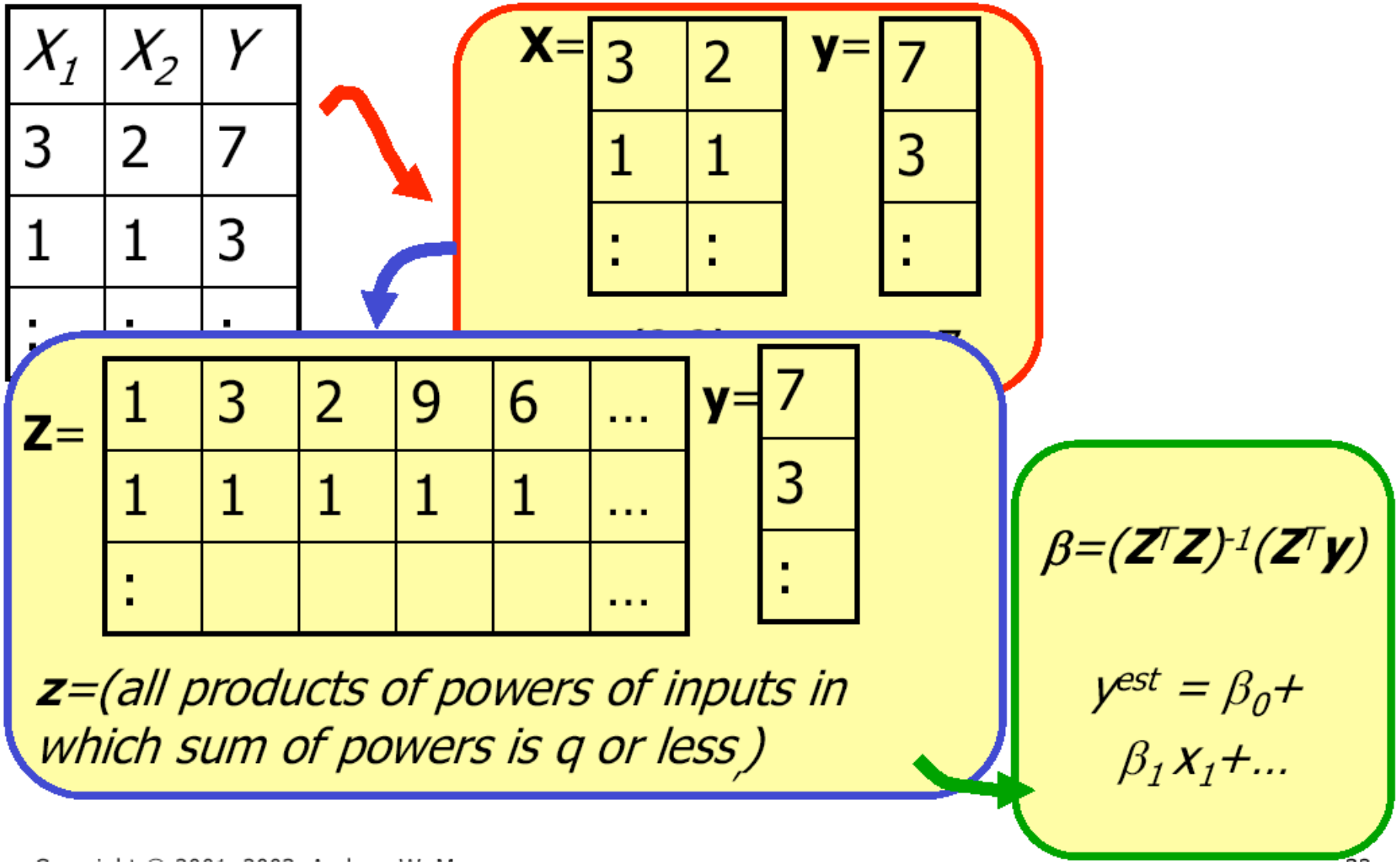
How many terms in a quadratic regression with *m* inputs?

- 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



m inputs, degree Q: how many terms?

= the number of unique terms of the form

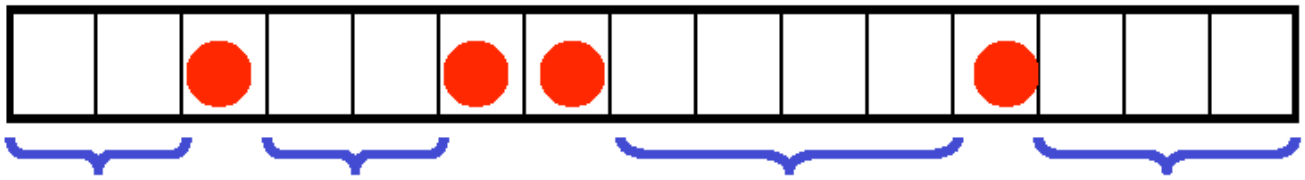
$$x_1^{q_1} x_2^{q_2} \ldots x_m^{q_m} \text{ where } \sum_{i=1}^{m} q_i \le Q$$

= 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 $\Sigma q_i = Q$

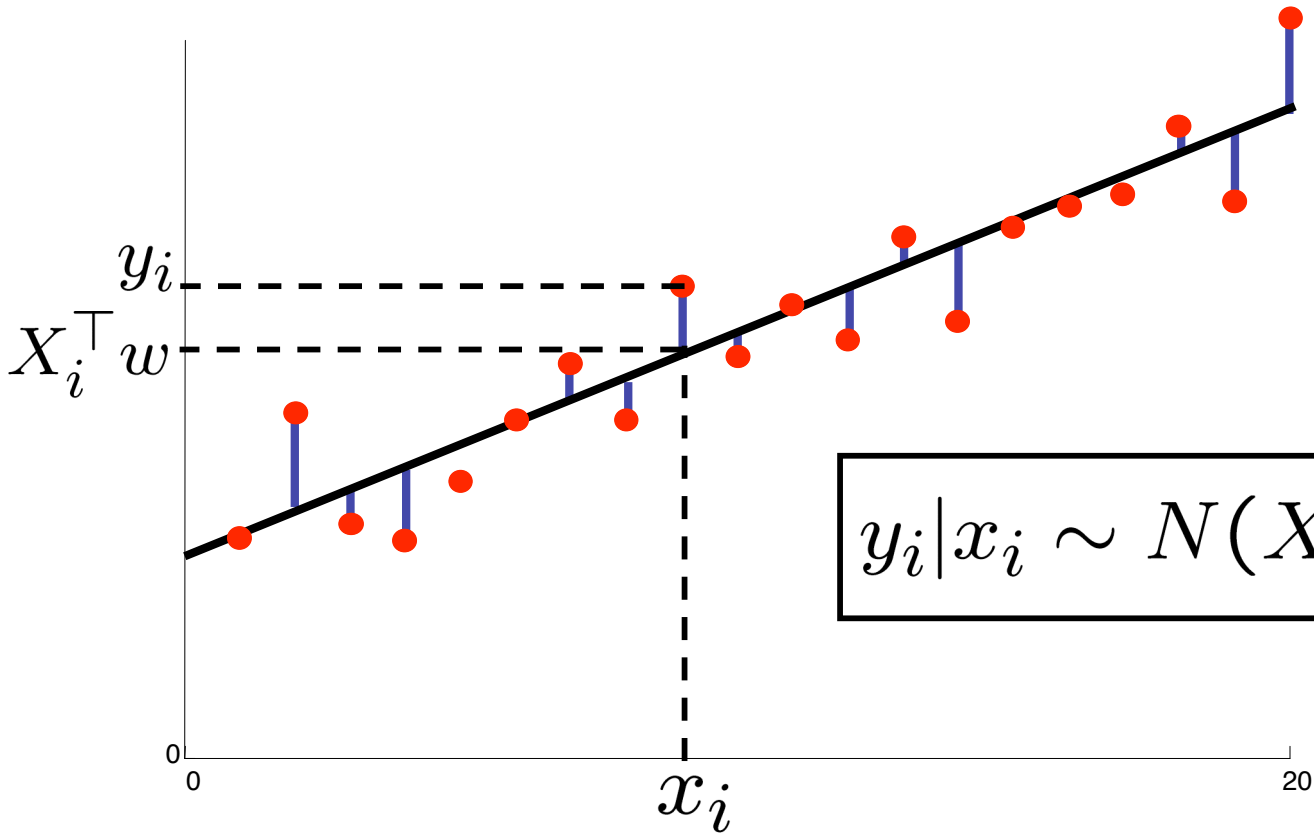
= the number of ways of placing Q red disks on a row of squares of length Q+m = (Q+m)-choose-Q



Q=11, m=4

$q_0=2$ $q_1=2$ $q_2=0$ $q_3=4$ $q_4=3$

Probabilistic interpretation

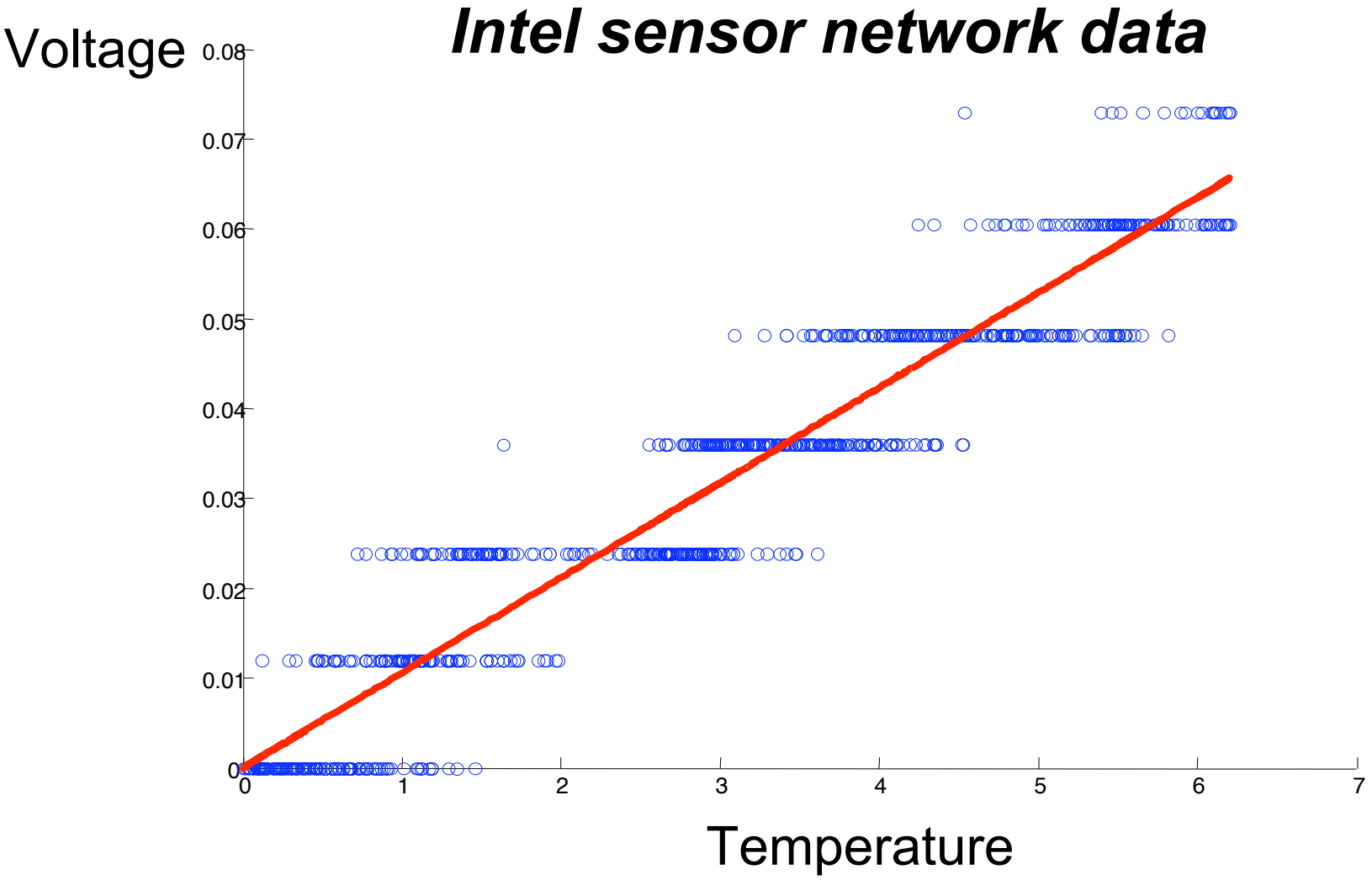


$$y_i | x_i \sim N(X_i^\top w, \sigma^2)$$

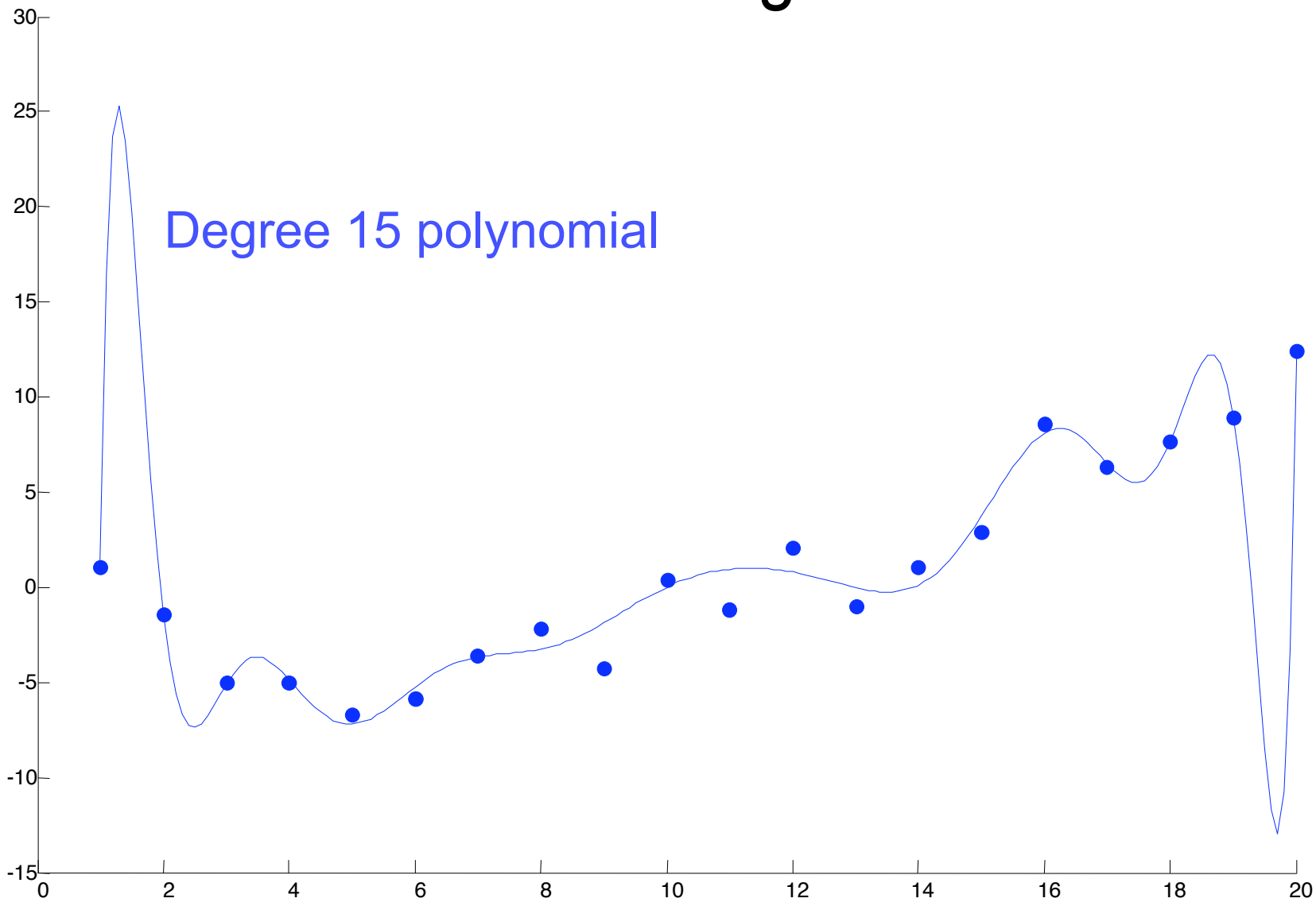
Likelihood $L = \prod_i \exp -\frac{1}{2\sigma^2}(X_i^\top w - y_i)^2 = \exp -\frac{1}{2\sigma^2}\sum_i (X_i^\top w - y_i)^2$

$$\underset{w}{\operatorname{argmax}}\, L = \underset{w}{\operatorname{argmin}}\, E$$

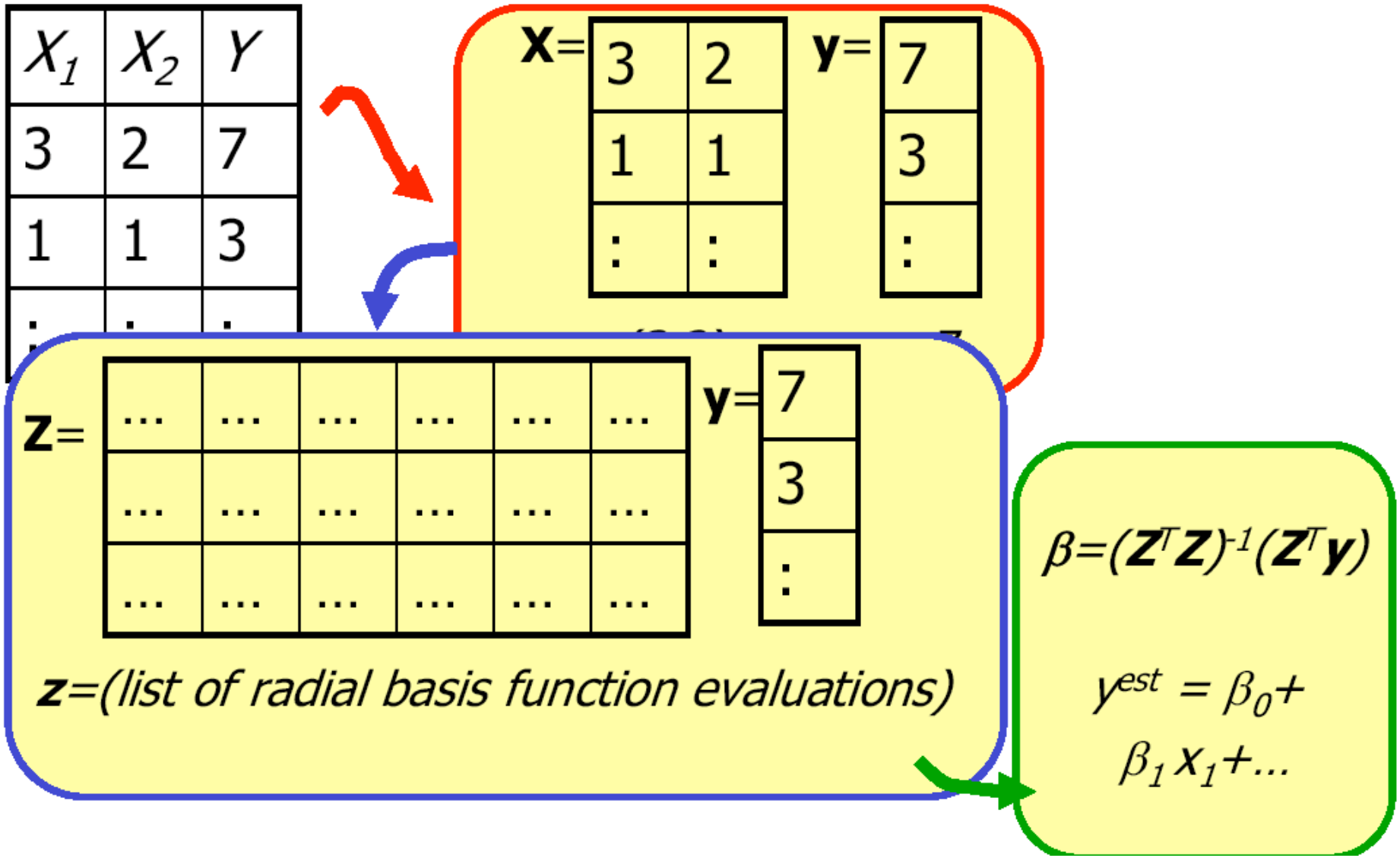
Assumptions vs. Reality



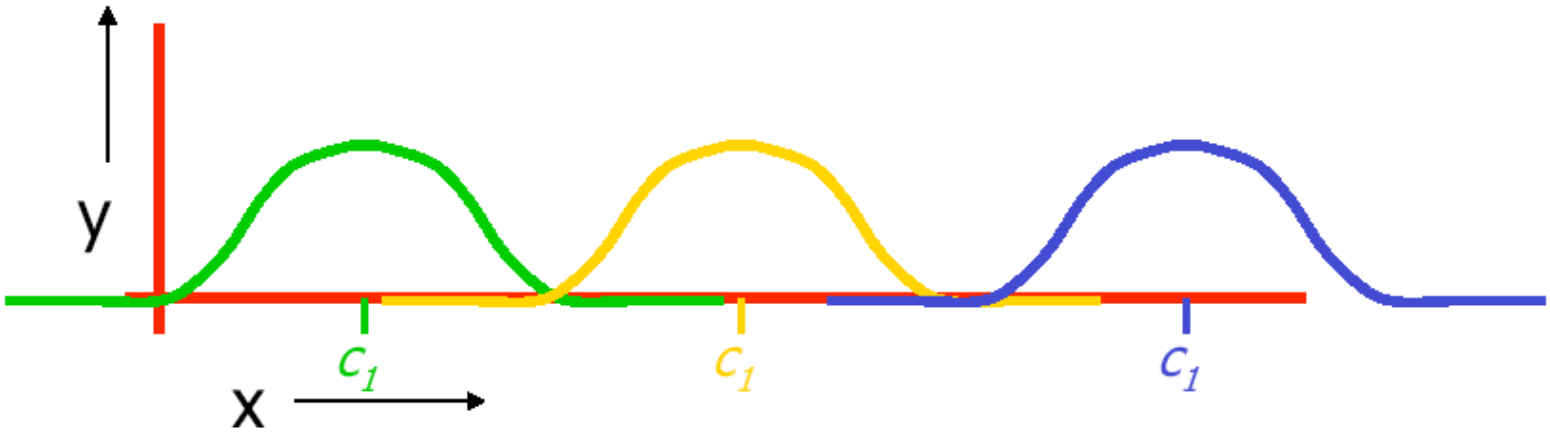
Overfitting



Radial Basis Functions (RBFs)



1-d RBFs

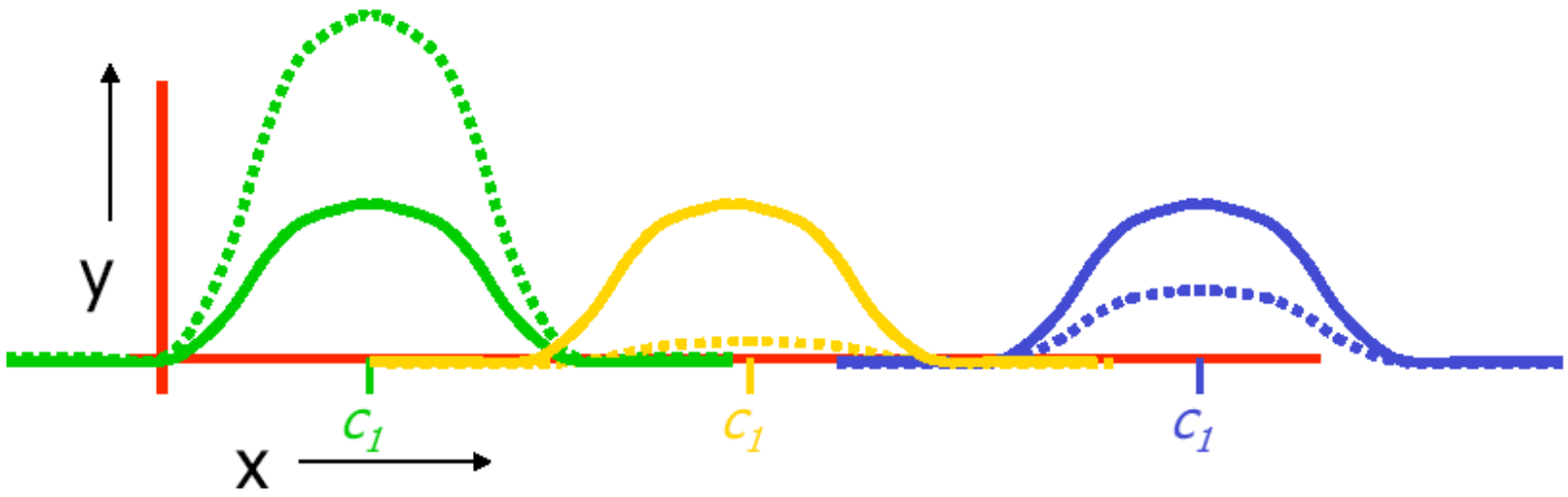


$$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$$

where

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

Example

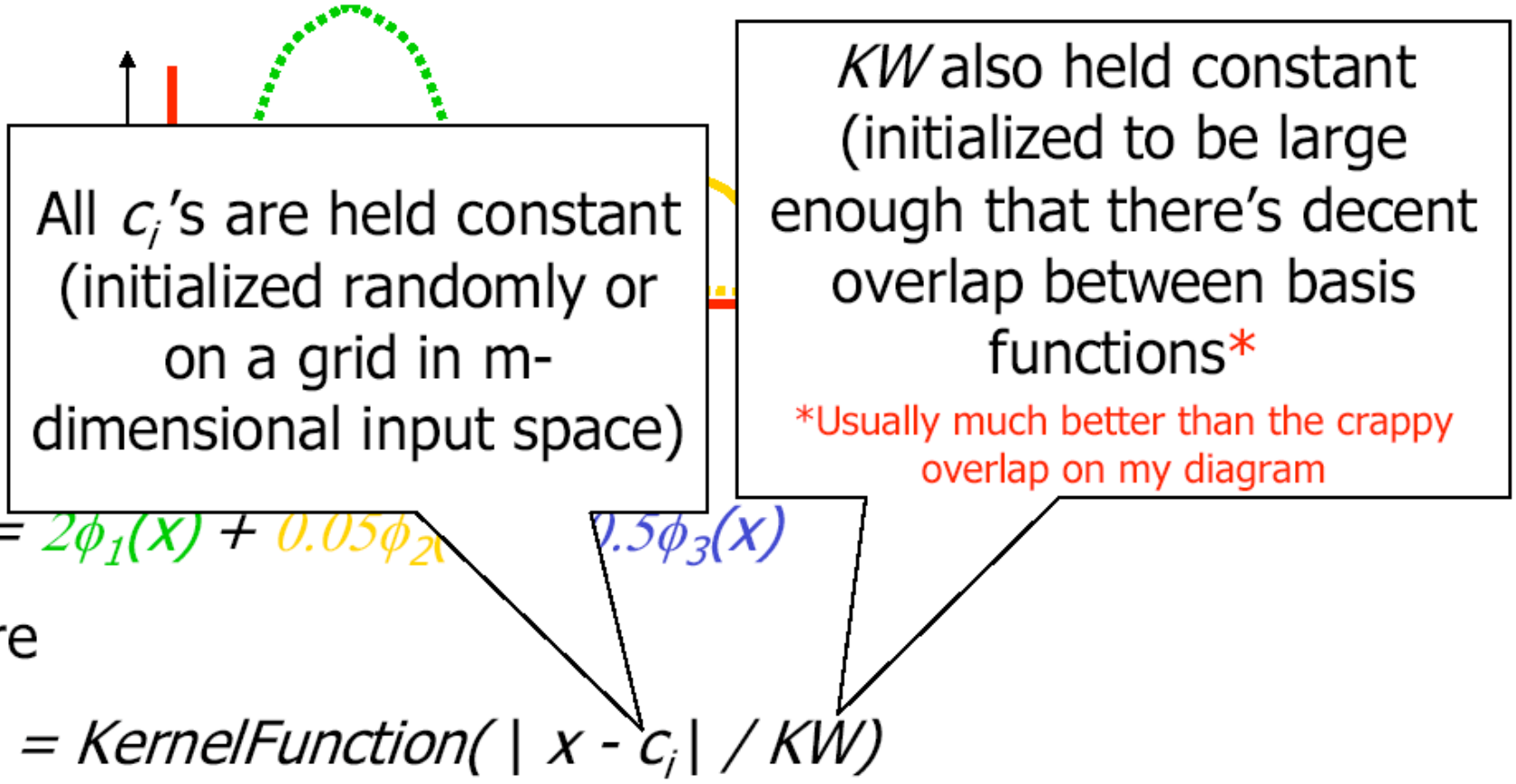


$$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$$

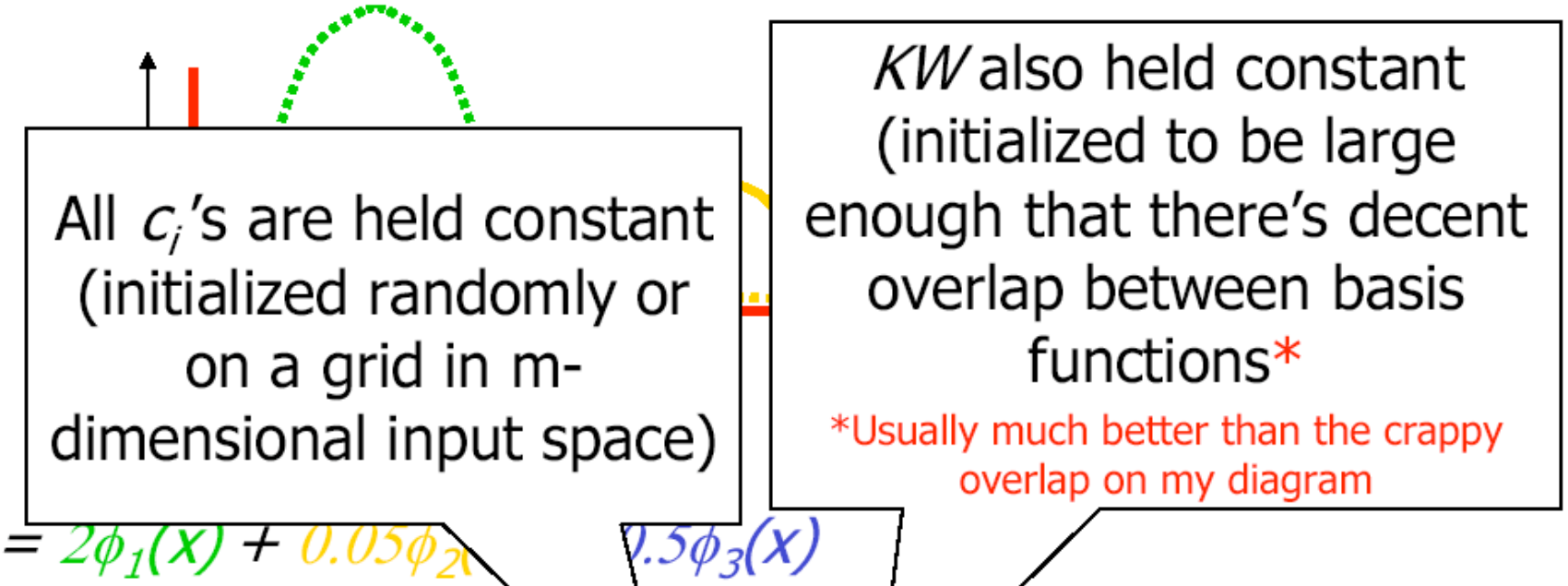
where

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

RBFs with Linear Regression



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

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

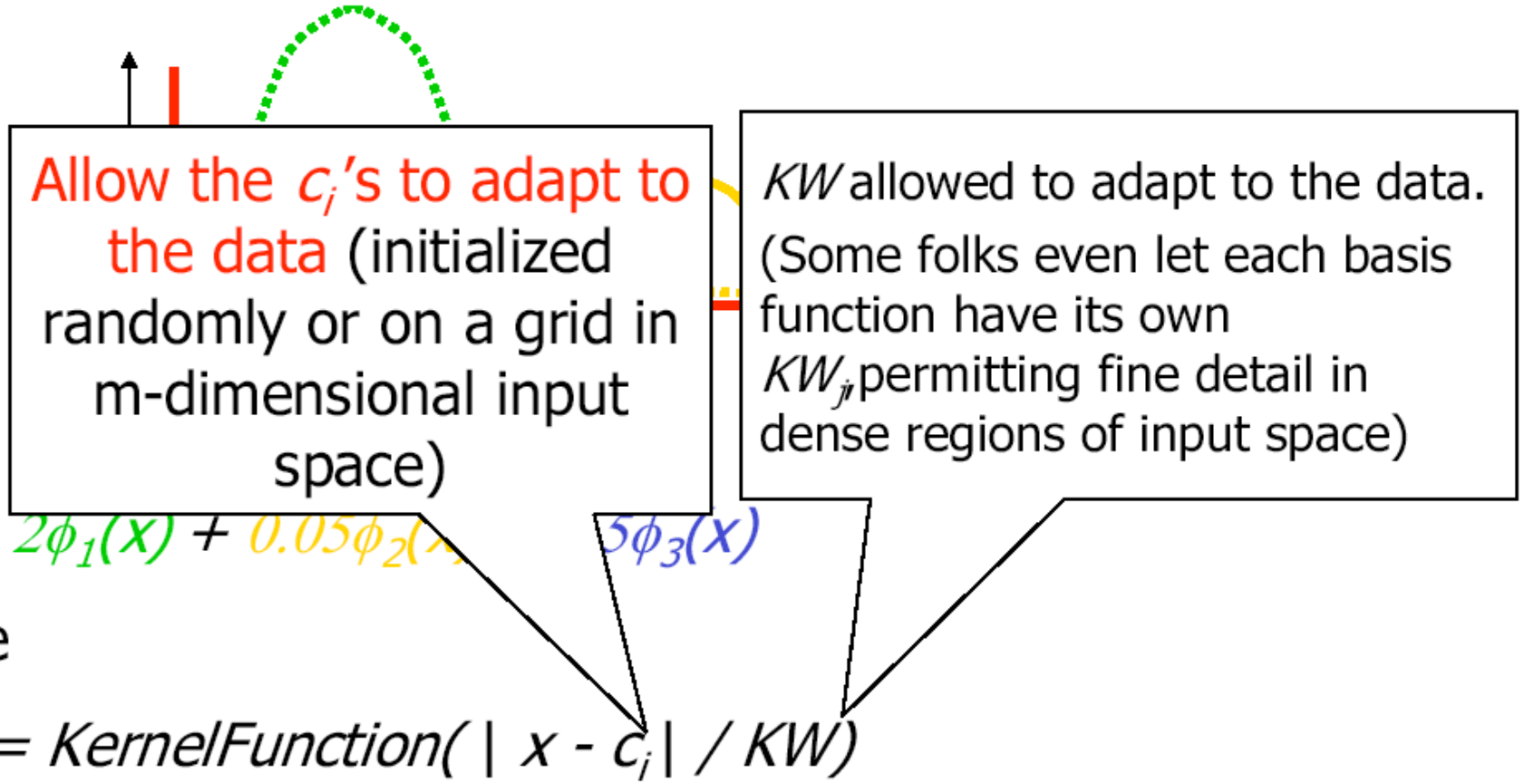
where

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

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

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

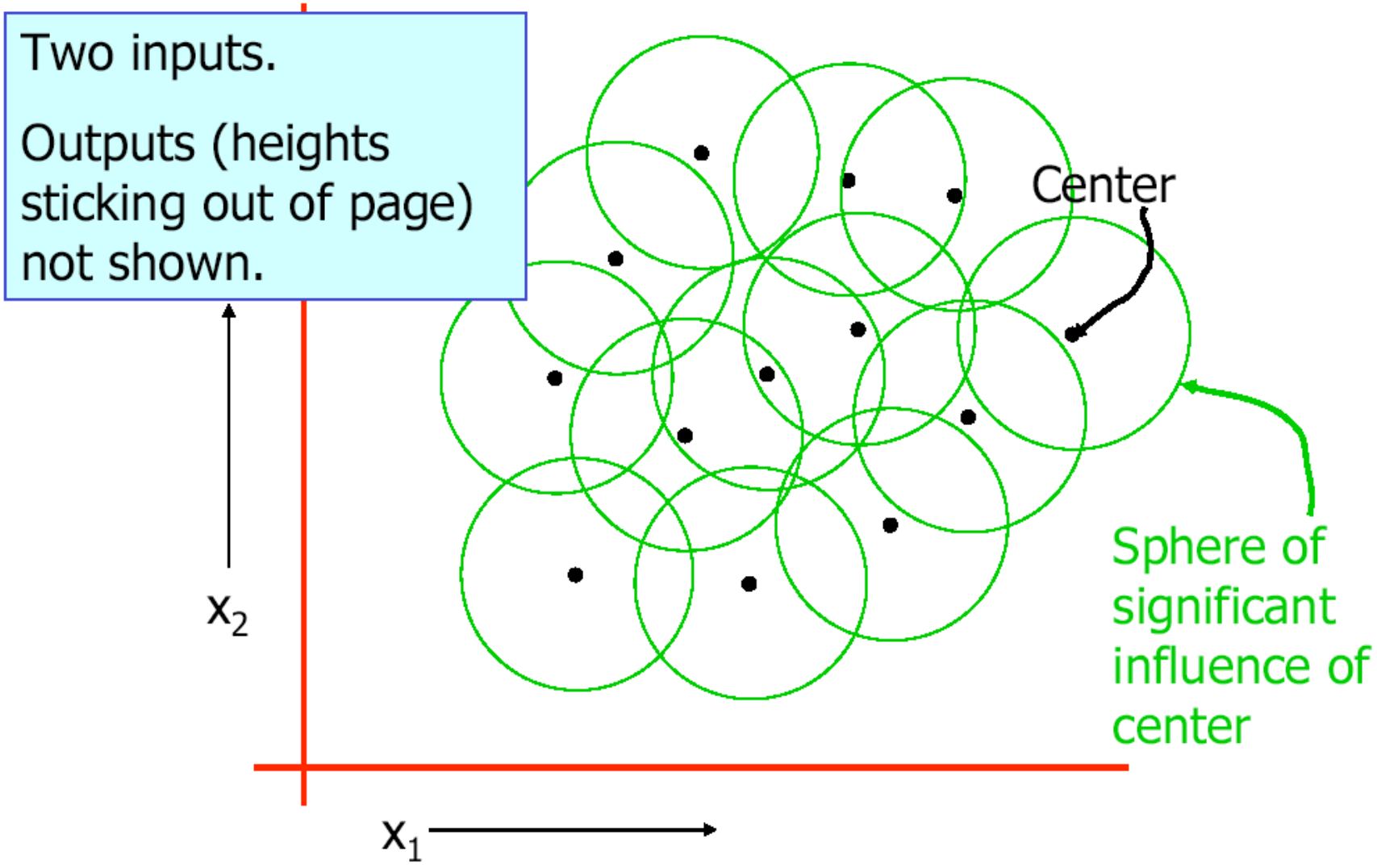
RBFs with NonLinear Regression



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

ANSWER: Gradient Descent

Radial Basis Functions in 2-d



Ordinary Least Squares Summary

Given examples $(x_i, y_i)_{i=1\ldots n}$

Let $X_i^\top = (f_1(x_i) \quad f_2(x_i) \quad \ldots \quad f_d(x_i))$

For example $X_i^\top = \begin{pmatrix} 1 & x_{i,1} & x_{i,2} & x_{i,1}^2 & x_{i,2}^2 & x_{i,1}x_{i,2} \end{pmatrix}$

Let $X = \begin{pmatrix} -X_1^\top- \\ -X_2^\top- \\ \ldots \end{pmatrix}$ n $\qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \end{pmatrix}$

d

Minimize $\|Xw - y\|_2^2$ by solving $\left(X^\top X\right) w = X^\top y$

Predict $\hat{y}_{n+1} = X_{n+1}^\top w$

Statistical Models

$$Y = f(X) + \varepsilon$$

with $\mathrm{E}(\varepsilon) = 0$ and X and ε independent.

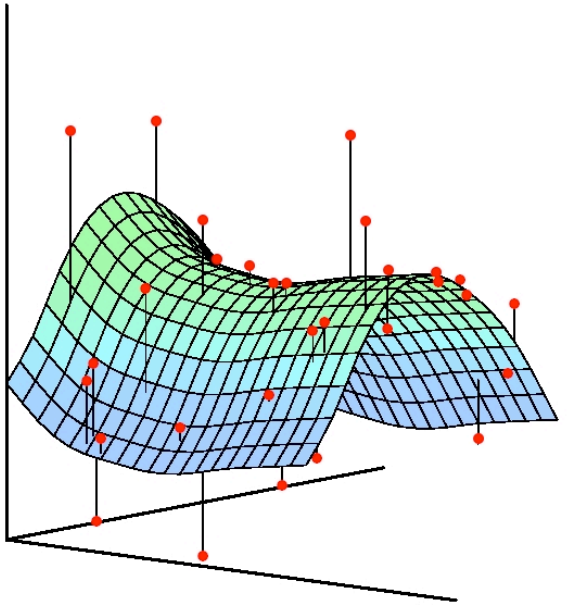
- $\mathrm{E}(Y|X) = f(X)$
- $\Pr(Y|X)$ depends on X only through $f(X)$.
- Useful approximation to the truth — all unmeasured variables captured by ε
- N realizations $y_i = f(x_i) + \varepsilon_i,\ i = 1, \ldots, N$
- Assume ε_i and ε_j are independent.

More generally can have, for example, $\mathrm{Var}(Y|X) = \sigma^2(X)$.

For qualitative outcomes $\{\Pr(G = \mathcal{G}_k|X)\}_1^K = p(X)$ which we model directly.

Function Approximation

$$\text{RSS}(\theta) = \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2$$



Assumes

- $x_i,\ y_i$ are points in, say $\mathbb{R}^{p+1}$.
- A (parametric) form for $f(X) = f_\theta(X)$, where θ is a collection of parameters, and can be quite complex.
- A loss function for measuring the quality of the approximation.

More generally, *Maximum Likelihood Estimation* (R.A Fisher, page 467) provides a natural basis for estimation. E.g. multinomial

$$\begin{aligned} \Pr(G = k|X) &= p_{k,\theta}(X) \\ \ell(\theta) &= \sum_{i=1}^{N} \log \Pr_{g_i,\theta}(x_i) \end{aligned}$$

Structured Regression Models

$$\text{RSS}(f) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Any function passing through (x_i, y_i) has $RSS = 0$
- Need to restrict the class
- Usually restrictions impose local behavior — see equivalent kernels in chapters 5 and 6
- Any method that attempts to approximate locally varying functions is "cursed"
- Alternatively, any method that "overcomes" the curse, assumes an implicit metric that does not allow neighborhoods to be simultaneously small in all directions.

Classes of Restricted Estimators

Some of the classes of restricted methods that we cover are

- Roughness Penalty and Bayesian Methods

$$\text{PRSS}(f, \lambda) = \text{RSS}(f) + \lambda J(f)$$

- Kernel Methods and Local Regression

$$\text{RSS}(f_\theta, x_0) = \sum_{i=1}^{N} K_\lambda(x_0, x_i)(y_i - f_\theta(x_i))^2$$

- Basis functions and dictionary methods

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

Model Selection and the Bias-Variance Tradeoff

Many of the flexible methods have a *complexity parameter:*

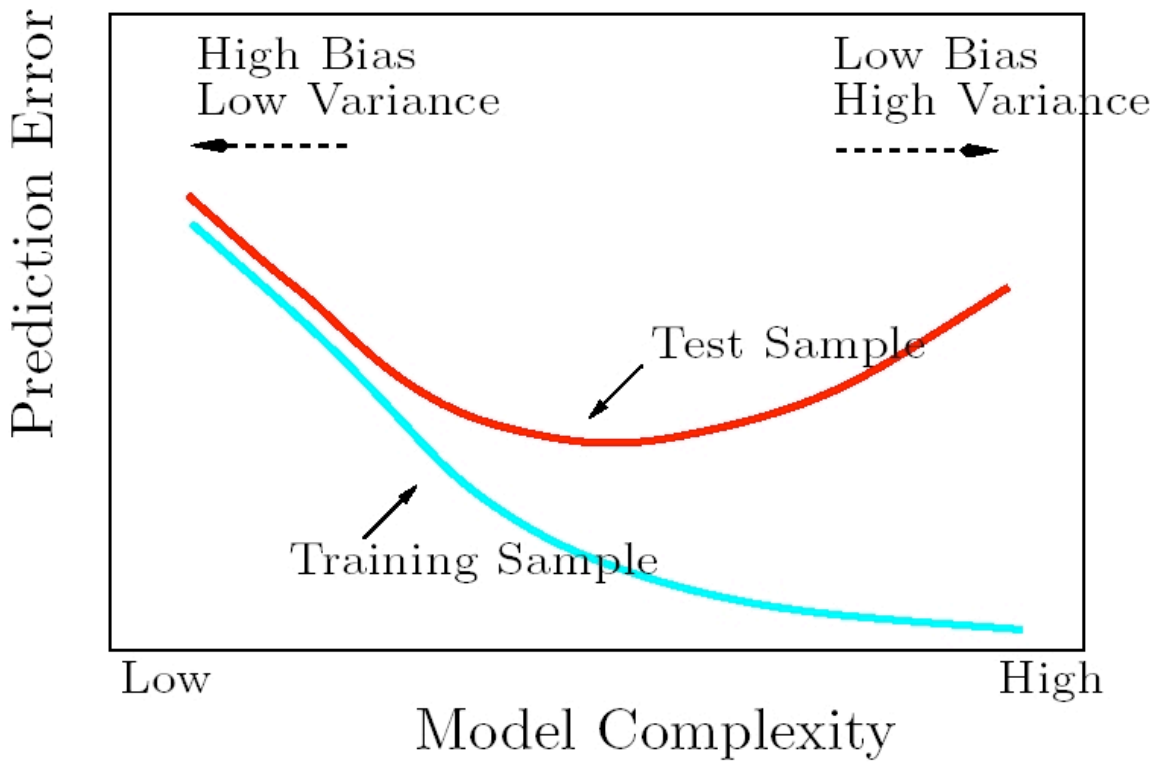
- the multiplier of the penalty term
- the width of the kernel
- the number of basis functions

Cannot use RSS to determine this parameter — why?

Can use *Prediction error* on unseen test cases to guide us

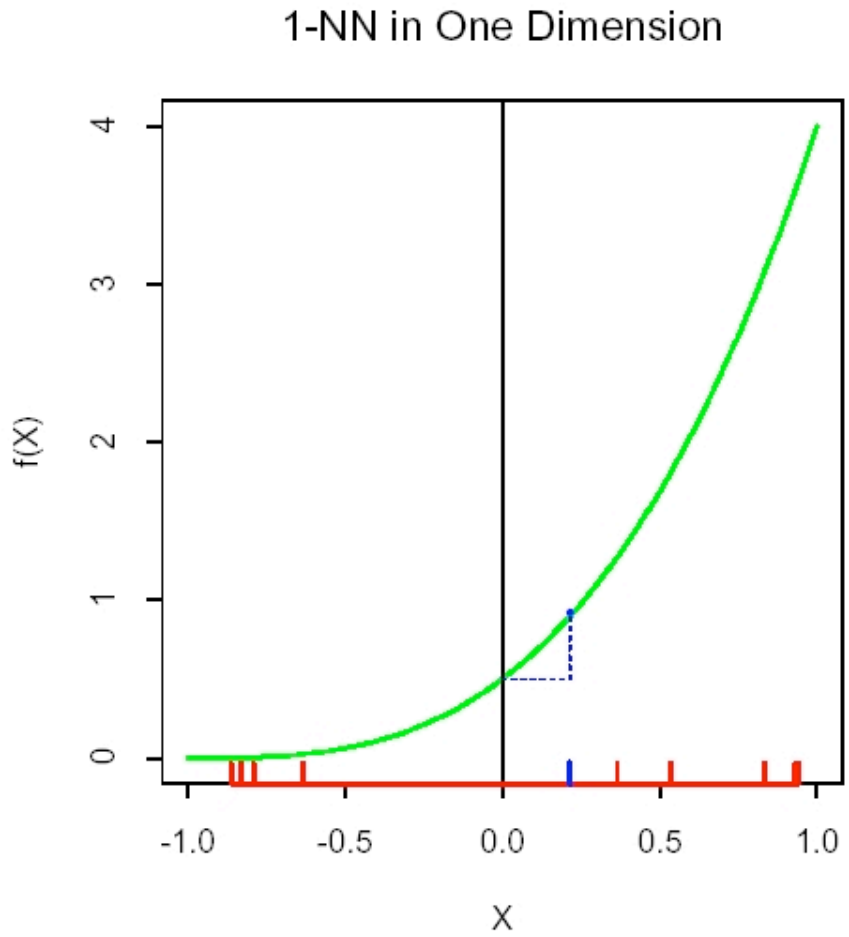
E.g. $Y = f(X) + \varepsilon$, K-nn (and assume the sample x_i are fixed):

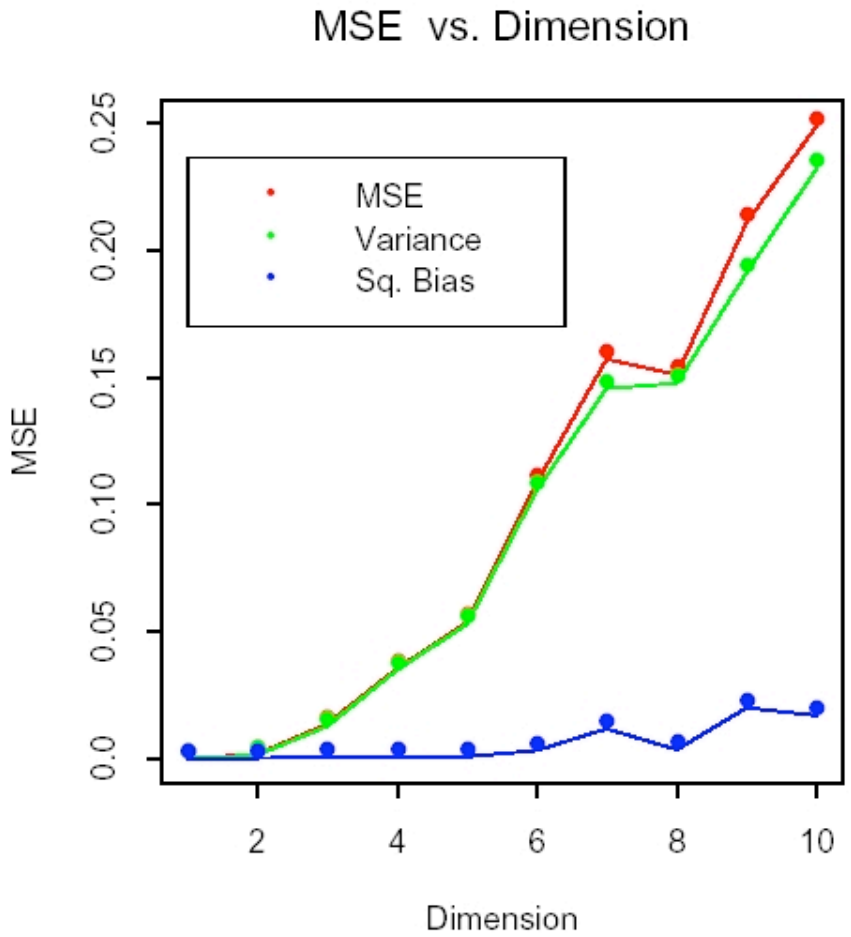
$$\begin{aligned}
\mathrm{E}[(Y - \hat{f}_k(x_0))^2 | X = x_0] &= \sigma^2 + \mathrm{Bias}^2(\hat{f}_k(x_0)) + \mathrm{Var}_{\mathcal{T}}(\hat{f}_k(x_0)) \\
&= \sigma^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})]^2 + \frac{\sigma^2}{k}
\end{aligned}$$



Selecting k amounts to a *bias-variance* tradeoff.

- Decreasing k increases the variance, but decreases bias
- $\cdots$ and vice versa





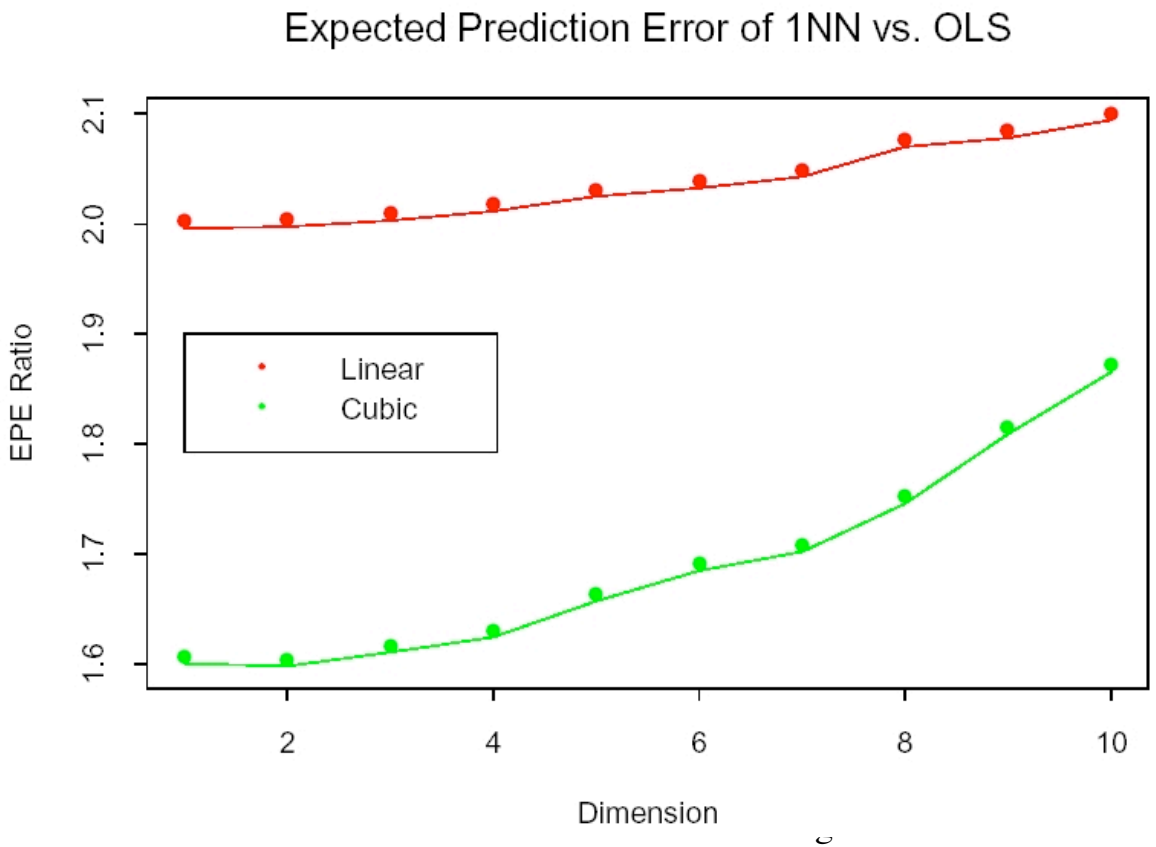
Here $f(X) = \frac{1}{2}(X_1 + 1)^3$. Here we have

- $\mathrm{E}\hat{f}(x_0) = \mathrm{E}f(X_{(1)}) \approx f(\mathrm{E}X_{(1)}) = f(x_0)$ for all dimensions
- $\mathrm{Var}\hat{f}(x_0) = \mathrm{Var}f(X_{(1)}) \uparrow$ with dimension

Example 2

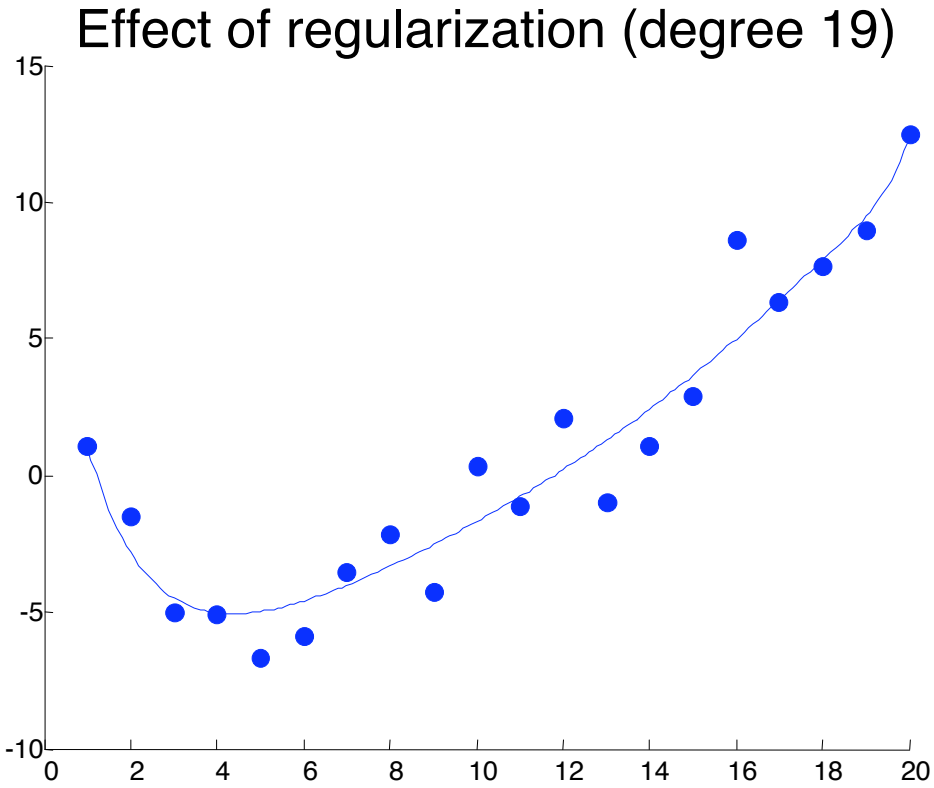
If the linear model is correct, or almost correct, K-nearest neighbors will do much worse than linear regression.

In cases like this (and of course, assuming we know this is the case), simple linear regression methods are not affected by the dimension.



The curves show the expected prediction error (at $x_0 = 0$) for 1-nearest neighbor relative to least squares for the model $Y = f(X) + \varepsilon$. For the red curve, $f(x) = x_1$, while for the green curve $f(x) = \frac{1}{2}(x_1 + 1)^3$.

Ridge Regression (Regularization)



$$A = X^\top X$$

$$b = X^\top y$$

with ϵ "small"

Minimize $\frac{1}{2}\|Xw - y\|_2^2 + \epsilon\|w\|_2^2$ by solving $(A + \epsilon I)w = b$

Probabilistic interpretation

Likelihood $$y_i|x_i \sim N(X_i^\top w, \sigma^2)$$

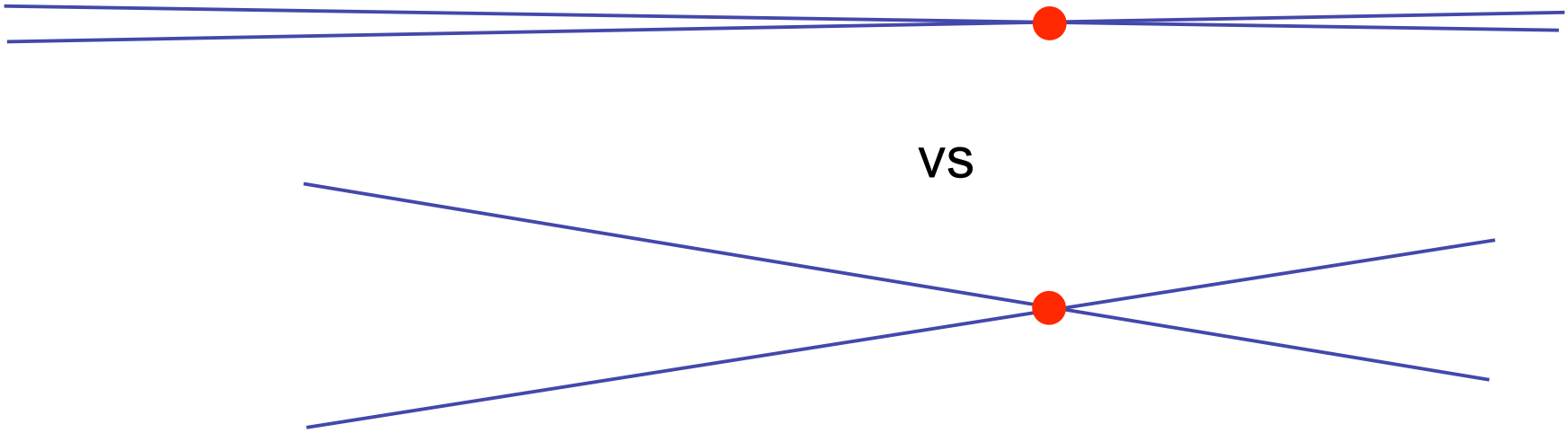
Prior $$w \sim N\left(0, \frac{\sigma^2}{\epsilon}\right)$$

Posterior $$\begin{aligned} P(w|x_1, \ldots x_n) &= \frac{P(w, x_1, \ldots x_n)}{P(x_1, \ldots x_n)} \\ &\propto P(w, x_1, \ldots x_n) \end{aligned}$$

$$\begin{aligned} P(w, x_1, \ldots x_n) &= \exp -\left\{\frac{\epsilon}{2\sigma^2}\|w\|_2^2\right\} \prod_i \exp -\frac{1}{2\sigma^2}(X_i^\top w - y_i)^2 \\ &= \exp -\frac{1}{2\sigma^2}\left[\epsilon\|w\|_2^2 + \sum_i (X_i^\top w - y_i)^2\right] \end{aligned}$$

Numerical Accuracy

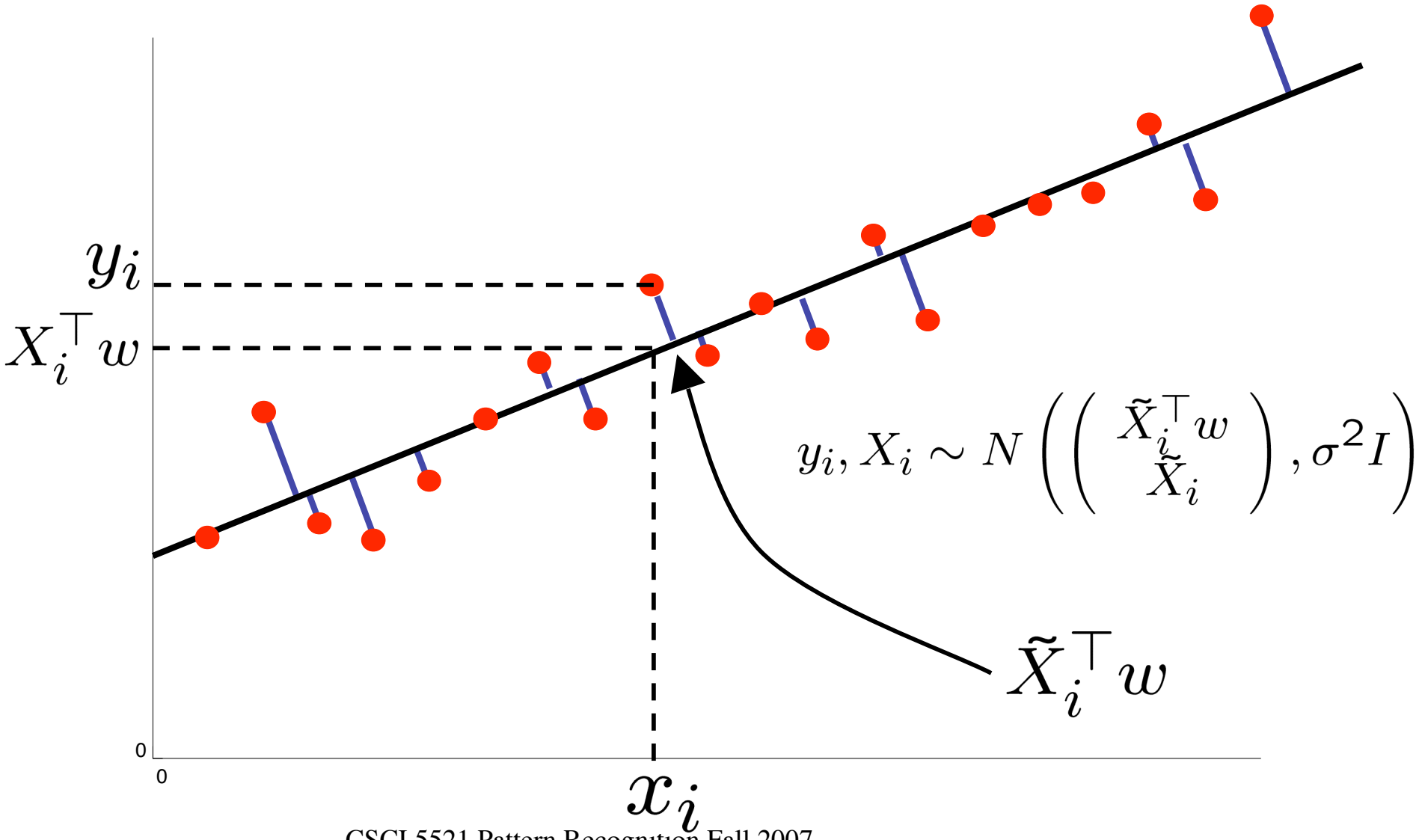
Condition number $\kappa(A) = \|A\|\|A^{-1}\|$



We want covariates as perpendicular as possible, and roughly the same scale

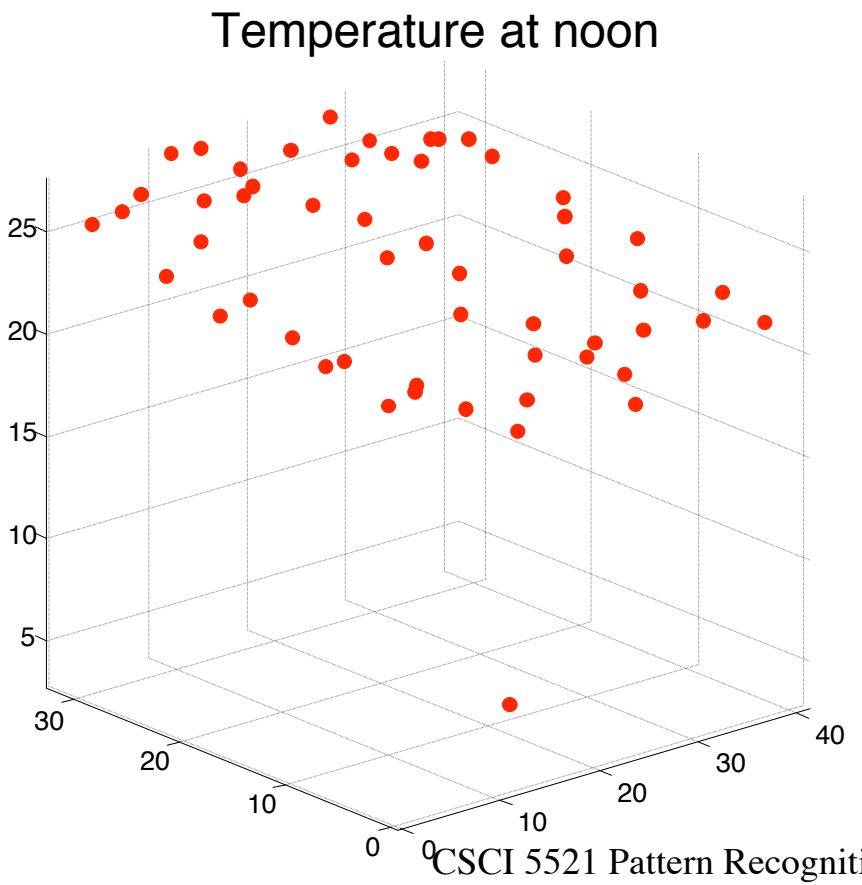
- Regularization
- Preconditioning

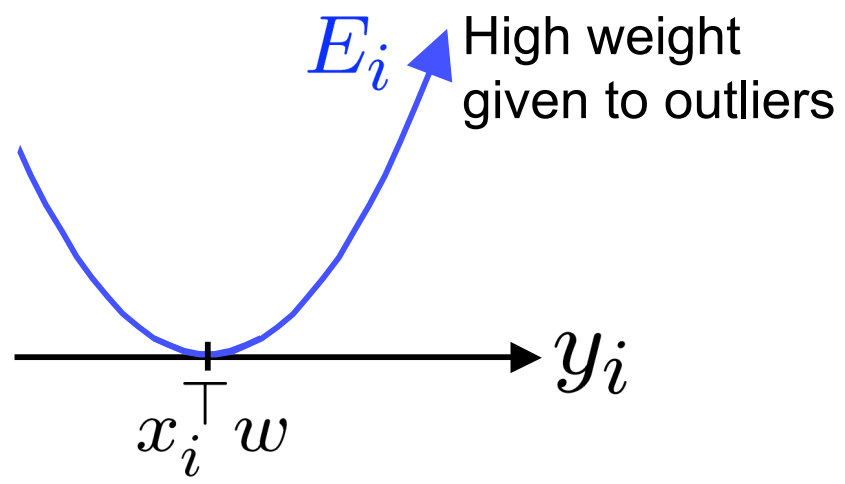
Errors in Variables (Total Least Squares)

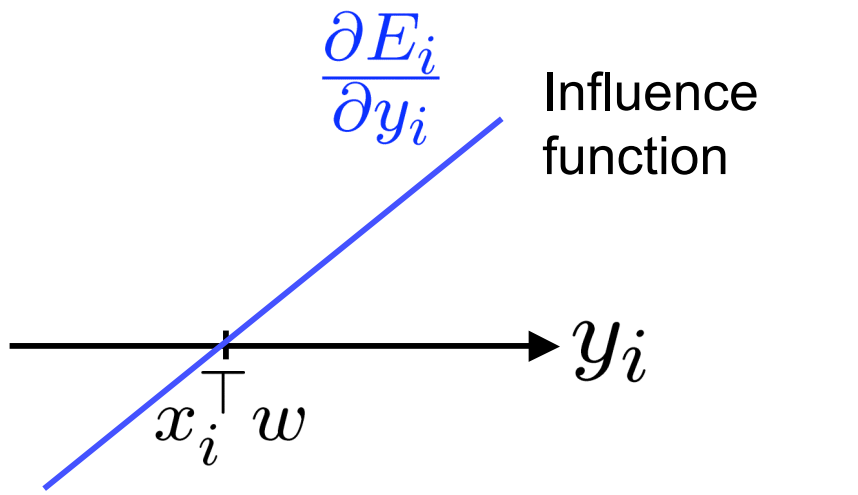


Sensitivity to outliers

$$E = \sum_i (x_i^\top w - y_i)^2 = \sum_i E_i$$

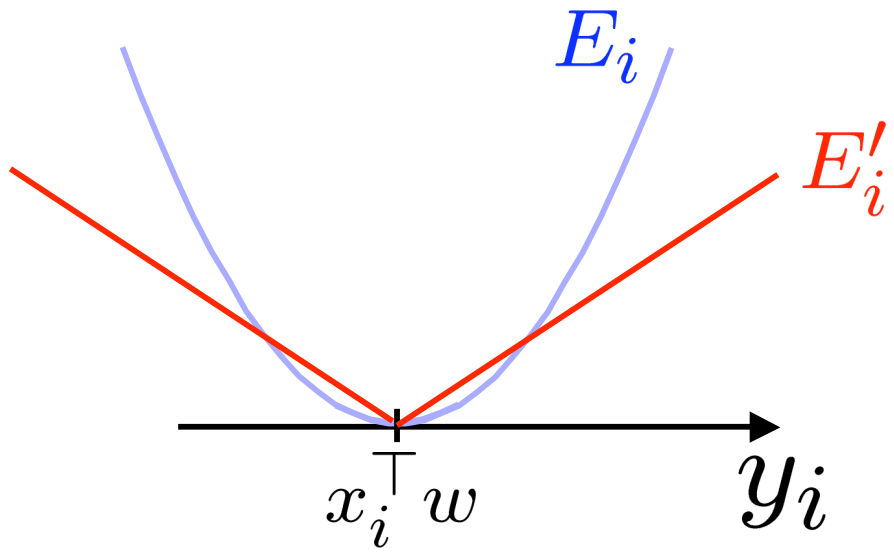






L_1 Regression

$$E' = \sum_i |x_i^\top w - y_i|$$
$$= \sum_i E'_i$$

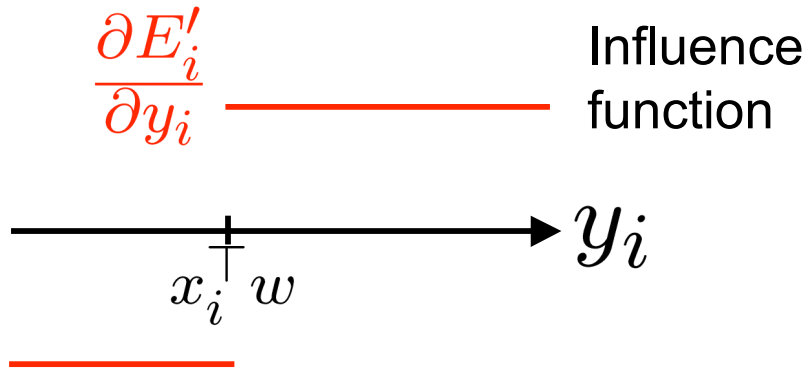


Linear program

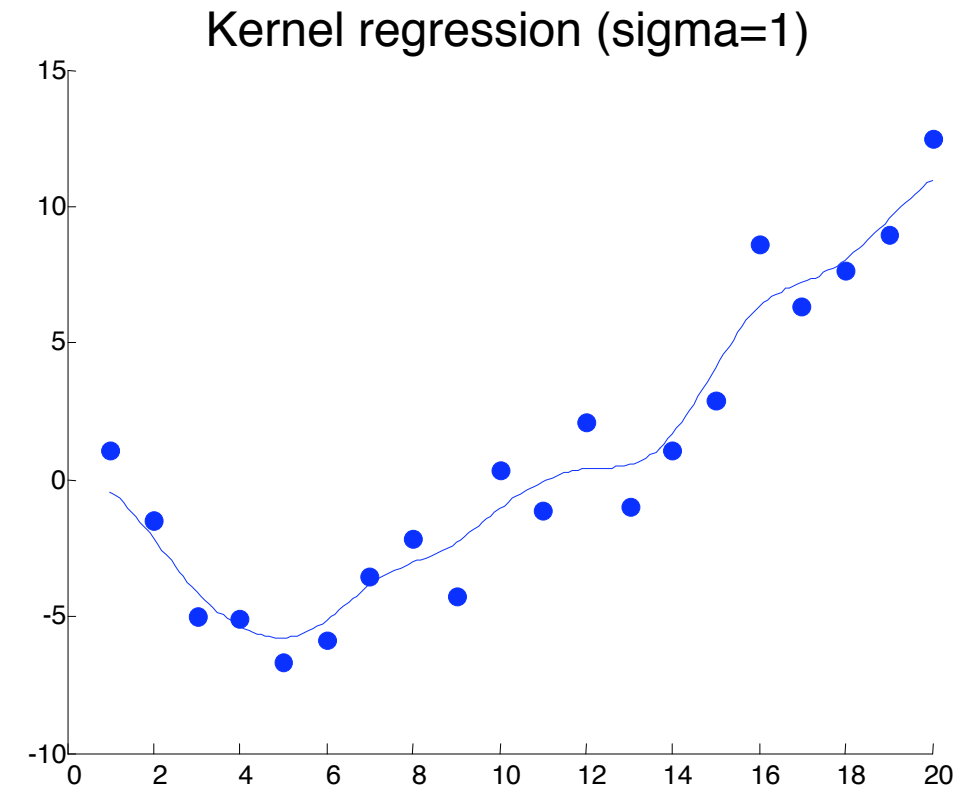
$$\min_{w,c} \sum_i c_i$$

s.t. $x_i^\top w - y_i \leq c_i \quad \forall i$

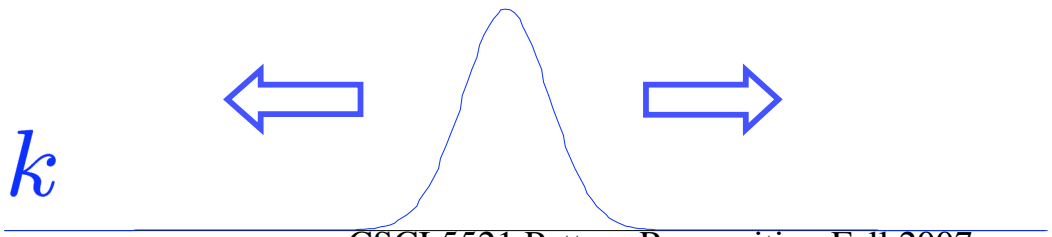
$y_i - x_i^\top w \leq c_i \quad \forall i$



Kernel Regression

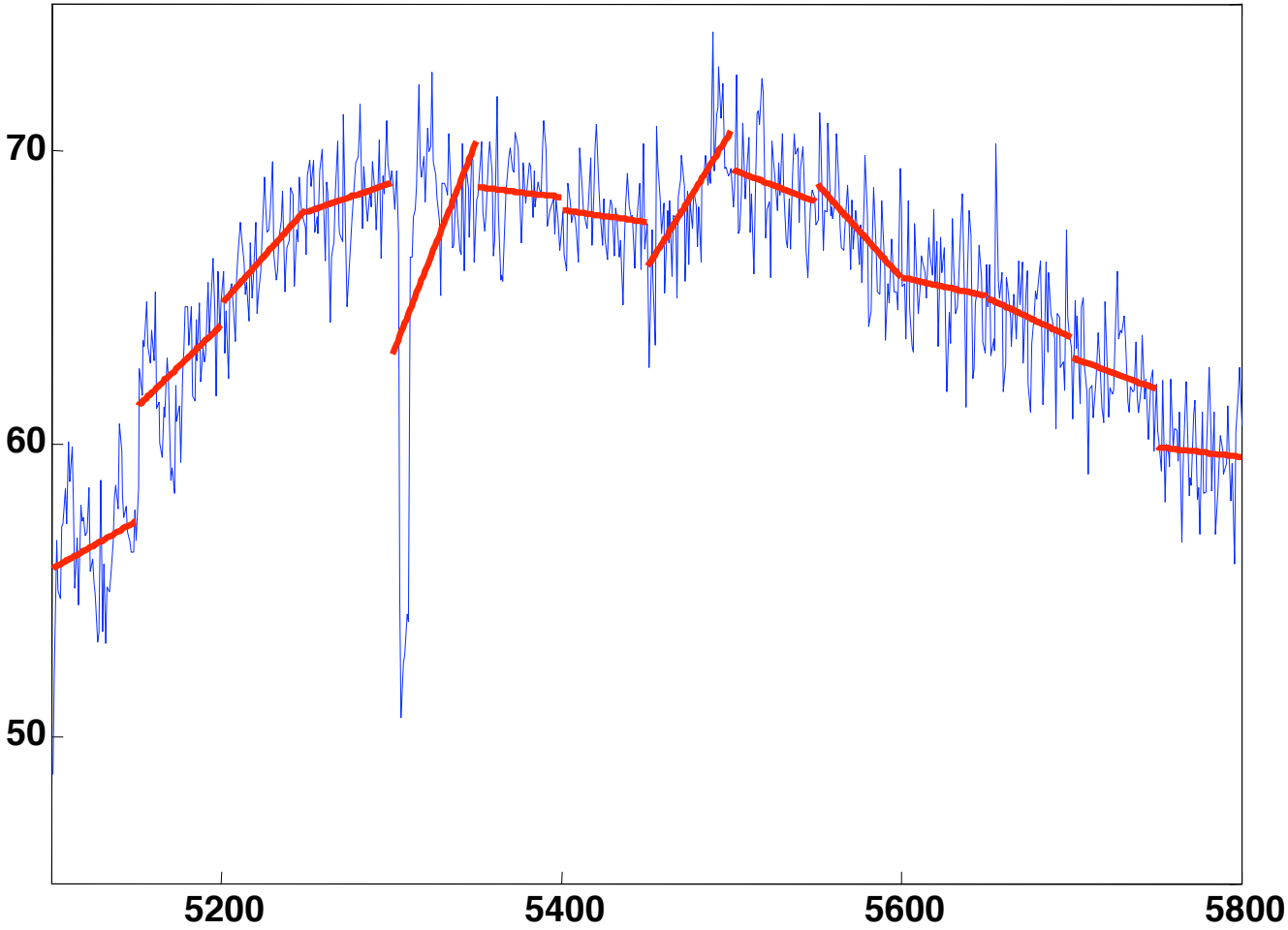


$$\widehat{y}(x) = \frac{\sum_i y_i k(x_i - x)}{\sum_i k(x_i - x)}$$



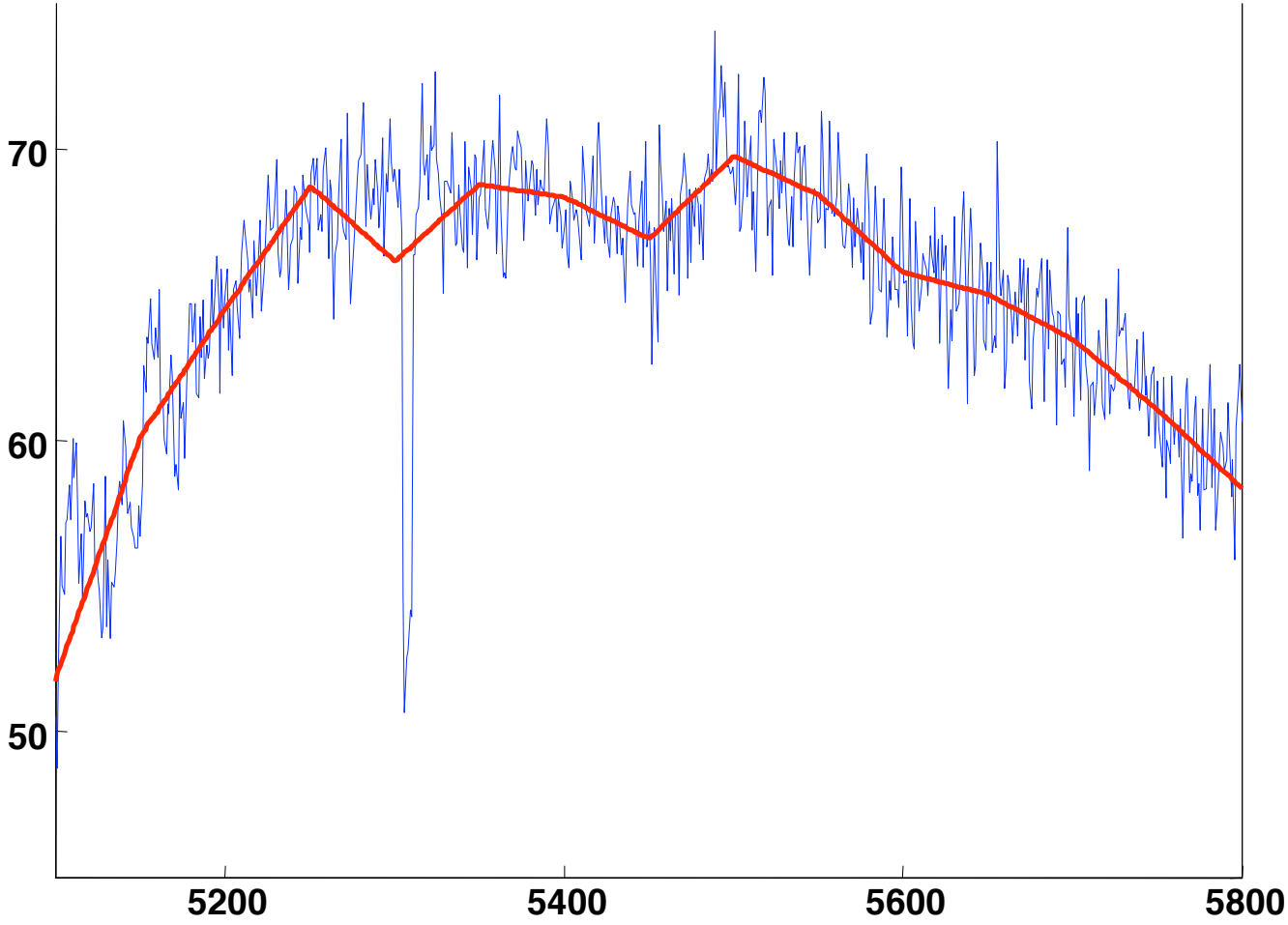
Spline Regression

Regression on each interval



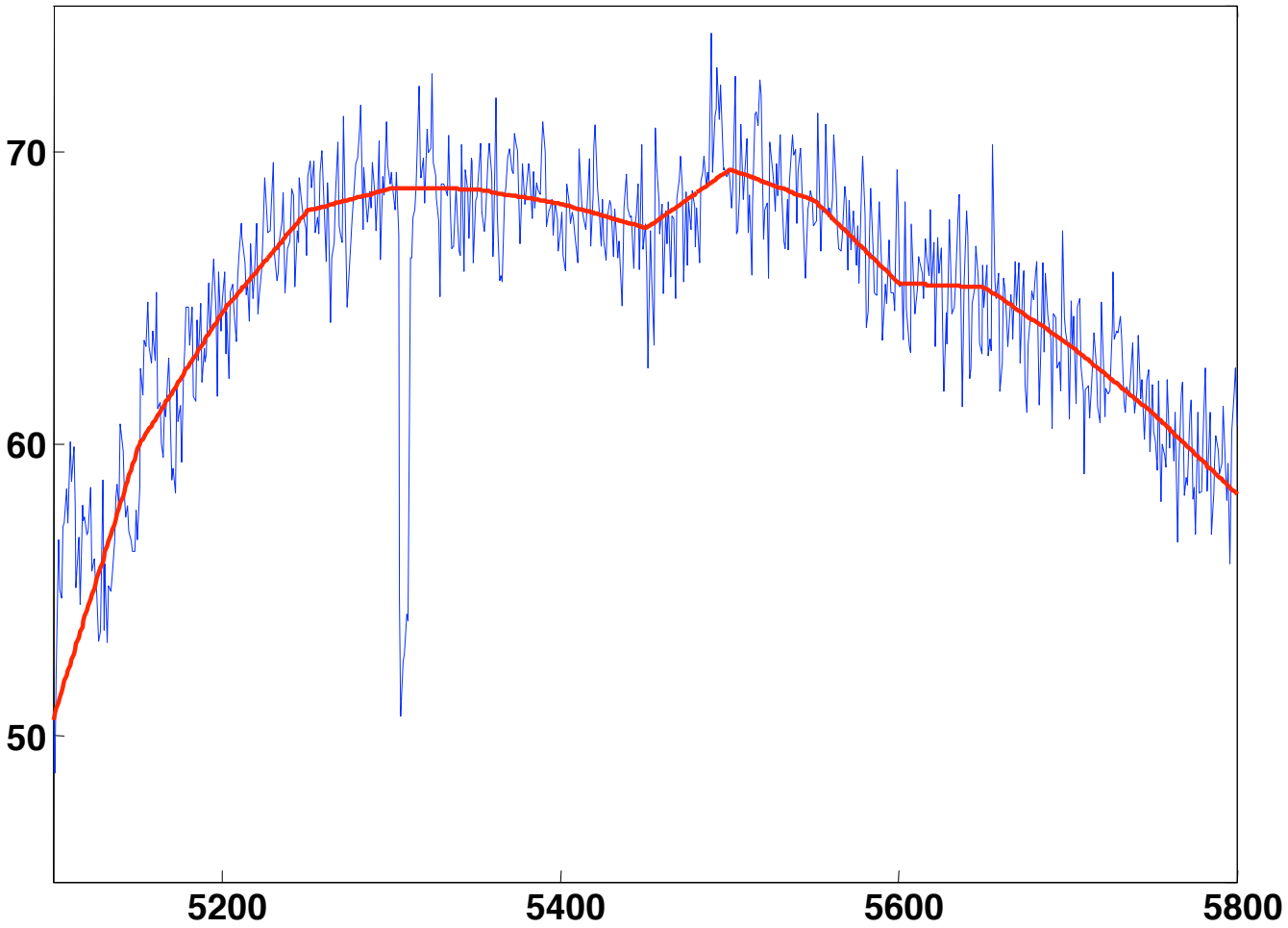
Spline Regression

With equality constraints

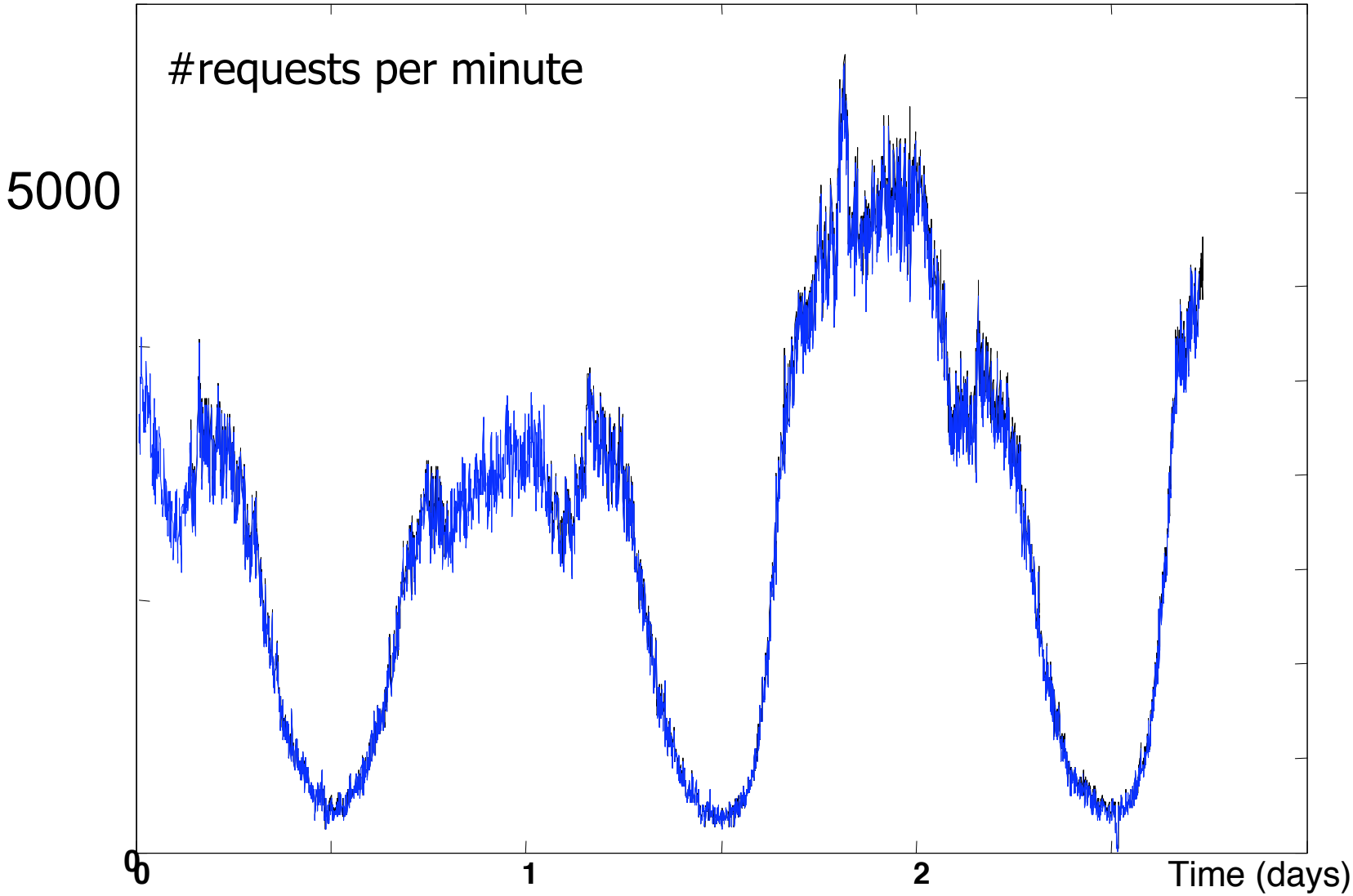


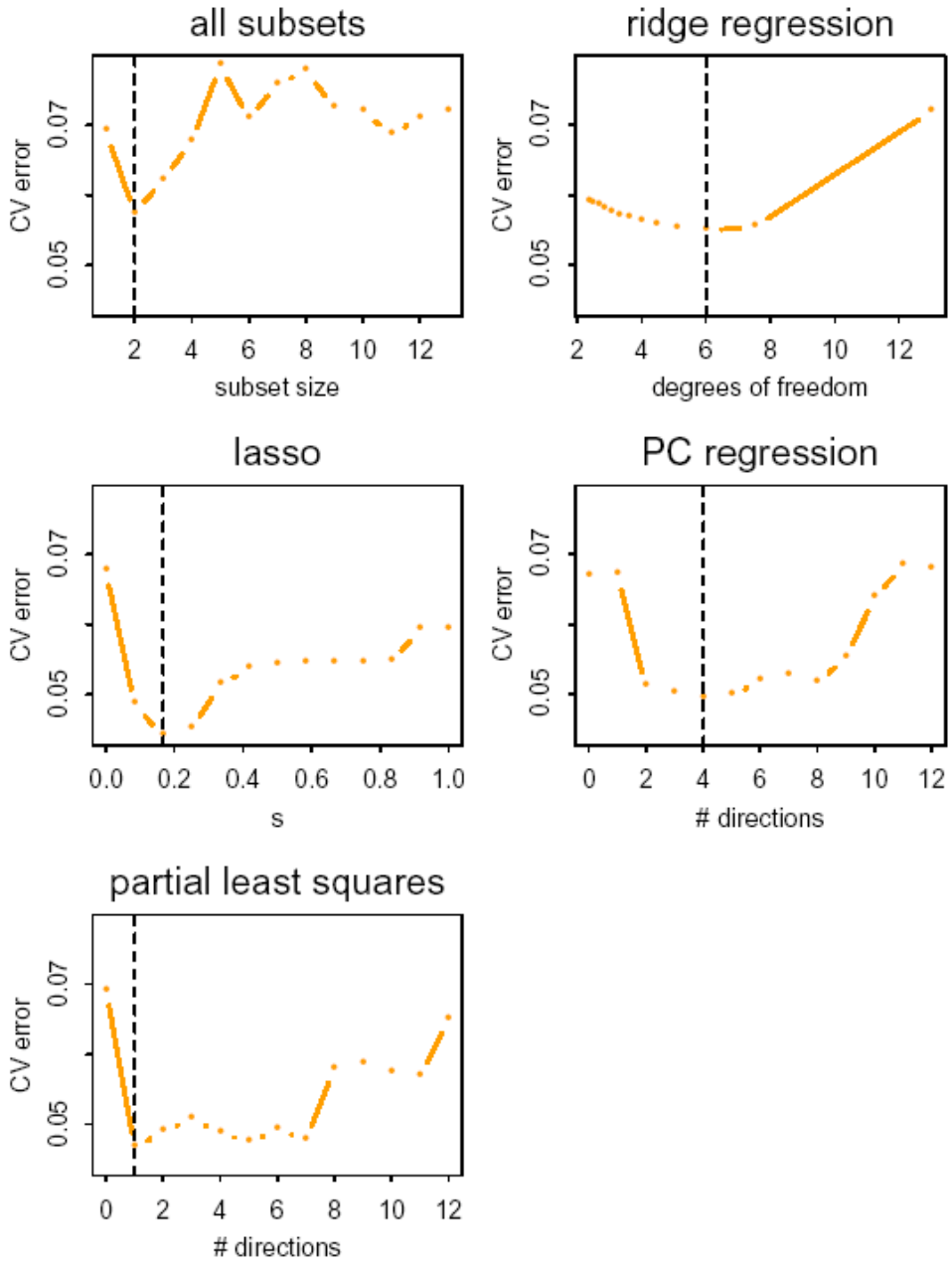
Spline Regression

With L_1 cost



Heteroscedasticity





Estimated prediction error curves for the various selection and shrinkage methods. The arrow indicates the estimated minimizing value of the complexity parameter. Training sample size = 50.

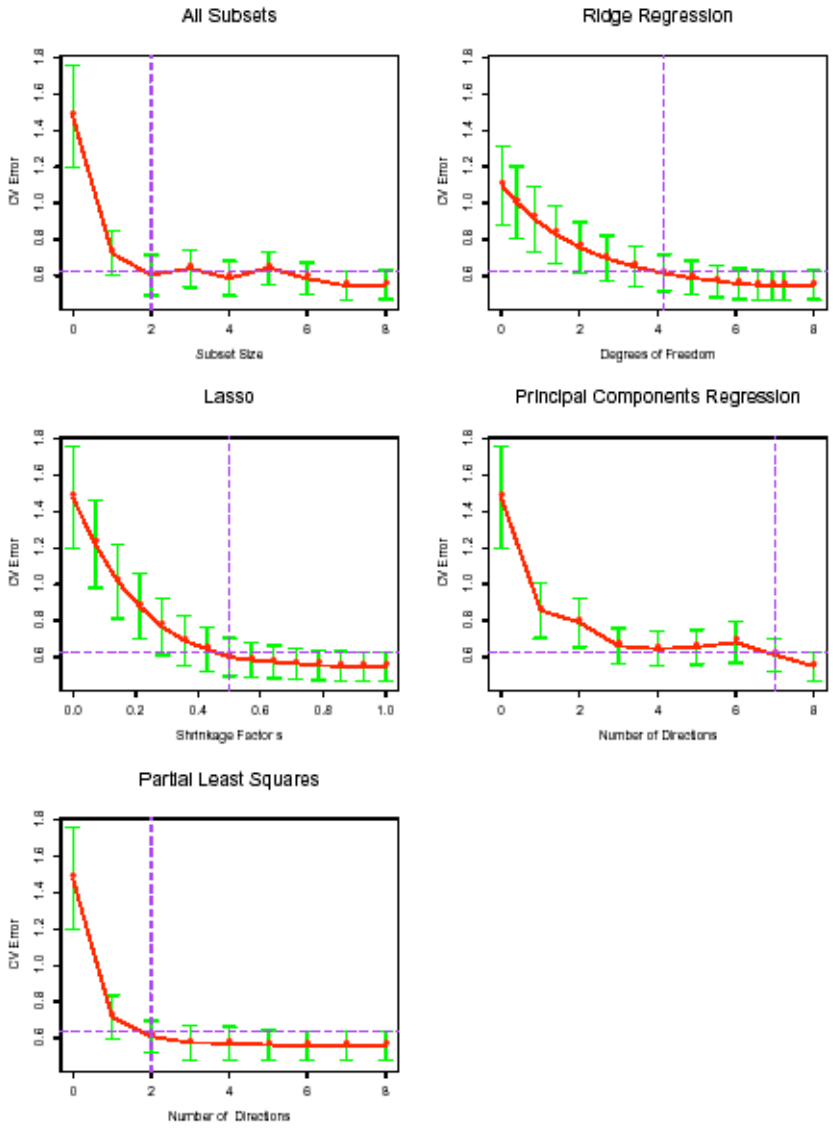


Figure 3.6: *Estimated prediction error curves and their standard errors for the various selection and shrinkage methods, found by 10-fold cross-validation.*

Shrinkage methods

Ridge regression

The ridge estimator is defined by

$$\hat{\beta}^{\text{ridge}} = \text{argmin}(\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta) + \lambda\beta^T\beta$$

Equivalently,

$$\begin{aligned}\hat{\beta}^{\text{ridge}} \quad = \quad & \text{argmin } (\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta) \\ & \text{subject to } \sum \beta_j^2 \leq s.\end{aligned}$$

The parameter $\lambda > 0$ penalizes β_j proportional to its size β_j^2. Solution is

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

where I is the identity matrix. This is a biased estimator that for some value of $\lambda > 0$ may have smaller mean squared error than the least squares estimator.

Note $\lambda = 0$ gives the least squares estimator; if $\lambda \to \infty$, then $\hat{\beta} \to 0$.

The Lasso

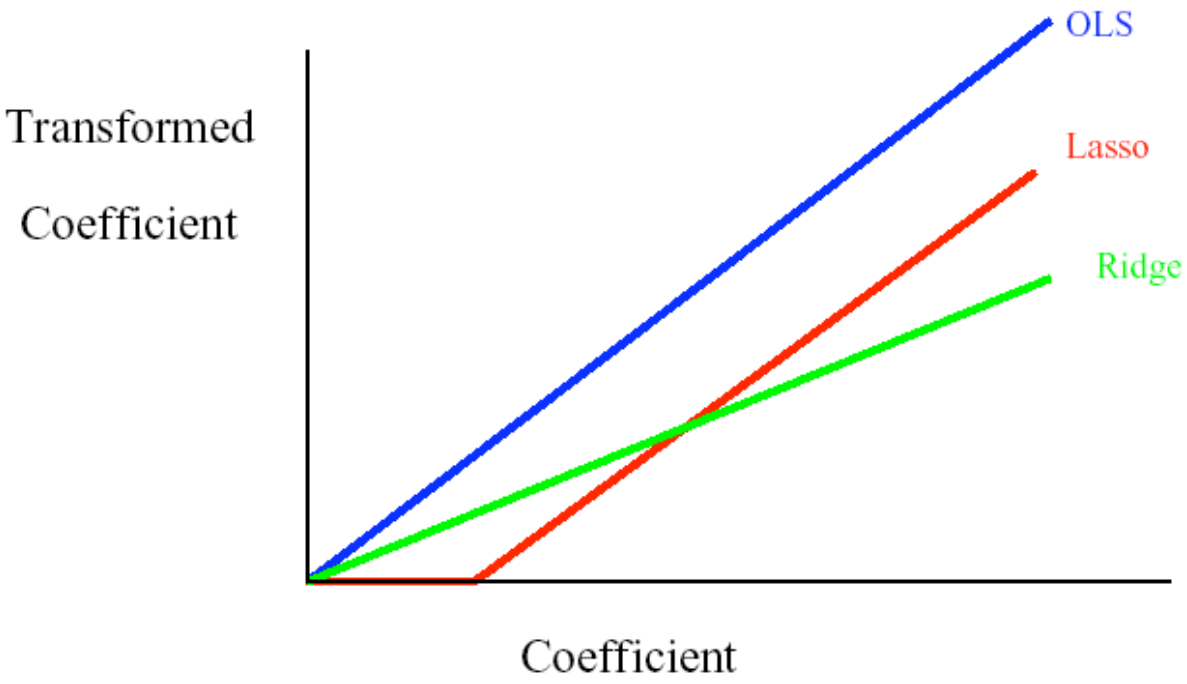
The lasso is a shrinkage method like ridge, but acts in a nonlinear manner on the outcome y.

The lasso is defined by

$$\begin{aligned} \hat{\beta}^{\text{lasso}} &= \text{argmin } (\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta) \\ &\quad \text{subject to } \sum |\beta_j| \leq t \end{aligned}$$

- Notice that ridge penalty $\sum \beta_j^2$ is replaced by $\sum |\beta_j|$.
- this makes the solutions nonlinear in $\mathbf{y}$, and a quadratic programming algorithm is used to compute them.
- because of the nature of the constraint, if t is chosen small enough then the lasso will set some coefficients exactly to zero. Thus the lasso does a kind of continuous model selection.

- The parameter t should be adaptively chosen to minimize an estimate of expected, using say cross-validation
- *Ridge vs Lasso:* if inputs are orthogonal, ridge *multiplies* least squares coefficients by a constant < 1, lasso *translates* them towards zero by a constant, truncating at zero.



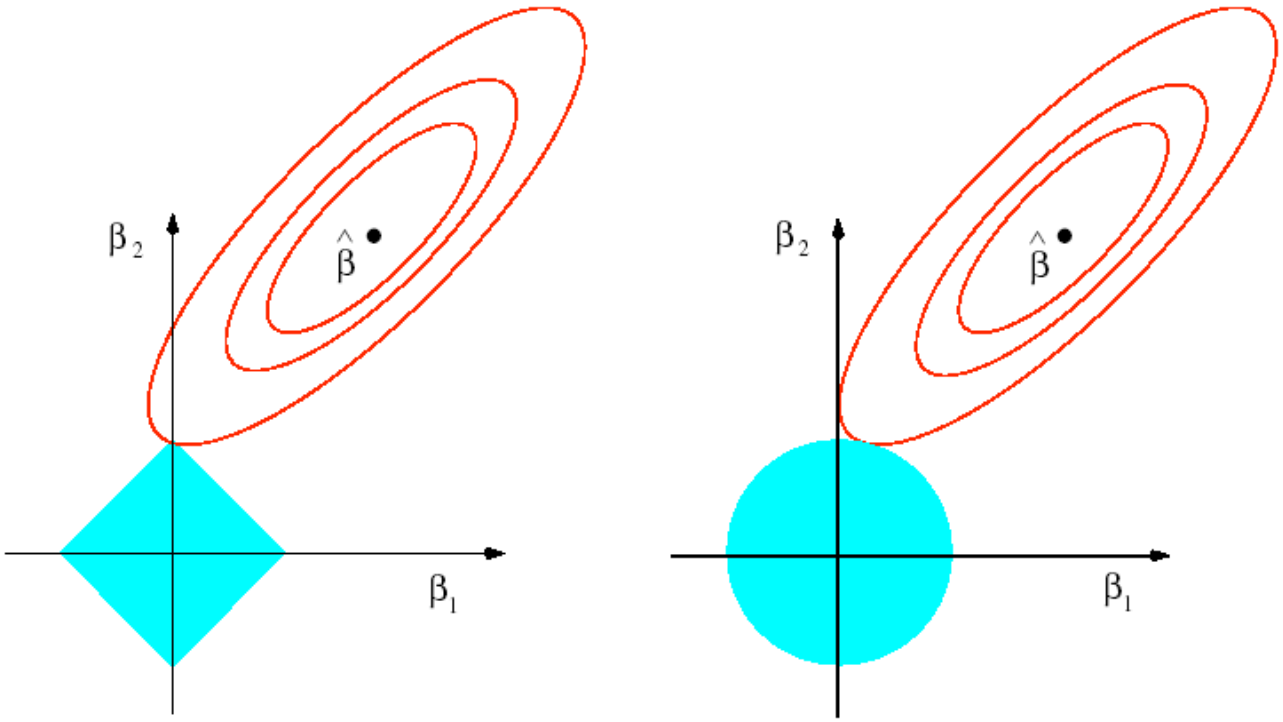


Figure 3.12: *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$, *respectively, while the red ellipses are the contours of the least squares error function.*

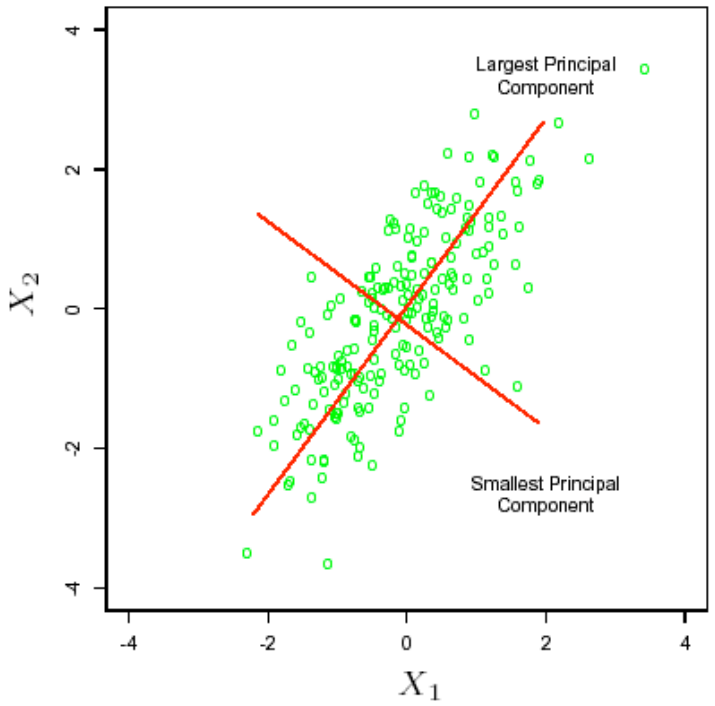


Figure 3.8: *Principal components of some input data points. The largest principal component is the direction that maximizes the variance of the projected data, and the smallest principal component minimizes that variance. Ridge regression projects* $\mathbf{y}$ *onto these components, and then shrinks the coefficients of the low-variance components more than the high-variance components.*

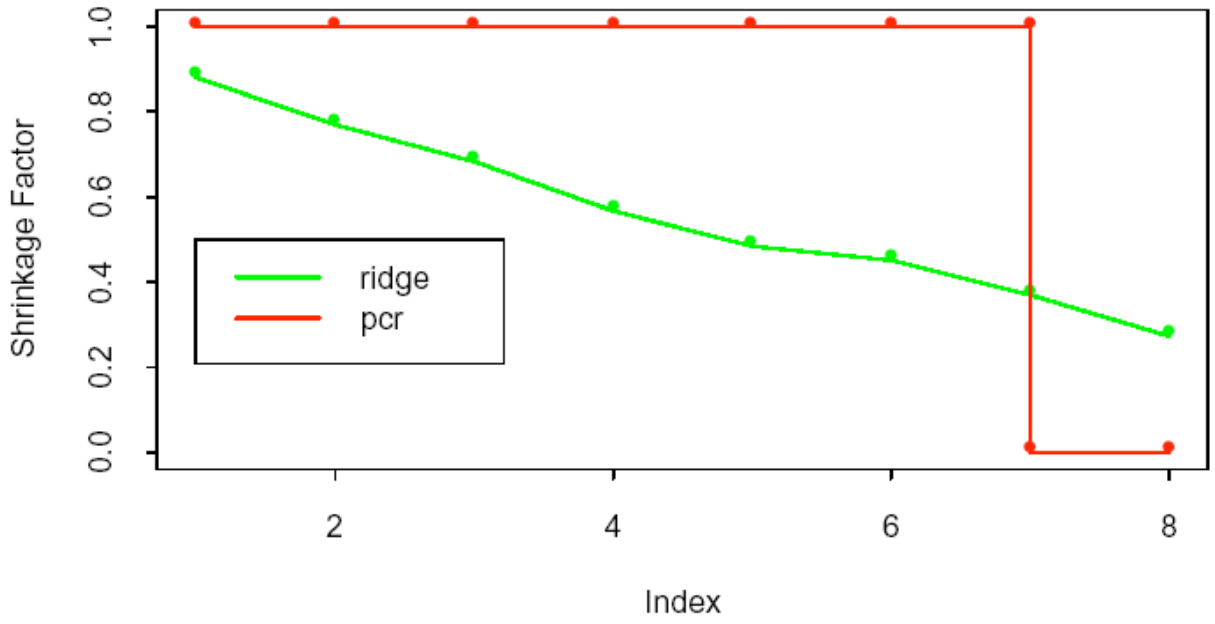
PCA regression continued

- Write $\mathbf{q}_{(j)}$ for the ordered principal components, ordered from largest to smallest value of d_j^2.
- Then principal components regression computes the derived input columns $\mathbf{z}_j = X\mathbf{q}_{(j)}$ and then regresses $\mathbf{y}$ on $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_J$ for some $J \leq p$.
- Since the $\mathbf{z}_j$s are orthogonal, this regression is just a sum of univariate regressions:

$$\hat{\mathbf{y}}^{\mathrm{pcr}} = \bar{y} + \sum_{j=1}^{J} \hat{\gamma}_j \mathbf{z}_j$$

where $\hat{\gamma}_j$ is the univariate regression coefficient of $\mathbf{y}$ on $\mathbf{z}_j$.

- Principal components regression is very similar to ridge regression: both operate on the principal components of the input matrix.
- Ridge regression shrinks the coefficients of the principal components, with relatively more shrinkage applied to the smaller components than the larger; principal components regression discards the $p - J + 1$ smallest eigenvalue components.



Partial least squares

This technique also constructs a set of linear combinations of the x_js for regression, but unlike principal components regression, it uses $\mathbf{y}$ (in addition to X) for this construction.

- We assume that $\mathbf{y}$ is centered and begin by computing the univariate regression coefficient $\hat{\gamma}_j$ of $\mathbf{y}$ on each $\mathbf{x}_j$
- From this we construct the derived input $\mathbf{z}_1 = \sum \hat{\gamma}_j \mathbf{x}_j$, which is the first partial least squares direction.
- The outcome $\mathbf{y}$ is regressed on $\mathbf{z}_1$, giving coefficient $\hat{\beta}_1$, and then we orthogonalize $\mathbf{y}, \mathbf{x}_1, \ldots \mathbf{x}_p$ with respect to $\mathbf{z}_1$: $\mathbf{r}_1 = \mathbf{y} - \hat{\beta}_1 \mathbf{z}_1$, and $\mathbf{x}^*_\ell = \mathbf{x}_\ell - \hat{\theta}_\ell \mathbf{z}_1$
- We continue this process, until J directions have been obtained.

- In this manner, partial least squares produces a sequence of derived inputs or directions $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_J$.
- As with principal components regression, if we continue on to construct $J = p$ new directions we get back the ordinary least squares estimates; use of $J < p$ directions produces a reduced regression
- Notice that in the construction of each $\mathbf{z}_j$, the inputs are weighted by the strength of their univariate effect on $\mathbf{y}$.
- It can also be shown that the sequence $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_p$ represents the conjugate gradient sequence for computing the ordinary least squares solutions.

Ridge vs PCR vs PLS vs Lasso

Recent study has shown that ridge and PCR outperform PLS in prediction, and they are simpler to understand.

Lasso outperforms ridge when there are a moderate number of sizable effects, rather than many small effects. It also produces more interpretable models.

These are still topics for ongoing research.

Use of derived input directions

Principal components regression

We choose a set of linear combinations of the x_js, and then regress the outcome on these linear combinations.

The particular combinations used are the sequence of principal components of the inputs. These are uncorrelated and ordered by decreasing variance.

If S is the sample covariance matrix of $x_1, \ldots, x_p$, then the eigenvector equations

$$S\mathbf{q}_\ell = d_j^2 \mathbf{q}_\ell$$

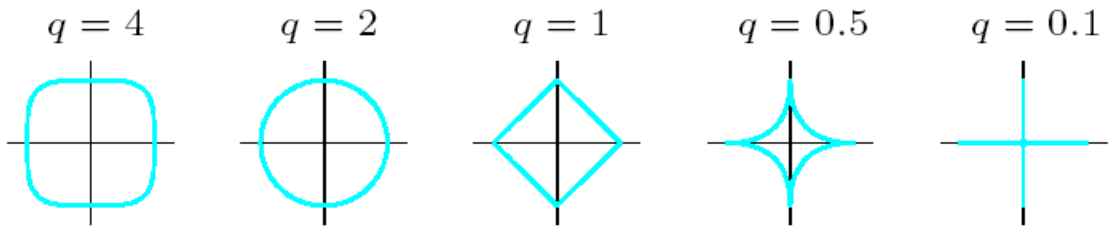
define the principal components of S.

A family of shrinkage estimators

Consider the criterion

$$\tilde{\beta} = \text{argmin}_{\beta} \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$
$$\text{subject to} \sum |\beta_j|^q \leq s$$

for $q \geq 0$. The contours of constant value of $\sum_j |\beta_j|^q$ are shown for the case of two inputs.



Contours of constant value of $\sum_j |\beta_j|^q$ for given values of q.

Thinking of $|\beta_j|^q$ as the log-prior density for β_j, these are also the equi-contours of the prior.