CPSC 340:
Machine Learning and Data Mining

Non-Linear Regression

Fall 2016

Admin

- Assignment 2 is due now.
  - 1 late day to hand it in on Wednesday, 2 for Friday, 3 for next Monday.
- Assignment 3 will be out by early next week.
  - Due October 19 (so we can release solutions before the midterm).
- We will have tutorials on Tuesday/Wednesday of next week:
  - Focusing on multivariate calculus in matrix notation.
- Tutorial room change: T1D (Monday @5pm) moved to DMP 101.

“In this talk, Eli will provide an overview of how big data is used to create and power Silicon Valley's greatest companies, with specific examples from Udacity.”

Location TBA!

There will be free food and drinks. More info and to get your tickets:
https://goo.gl/QGsnUU

https://www.facebook.com/events/645894388926612/

Last Time: Linear Regression

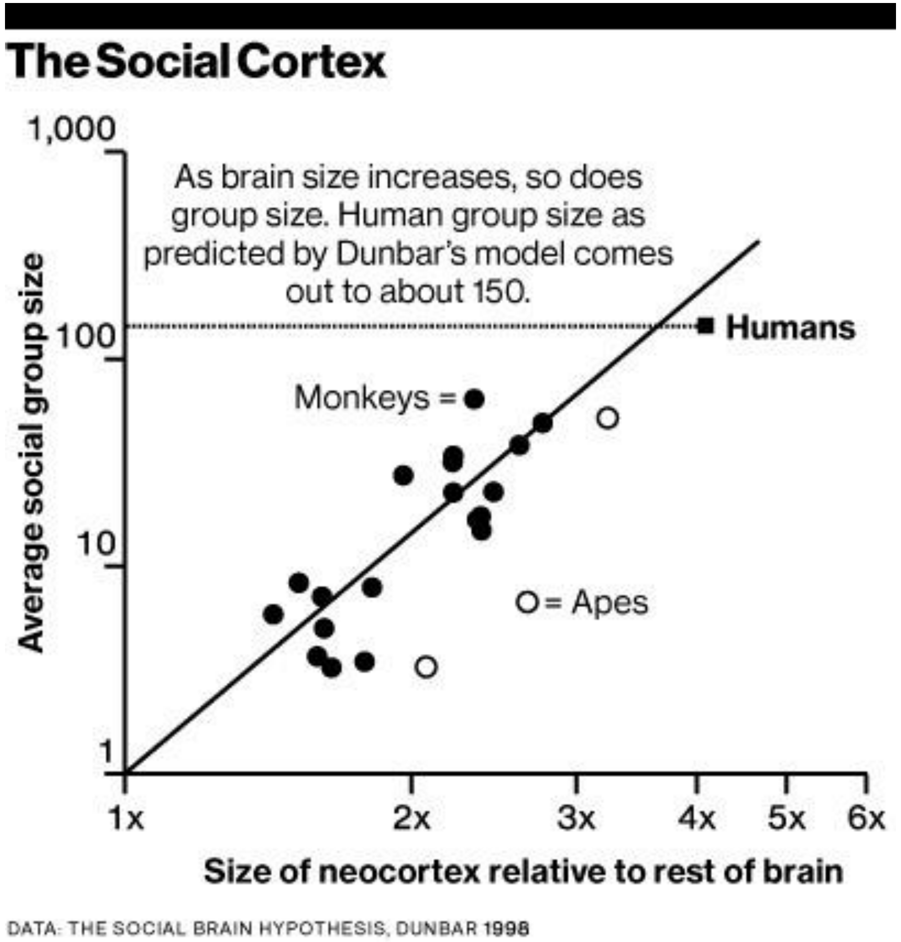
- We discussed linear models:

$$y_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= \sum_{j=1}^{d} w_j x_{ij} = w^T x_i$$

- "Multiply feature x_{ij} by weight w_j, add them to get y_i".
- We discussed squared error function:

$$f(w) = \frac{1}{2} \sum_{i=1}^{n} (\underbrace{w^T x_i}_{\text{Predicted value}} - \underbrace{y_i}_{\text{True value}})^2$$

- Interactive demo:
  - http://setosa.io/ev/ordinary-least-squares-regression

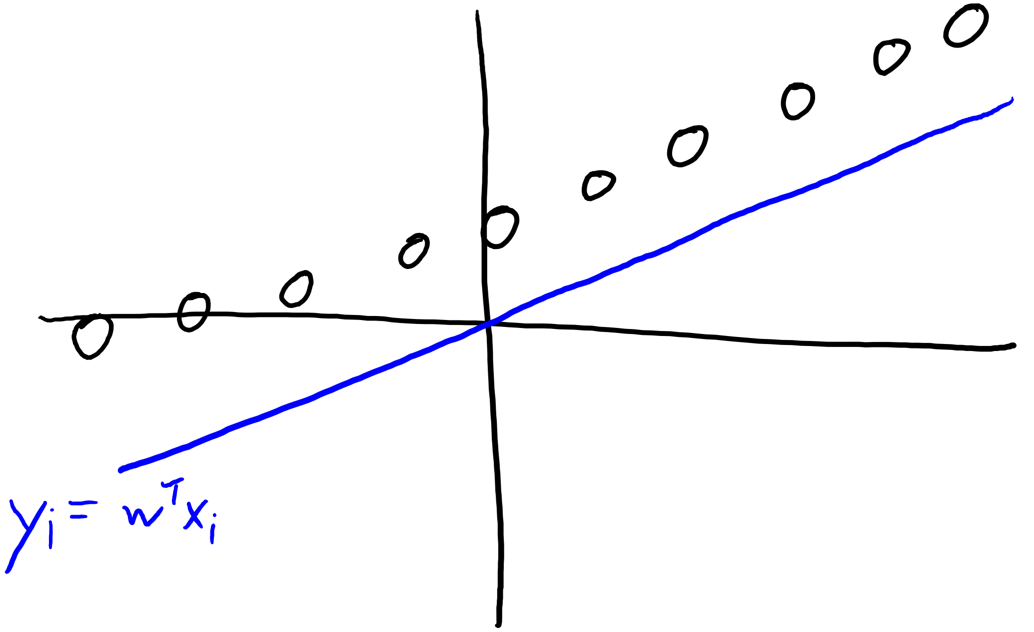


To predict on test case $\hat{x}_i$
use $\hat{y}_i = w^T \hat{x}_i$

http://www.bloomberg.com/news/articles/2013-01-10/the-dunbar-number-from-the-guru-of-social-networks

Why don't we have a y-intercept?

- Last time: Linear models with no y-intercept.
  - Linear model is $y_i = w^T x_i$ instead of $y_i = w^T x_i + w_0$ with y-intercept w_0.
  - So if $x_i = 0$ then we must predict $y_i = 0$.

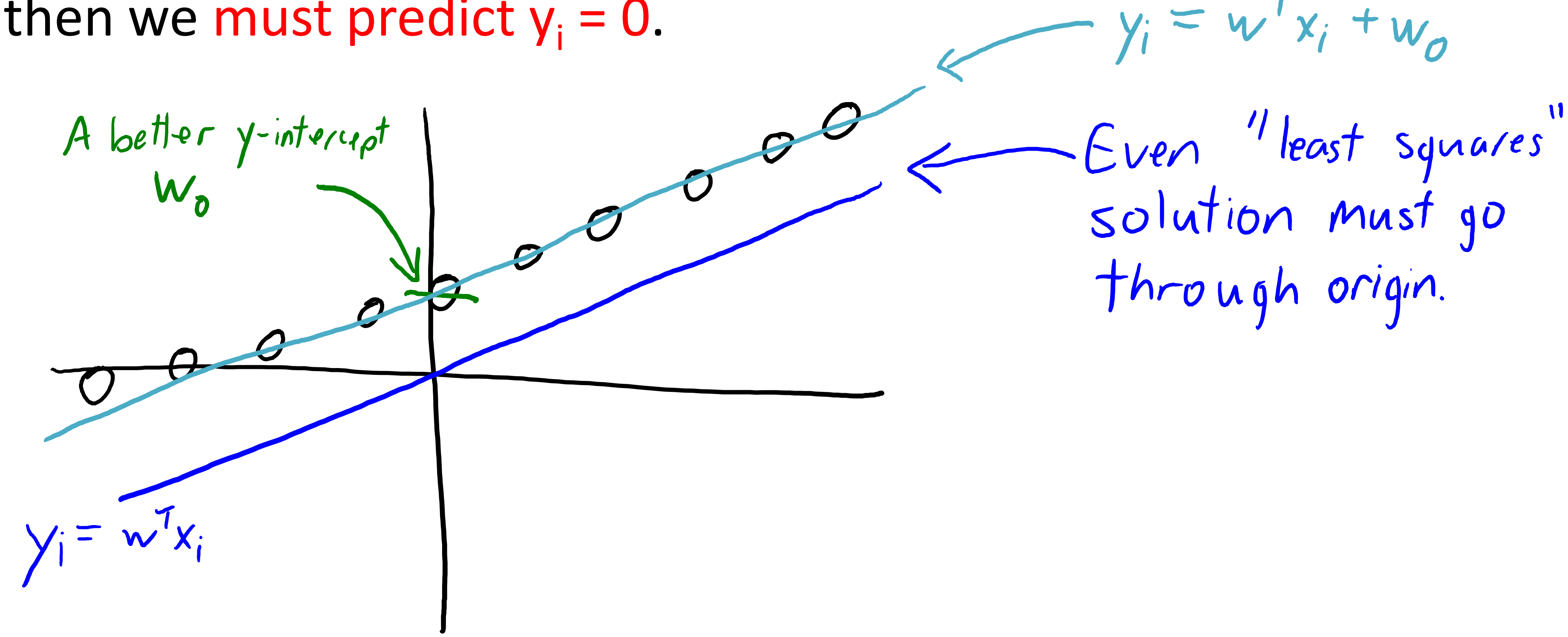


Even "least squares" solution must go through origin.

Why don't we have a y-intercept?

- Last time: Linear models with no y-intercept.
  - Linear model is $y_i = w^T x_i$ instead of $y_i = w^T x_i + w_0$ with y-intercept w_0.
  - So if $x_i = 0$ then we must predict $y_i = 0$.

Adding y-intercept fixes this.



Adding a Bias Variable

- Simple trick to add a y-intercept ("bias") variable:
  - Make a new matrix "Z" with an extra feature that is always "1".

$$X = \begin{bmatrix} 0.1 & 0.3 \\ 0.5 & -0.6 \\ 0.2 & 0.4 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0.1 & 0.3 \\ 1 & 0.5 & -0.6 \\ 1 & 0.2 & 0.4 \end{bmatrix}$$

X

- Now use "Z" as features to get a model with a non-zero y-intercept:

$$y_i = w_0 \underbrace{z_{i0}}_{\hookrightarrow \text{"1"}} + w_1 \underbrace{z_{i1}}_{\hookrightarrow x_{i1}} + w_2 \underbrace{z_{i2}}_{\hookrightarrow x_{i2}}$$

$$= w_0 + w_1 x_{i1} + w_2 x_{i2}$$

- So we can have a non-zero y-intercept by changing features.

Linear Least Squares

Training:

$$w = (X'*X)\backslash(X'*y)$$

Prediction:

$$\underbrace{y = X*w}_{\text{Why?}}$$

$y_i = w^T x_i$ so

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} w^T x_1 \\ w^T x_2 \\ \vdots \\ w^T x_n \end{bmatrix} = \begin{bmatrix} x_1^T w \\ x_2^T w \\ \vdots \\ x_n^T w \end{bmatrix} = \underbrace{\begin{bmatrix} — x_1^T — \\ — x_2^T — \\ \vdots \\ — x_n^T — \end{bmatrix}}_{X} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = Xw$$

Linear Least Squares

Training:

$$\underbrace{w = (X' * X) \backslash (X' * y)}_{\text{Why?}}$$

Want 'w' that minimizes

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(\underbrace{w^Tx_i - y_i}_{r_i})^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2 = \frac{1}{2} r^T r = \frac{1}{2}\|r\|_2^2 = \boxed{\frac{1}{2}\|Xw - y\|^2}$$

Define "residual" r_i as signed error on example 'i':

$$r_i = w^Tx_i - y_i$$

$$r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} w^Tx_1 - y_1 \\ w^Tx_2 - y_2 \\ \vdots \\ w^Tx_n - y_n \end{bmatrix} = \underbrace{\begin{bmatrix} w^Tx_1 \\ w^Tx_2 \\ \vdots \\ w^Tx_n \end{bmatrix}}_{Xw} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = Xw - y$$

Linear Least Squares

$\|r\|_2 = \sqrt{\sum_{i=1}^{d} r_i^2}$

Training:

$$\underbrace{w = (X'*X)\backslash(X'*y)}_{\text{Why?}}$$

$r^T r = \|r\|^2$

Want 'w' that minimizes

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 = \underbrace{\frac{1}{2}\|Xw - y\|_2^2}_{\text{Let's expand then compute gradient.}} = \frac{1}{2}(Xw - y)^T(Xw - y)$$

$$= \frac{1}{2}((Xw)^T - y^T)(Xw - y)$$

$(AB)^T = B^T A^T$

$$= \frac{1}{2}(w^T X^T - y^T)(Xw - y)$$

"distributive"

$$= \frac{1}{2}(w^T X^T (Xw - y) - y^T(Xw - y))$$

$$= \frac{1}{2}(w^T X^T Xw - w^T X^T y - y^T Xw + y^T y)$$

$$= \frac{1}{2} w^T X^T Xw - w^T X^T y + \frac{1}{2} y^T y$$

$w^T v = v^T w$

$v = X^T y$

A good way to check your step: make sure that dimensions all make sense.

Linear Least Squares

Training:

$$\underbrace{w = (X' * X) \backslash (X' * y)}_{\text{Why?}}$$

Want 'w' that minimizes

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 = \frac{1}{2}\|Xw - y\|_2^2 = \underbrace{\frac{1}{2}w^T\overbrace{X^TX}^{A}w - w^TX^Ty + \frac{1}{2}y^Ty}_{\text{What are the gradients of these terms?}}$$

} a quadratic function (in matrix notation)

$ax^2 + bx + c$

$$\nabla f(w) = X^TXw - X^Ty + 0$$

So at a minimizer where $\nabla f(w) = 0$

we have: $\boxed{X^TXw = X^Ty}$

Cheat sheet:

$\nabla_w[c] = 0$

$\nabla_w[w^Tb] = b$

$\nabla_w[\frac{1}{2}w^TAw] = Aw$ for symmetric A.

Like $\frac{d}{dx}[\frac{1}{2}ax^2] = ax$

See note on webpage for derivation

Linear Least Squares

Training:

$$w = \underbrace{(X'*X)}_{A} \setminus \underbrace{(X'*y)}_{b}$$

"solve linear system"

Why?

Want 'w' that minimizes

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 = \frac{1}{2}\|Xw - y\|_2^2$$

We know that minimizer must have

$$\underbrace{X^T X}_{\text{Some matrix}} w = \underbrace{X^T y}_{\text{Some vector}}$$

"normal equations"

This is a linear system $Aw = b$

for some matrix 'A' and vector 'b'

Note that

$f(w)$ is a "convex" function

so solving $\nabla f(w) = 0$ gives minimizer

Least Squares Issues

- Issues with least squares model:
  - Solution might not be unique.
  - It is sensitive to outliers.
  - It always uses all features.
  - Data can might so big we can't store X^TX.
  - It might predict outside range of y_i values.
  - It assumes a linear relationship between x_i and y_i.

X is $n \times d$

so X^T is $d \times n$

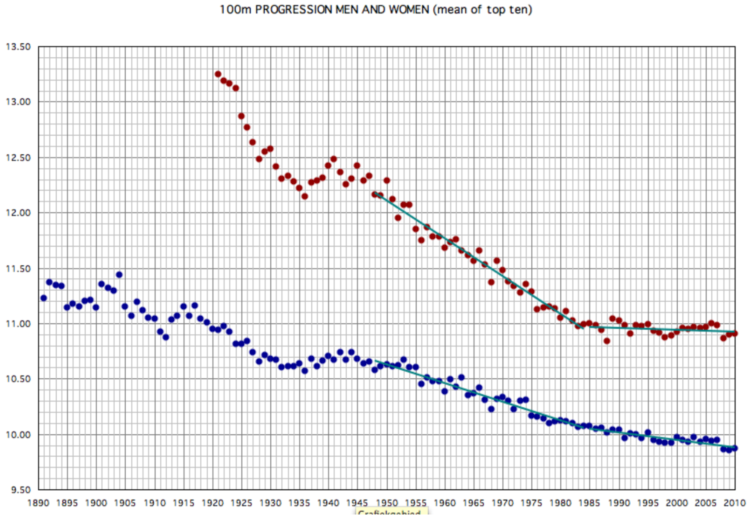
and X^TX is $d \times d$.

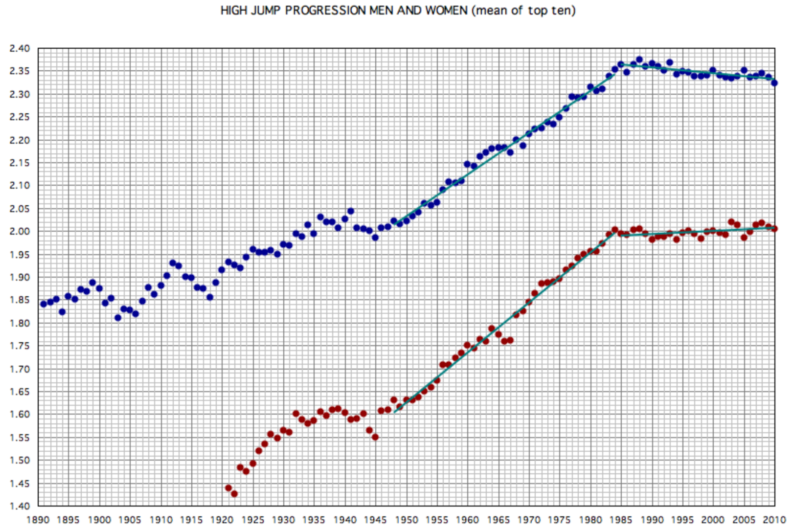
Costs $O(nd^2)$ to calculate:

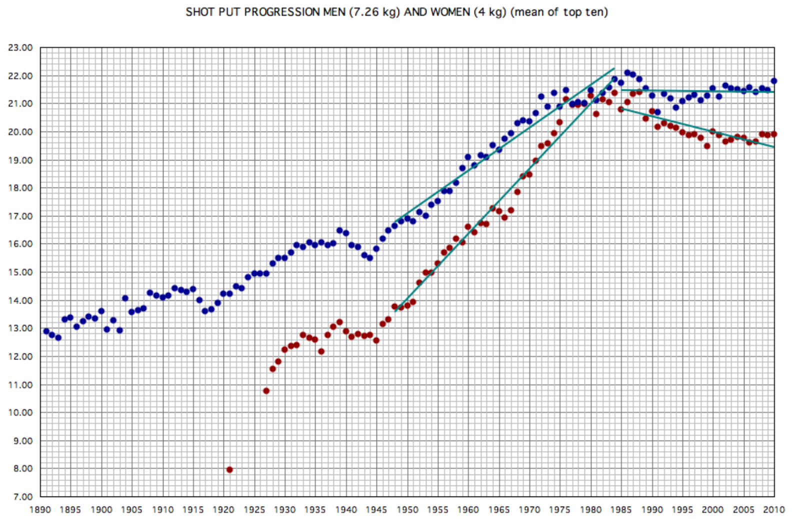
- Each of the $O(d^2)$ elements is an inner product between length 'n' vectors.

Example: Non-Linear Progressions in Athletics

- Are top athletes going faster, higher, and farther?







http://www.at-a-lanta.nl/weia/Progressie.html
https://en.wikipedia.org/wiki/Usain_Bolt
http://www.britannica.com/biography/Florence-Griffith-Joyner

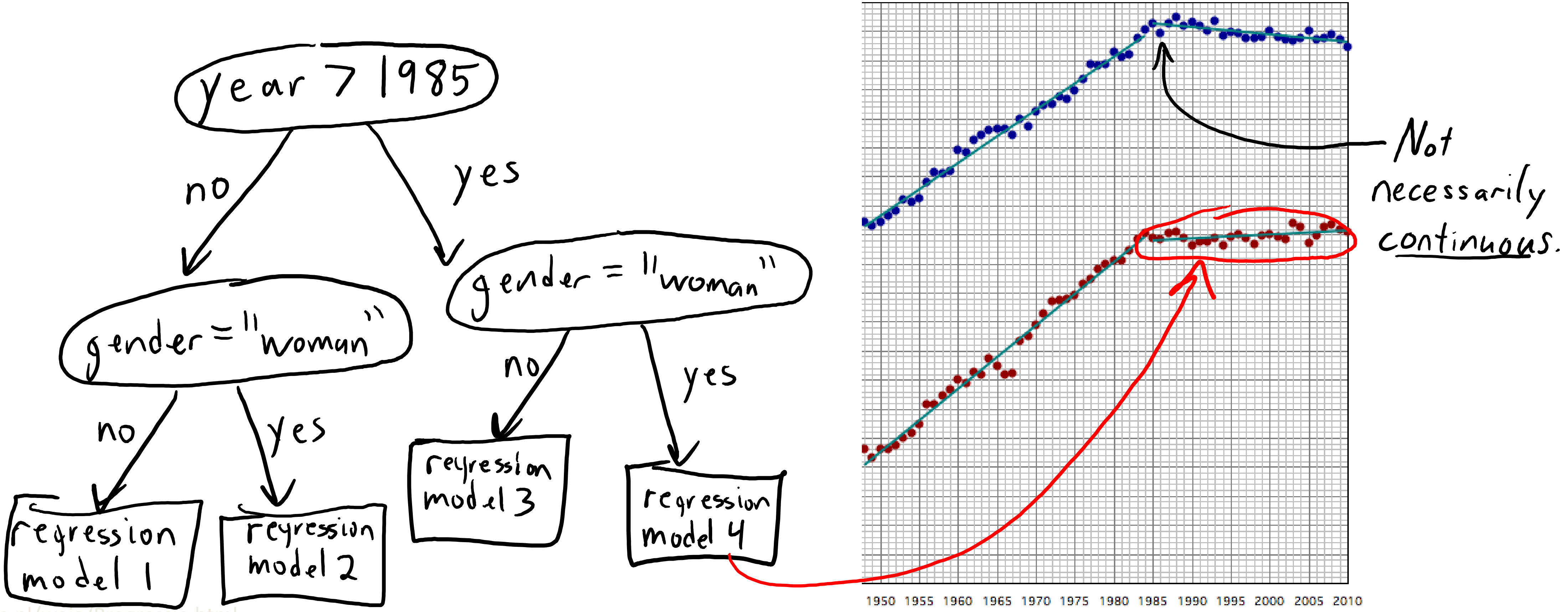
Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:

http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

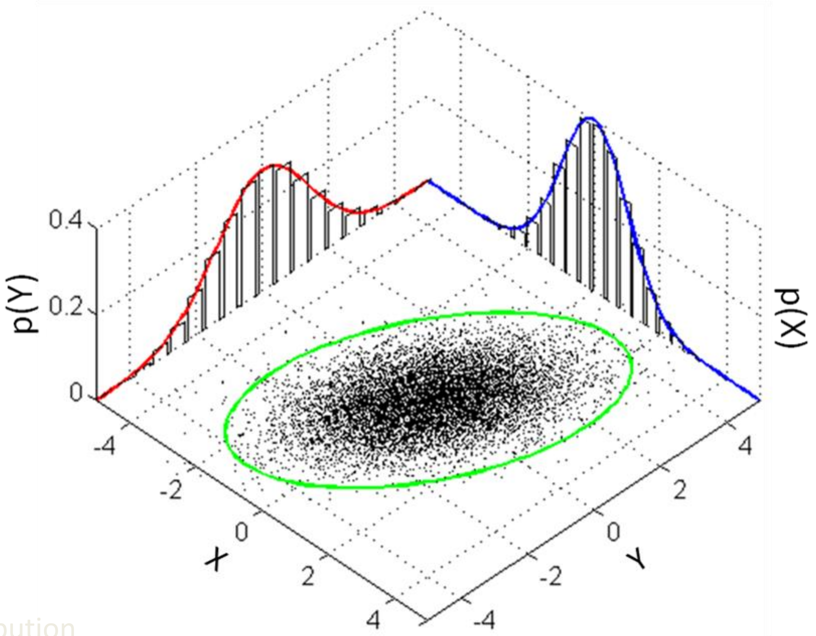
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.



http://www.at-a-lanta.nl/weia/Progressie.html

Adapting Counting/Distance-Based Methods

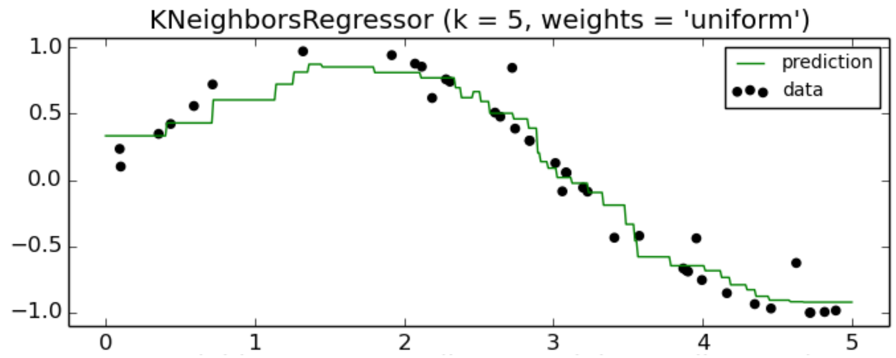
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Generative models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.



https://en.wikipedia.org/wiki/Multivariate_normal_distribution

Adapting Counting/Distance-Based Methods

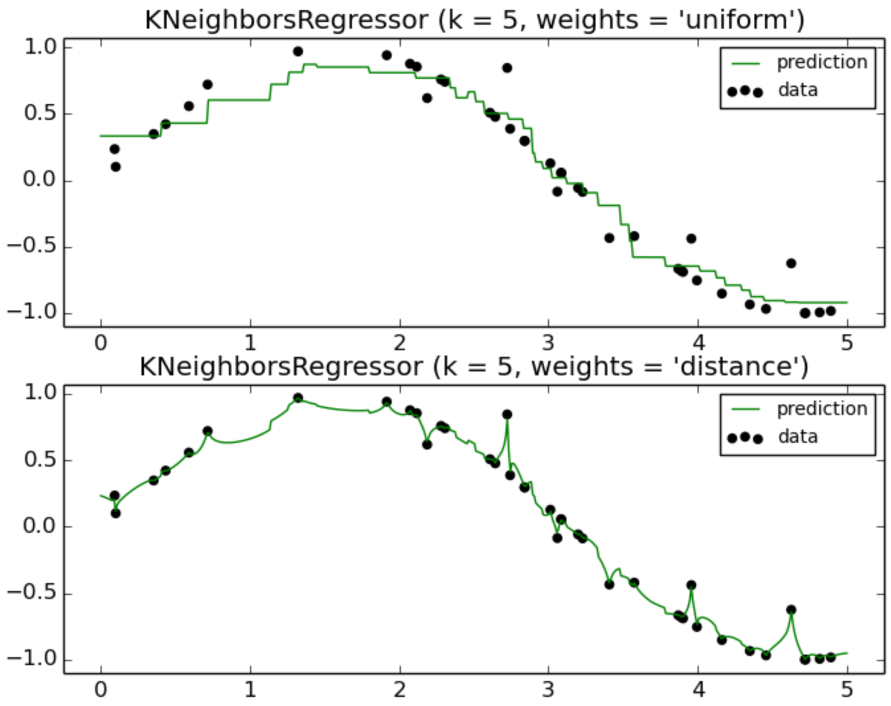
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Generative models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - Mean y_i among k-nearest neighbours.



http://scikit-learn.org/stable/modules/neighbors.html

Adapting Counting/Distance-Based Methods

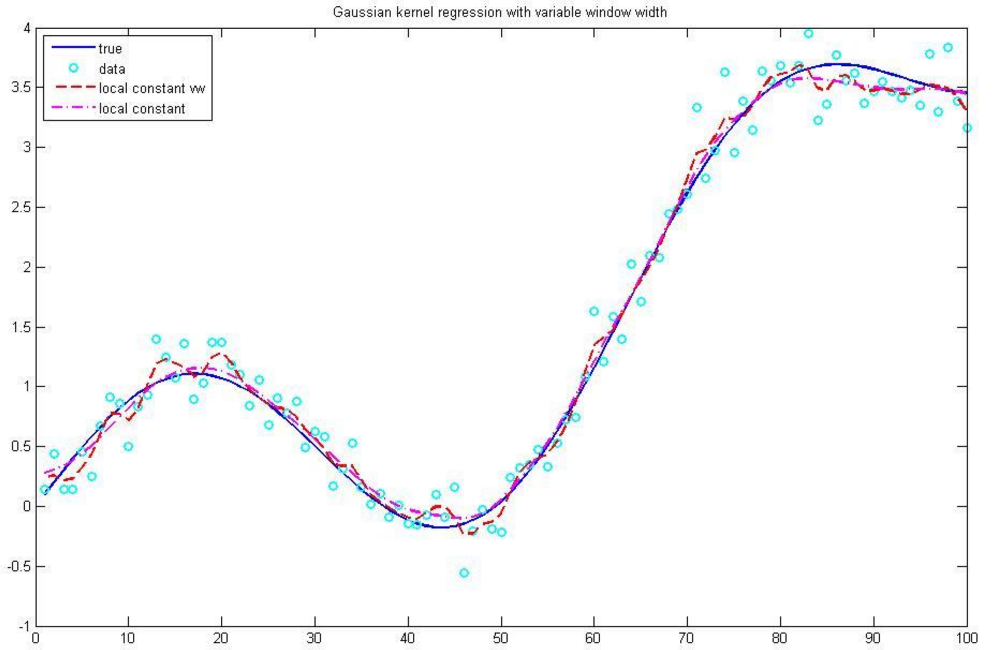
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Generative models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - Mean y_i among k-nearest neighbours.
    - Could be weighted by distance.
      - Close points 'j' get more "weight" w_{ij}.



Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Generative models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - Mean y_i among k-nearest neighbours.
    - Could be weighted by distance.
    - 'Nadaraya-Waston': weight *all* y_i by distance to x_i.

$$y_i = \frac{\sum_{j=1}^{n} w_{ij} y_j}{\sum_{j=1}^{n} w_{ij}}$$

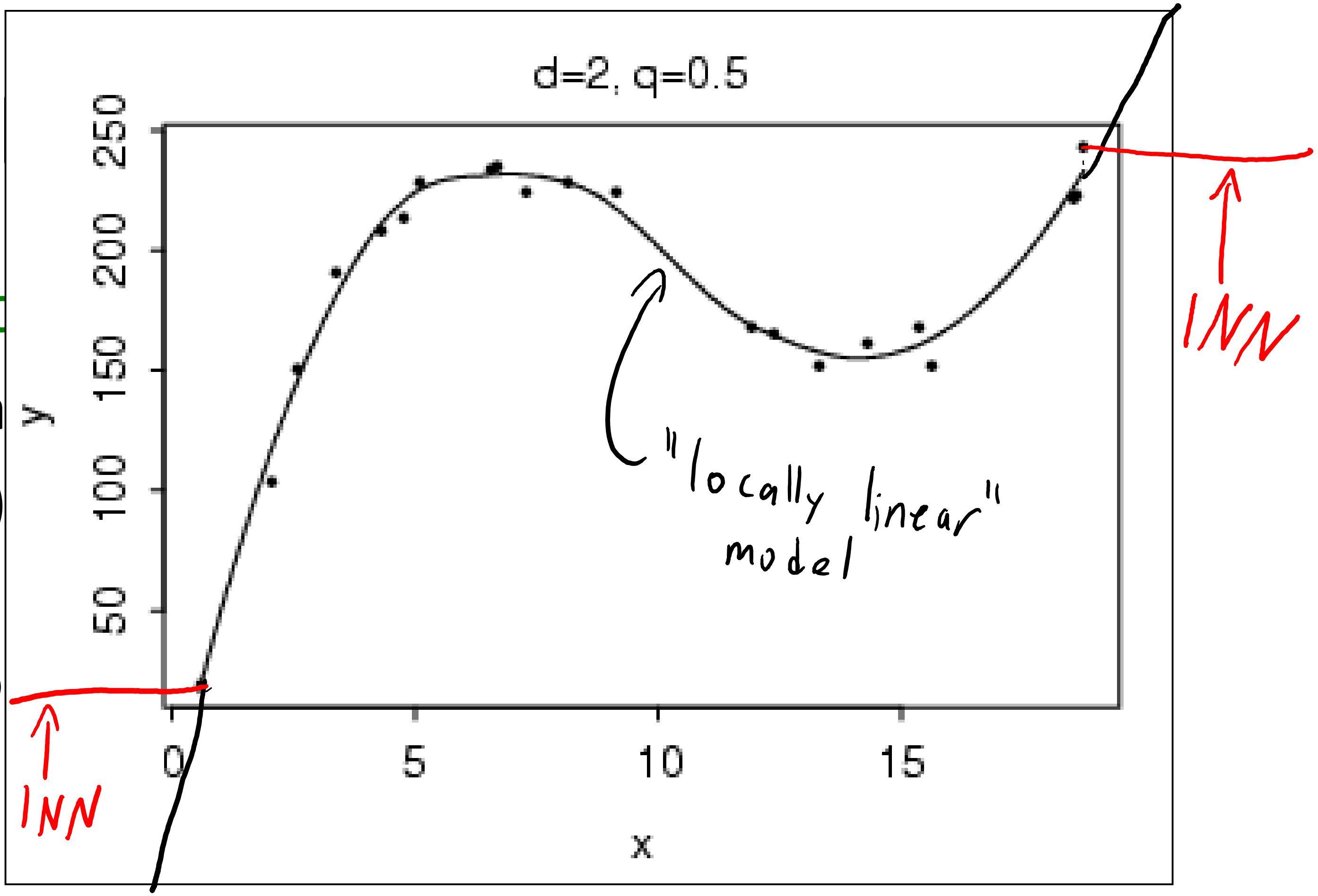


http://www.mathworks.com/matlabcentral/fileexchange/35316-kernel-regression-with-variable-window-width/content/ksr_vw.m

Adapting Counting/

- We can adapt our classificatio
  - Regression tree: tree with mea
  - Generative models: fit $p(x_i \mid y_i)$
  - Non-parametric models:
    - Mean y_i among k-nearest neighb
    - Could be weighted by distance.
    - 'Nadaraya-Waston': weight *all* y_i
    - '**Locally linear regression**': for each x_i a fit linear model weighted by distance.

      (Better than KNN and NW at boundaries.)



Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Generative models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - Mean y_i among k-nearest neighbours.
    - Could be weighted by distance.
    - 'Nadaraya-Waston': weight *all* y_i by distance to x_i.
    - 'Locally linear regression': for each x_i, fit linear model weighted by distance. (Better than KNN and NW at boundaries.)
  - Ensemble methods:
    - Can improve performance by averaging across regression models.

http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd423.htm

Regression Forests for Fluid Simulation

- https://www.youtube.com/watch?v=kGB7Wd9CudA

Linear Least Squares for Quadratic Models

- Can we use linear least squares to fit a quadratic model?

$$y_i = w_0 + w_1 x_i + w_2 x_i^2$$

- You can do this by changing the features (change of basis):

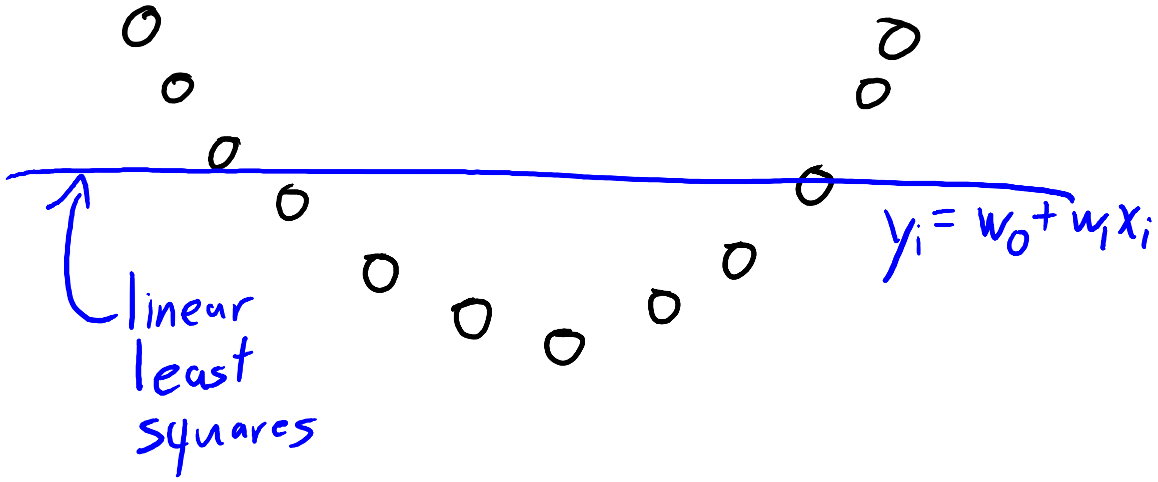
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad \begin{array}{c} Z = \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix} \\ \quad\;\; \text{y-int} \quad\; x \quad\;\; x^2 \end{array}$$

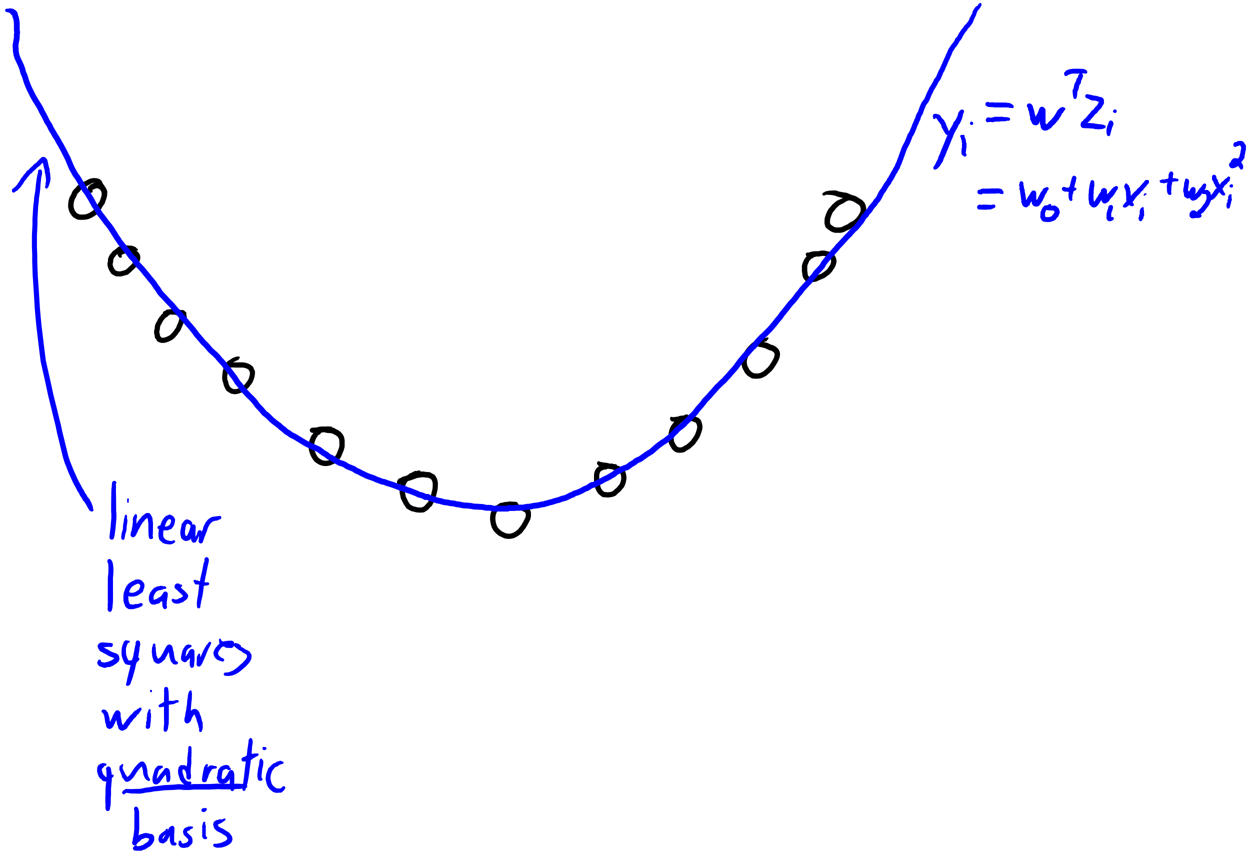
$$\begin{aligned} y_i &= w^T z_i \\ &= w_0 z_{i0} + w_1 z_{i1} + w_2 z_{i2} \\ &= w_0 + w_1 x_i + w_2 x_i^2 \end{aligned}$$

- It's a linear function of w, but a quadratic function of x_i.
- Fitting with least squares: $w = (Z^T Z) \backslash (Z^T y)$

To predict on new data $\hat{X}$, form $\hat{Z}$ from $\hat{X}$ and take $y = \hat{Z} w$

Linear Least Squares for Quadratic Models





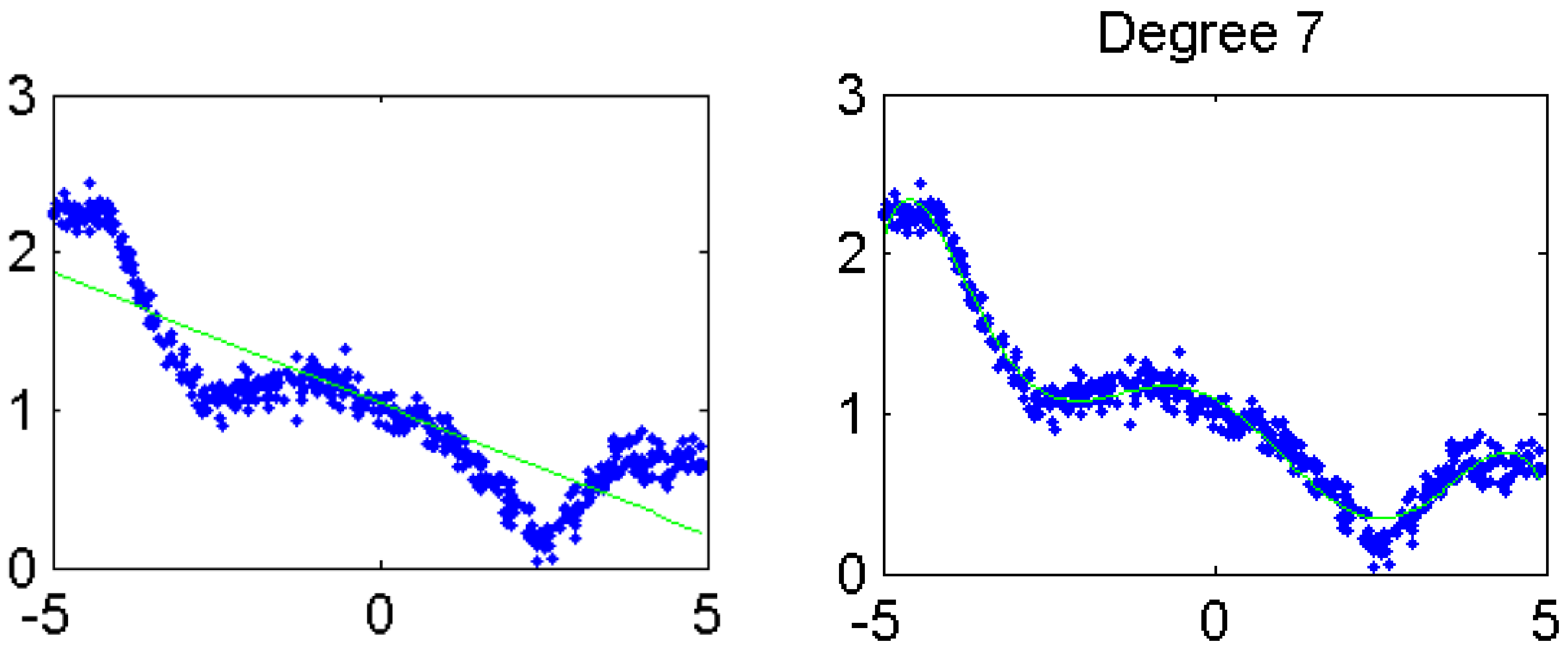
General Polynomial Basis

- We can have a polynomial of degree 'p' by using a basis:

$$Z = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$$

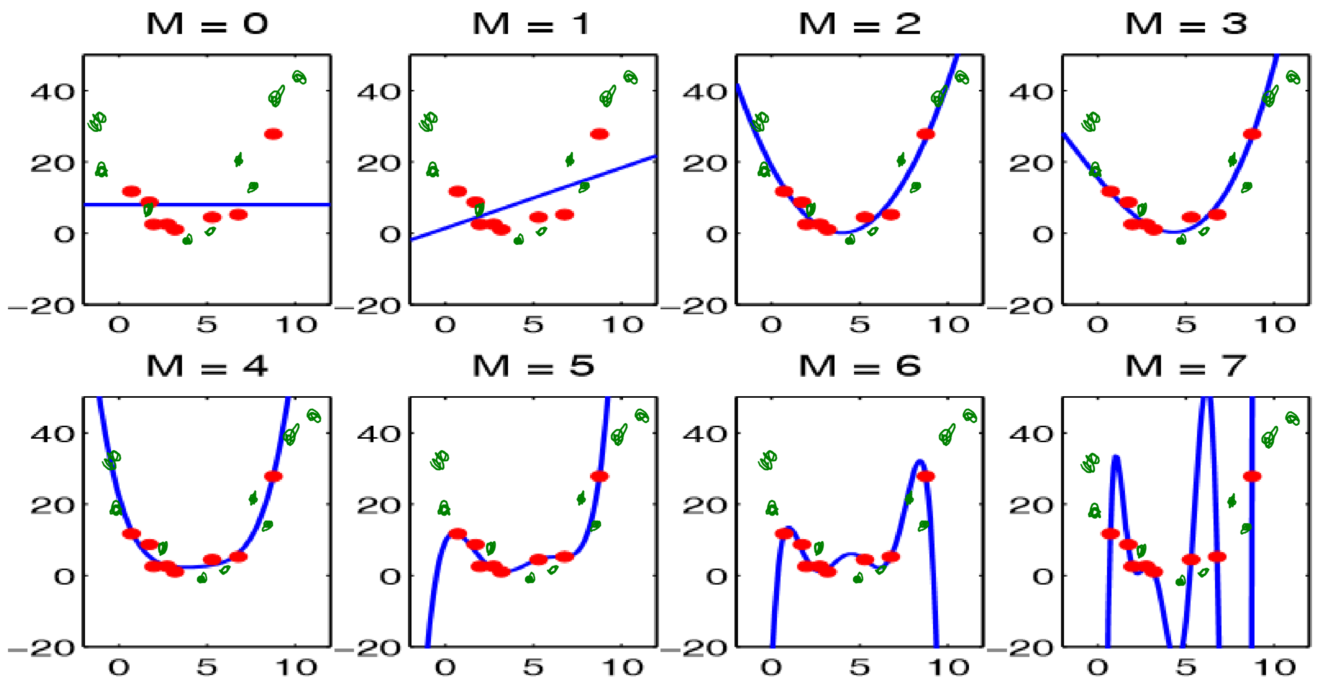
- There are polynomial basis functions that are numerically nicer:
  - E.g., Lagrange polynomials.

General Polynomial Basis



Degree of Polynomial and Fundamental Trade-Off

- As the polynomial degree increases, the training error goes down.



- But training error becomes worse approximation test error.
- Usual approach to selecting degree: validation or cross-validation.

Summary

- Y-intercept can be modeled by using a column of 1s.
- Linear least squares solution is given by normal equations:
  - Solve $(X^T X)w = X^T y$.
- Tree/generative/non-parametric/ensemble methods for regression.
- Change of basis allows linear models to model non-linear data:

- Next time:
  - Bases that can model any continuous function.

Bonus Slide: Householder(-ish) Notation

- Househoulder notation: set of (fairly-logical) conventions for math.

Use greek letters for scalars: $\alpha = 1$, $\beta = 3.5$, $\gamma = \pi$

Use first/last lowercase letters for vectors: $w = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $a = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, $b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

Assumed to be column-vectors.

Use first/last uppercase letters for matrices: X, Y, W, A, B

Indices use i, j, k.

Sizes use m, n, d, p, and k

hopefully meaning of 'k' is obvious from context

Sets use S, T, U, V

Functions use f, g, and h.

When I write x_i I mean "grab row 'i' of X and make a column-vector with its values."

Bonus Slide: Householder(-ish) Notation

- Househoulder notation: set of (fairly-logical) conventions for math:

Our ultimate least squares notation:

$$f(w) = \frac{1}{2}\|Xw - y\|^2$$

But if we agree on notation we can quickly understand:

$$g(x) = \frac{1}{2}\|Ax - b\|^2$$

If we use random notation we get things like:

$$H(\beta) = \frac{1}{2}\|R\beta - P_n\|^2$$

Is this the same model?