CPSC 340: Machine Learning and Data Mining

Basis and Regularization

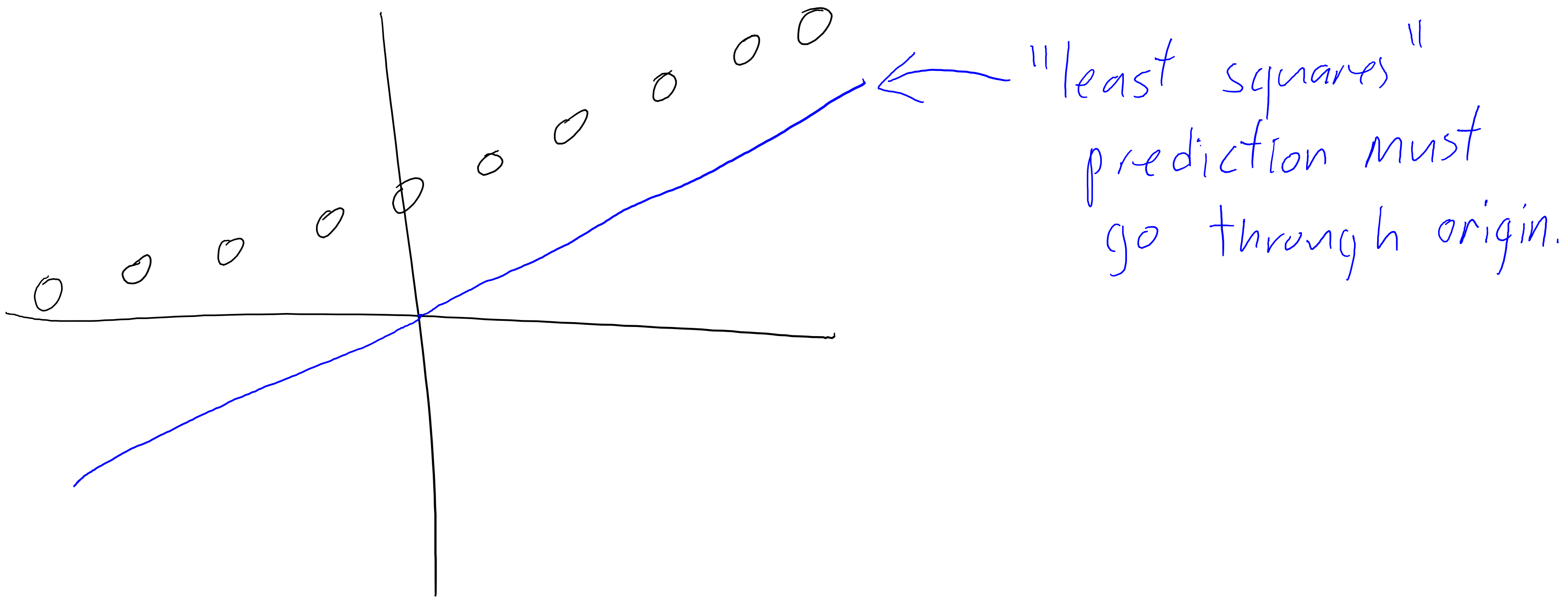
Fall 2015

Admin

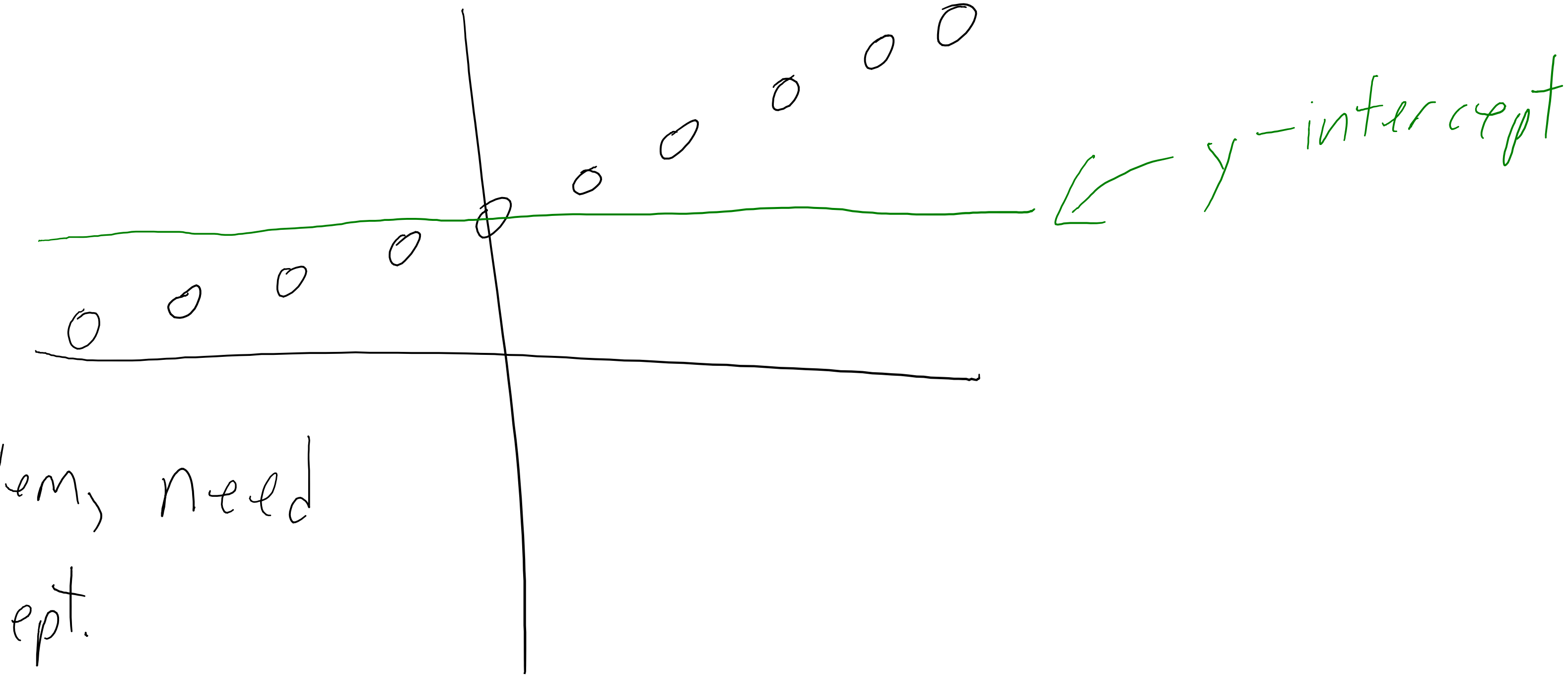
- Re-download a3.pdf (Q1.3 has changed).
- Re-download a3.zip (newsgroups.mat was updated).
- Should we have office hours tomorrow?
- Midterm moved to October 30.

Problem: y-intercept

Since $y_i = w x_i$, if $x_i = 0$ we predict $y_i = 0$.

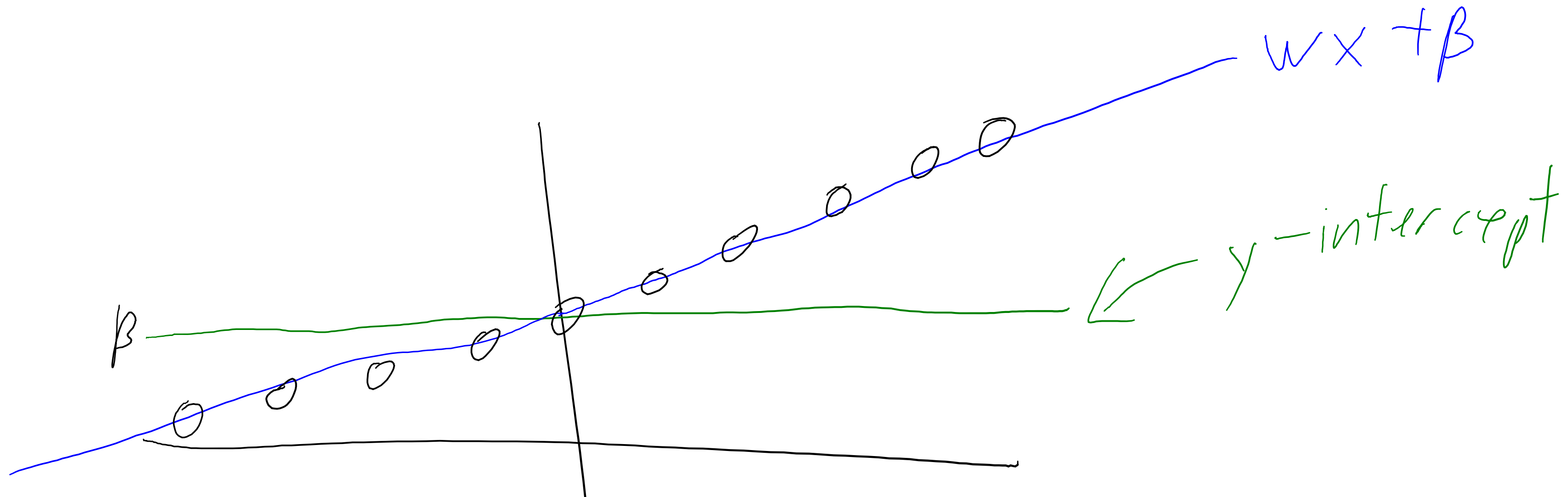


Problem: y-intercept



To fix this problem, need to add y-intercept.

Problem: y-intercept



To fix this problem, need to add y-intercept.

With y-intercept 'β', model becomes $y_i = wx_i + \beta$.

Incorporating a Bias Variable

- The simplest way to add the y-intercept is changing X:

$$X = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.5 \end{bmatrix} \implies \bar{X} = \begin{bmatrix} 1 & 0.1 \\ 1 & 0.2 \\ 1 & 0.5 \end{bmatrix}$$

- Column of '1' values allows us to write as basic linear model:

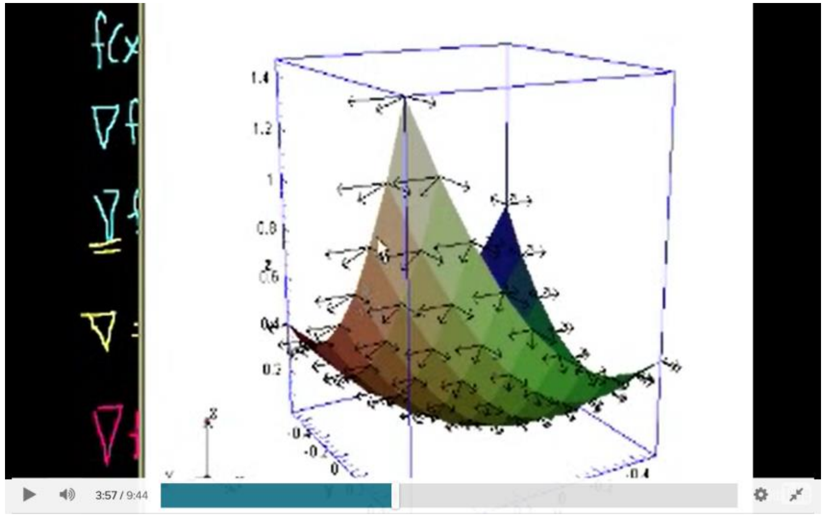
$$\begin{aligned} y_i &= \beta + w x_i \\ &= w_1 + w_2 \bar{x}_{i2} \\ &= w_1 \bar{x}_{i1} + w_2 \bar{x}_{i2} \\ &= \boxed{w^T \bar{x}_i} \end{aligned}$$

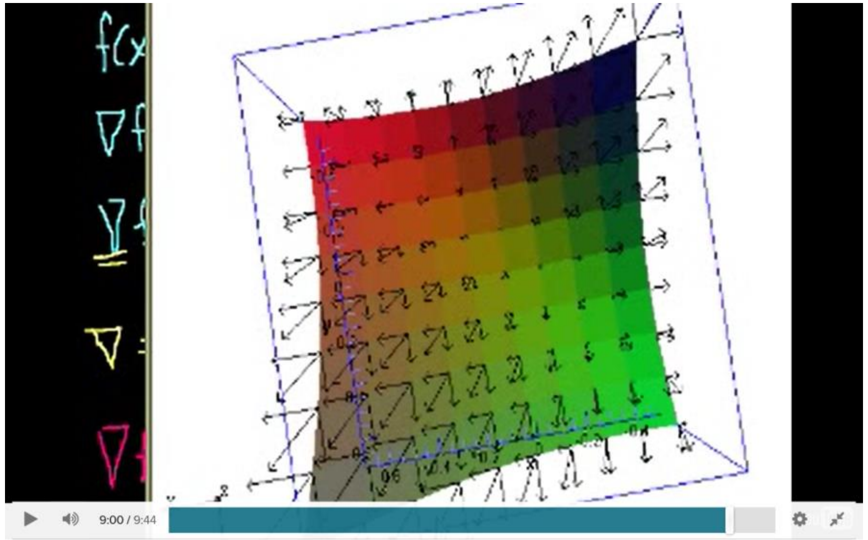
Just fit least squares with $\bar{X}$ as features.
The y-intercept will be w_1, the slope will be w_2.

Gradient Vector

- The gradient vector has the partial derivatives as elements:

$$\nabla f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$





- Element ‘j’ gives the slope if we move along dimension ‘j’.
- Gradient direction points in local direction of steepest increase.
- Negative gradient points in local direction of steepest decrease.
- If $\nabla f(w) = 0$, it means that the function is flat (stationary point).

https://www.khanacademy.org/math/multivariable-calculus/partial_derivatives_topic/gradient/v/gradient-1

Householder-ish Notation

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

Use first/last lower-case letters for vectors (assumed to be column vectors): $w = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $a = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ $b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

Use first/last upper-case letters for matrices: X, Y, W, A, B.

Indices use i, j, and k.

Sizes use m, n, d, and p.

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."

Householder Notation

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:

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

Is this the same model?

Least Squares (Matrix Notation)

- To derive the d-dimensional least squares solution, need matrix notation.

$$y_i = w^T x_i$$

- First let's define the usual suspects:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \qquad X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix} = \begin{bmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_n & — \end{bmatrix}$$

$$n \times 1 \qquad d \times 1 \qquad n \times d \qquad n \times d$$

Least Squares (Matrix Notation)

- Let's define the 'residual' vector:

$$r = \begin{bmatrix} y_1 - w^T x_1 \\ y_2 - w^T x_2 \\ \vdots \\ y_n - w^T x_n \end{bmatrix}, \quad \text{so } \sum_{i=1}^{n} (y_i - w^T x_i)^2 = \sum_{i=1}^{n} r_i^2 = r^T r.$$

we want residuals r_i to be close to zero.

- From the definition of matrix-vector product, we have:

$$Xw = \begin{bmatrix} w^T x_1 \\ w^T x_2 \\ \vdots \\ w^T x_n \end{bmatrix}, \quad \text{so } r = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} w^T x_1 \\ w^T x_2 \\ \vdots \\ w^T x_n \end{bmatrix} = y - Xw.$$

- So we can write least squares as:

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

Least Squares (Matrix Notation)

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

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

$(Ax)^T = x^T A^T$

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

$y^T(y - Xw) - w^T X^T(y - Xw)$

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

$y^T X w = w^T X^T y$

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

$5^T = 5$

Least Squares Solution (Normal Equations)

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

Like $\frac{d}{dw}[aw] = a.$

$\frac{d}{dw}[aw^2] = 2aw.$

$$\nabla f(w) = 0 - \underline{X^T y} + \underline{X^T X w}.$$

If $\nabla f(w) = 0$, then we must have

$$X^T X w = X^T y.$$

Assuming $(X^T X)$ is invertible, 'pre-multiply' by $(X^T X)^{-1}$,

$$(X^T X)^{-1}(X^T X) w = (X^T X)^{-1} X^T y,$$

$$\boxed{w = (X^T X)^{-1} X^T y}$$

Least Squares Issues

- Issues with least squares model:
  - X^TX might not be invertible.
  - 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 known range of y_i values.
  - It assumes a linear relationship between x_i and y_i.

$X \in \mathbb{R}^{n \times d}$

$X^T \in \mathbb{R}^{d \times n}$

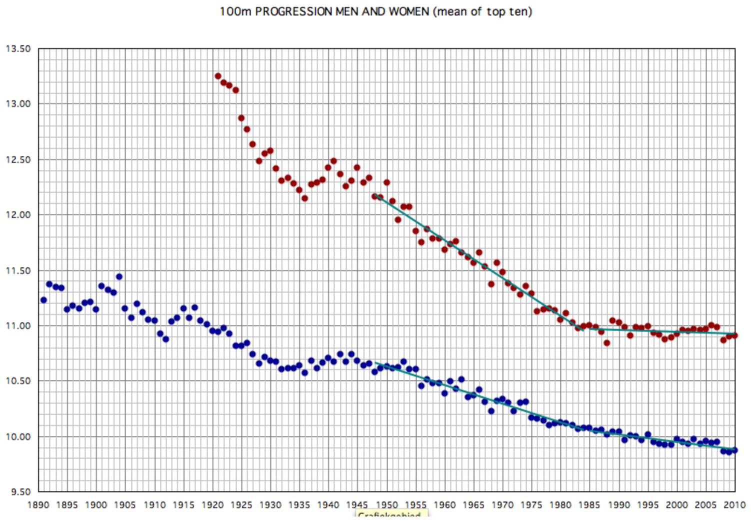
$X^TX \in \mathbb{R}^{d \times d}$

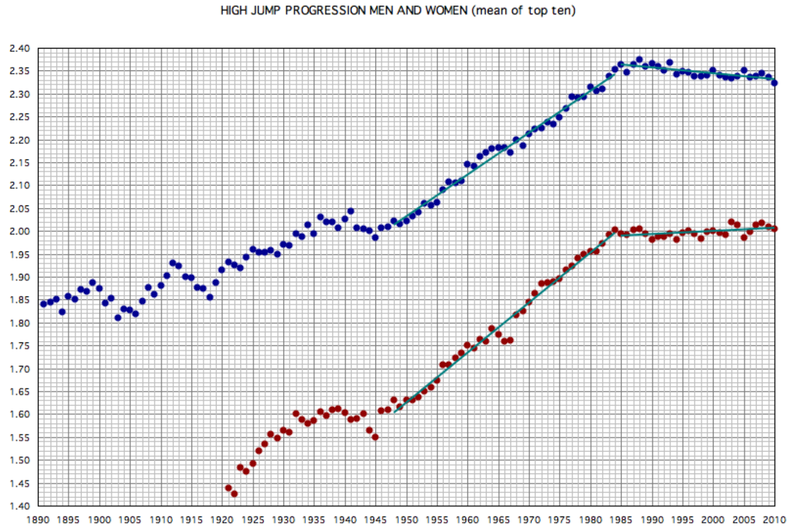
$d \times n \quad n \times d$

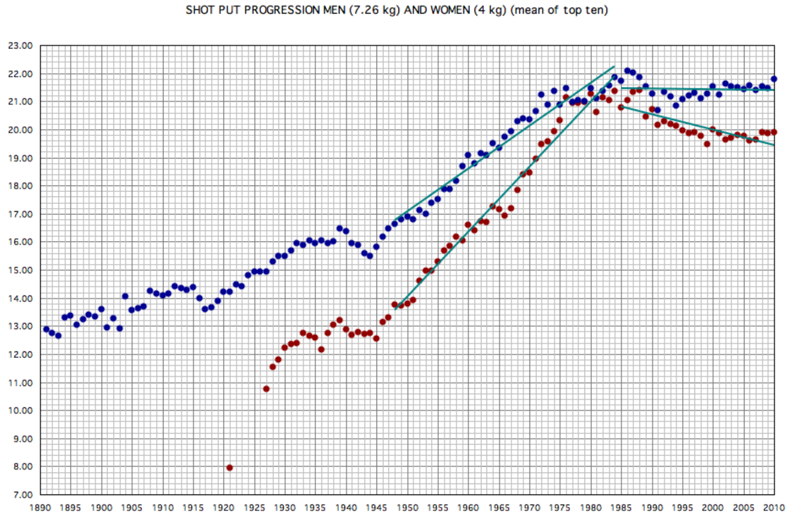
$O(nd^2)$ Cost of computing X^TX: d^2 elements, each is inner product of 'n'

Example: Non-Linear Progressions in Athletics

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



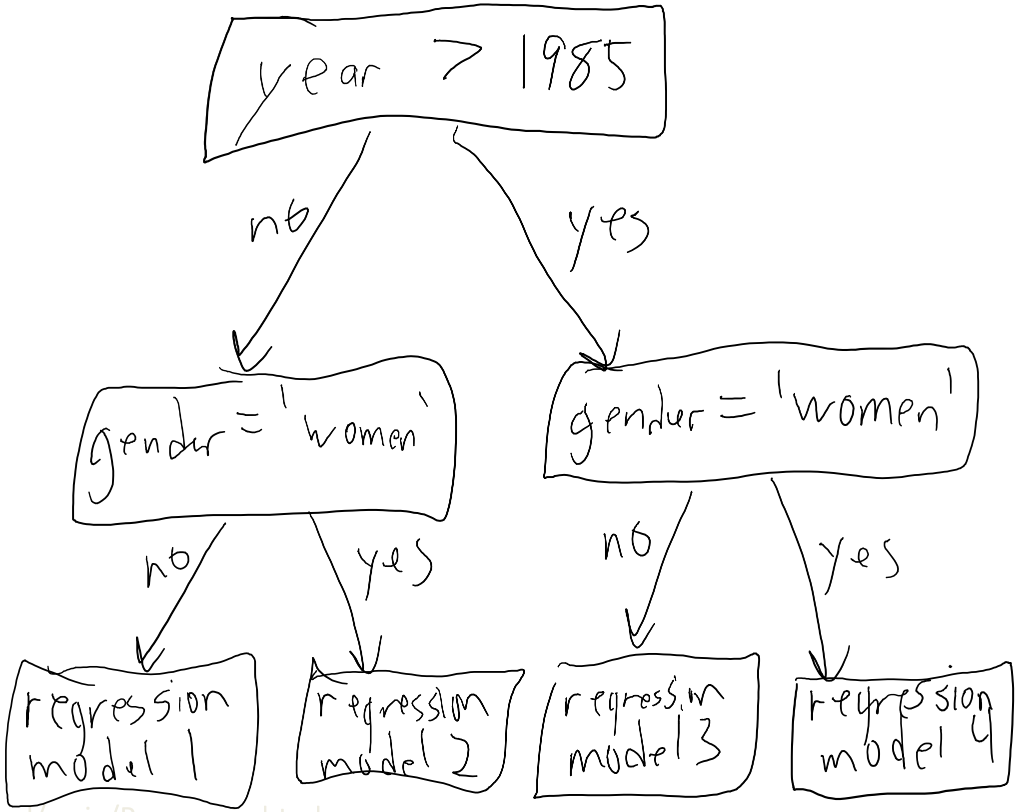


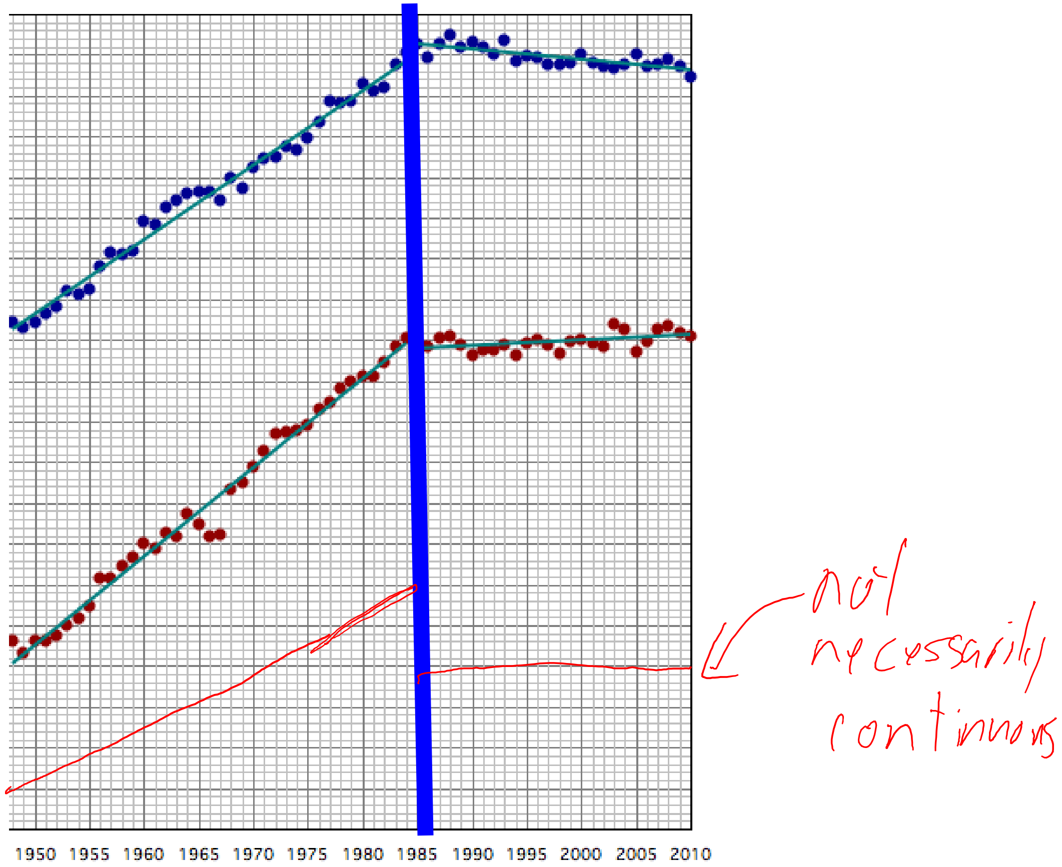


http://www.at-a-lanta.nl/weia/Progressie.html
https://en.wikipedia.org/wiki/Usain_Bolt
http://www.britannica.com/biography/Florence-Griffith-Joyner

Adapting Counting/Distance-Based Methods

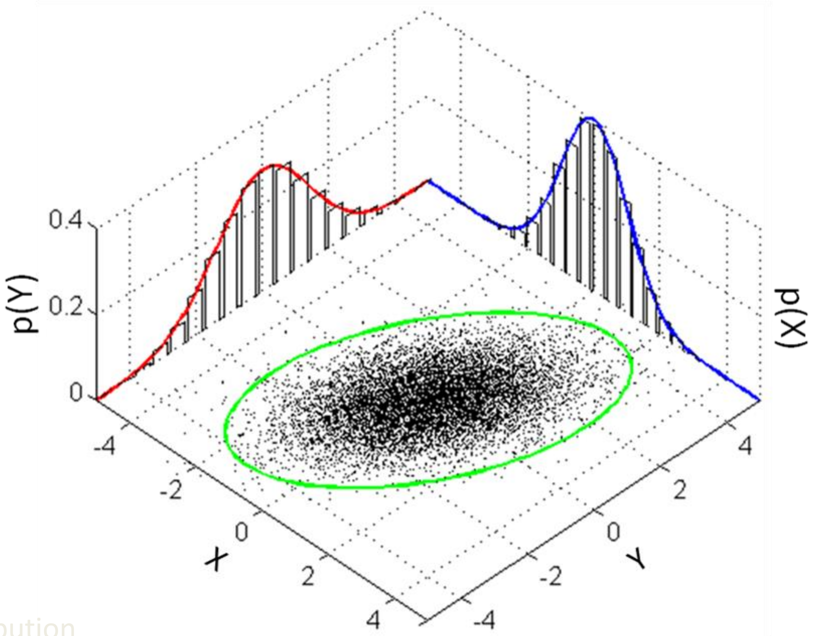
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
    - Gives linear model in each region.





Adapting Counting/Distance-Based Methods

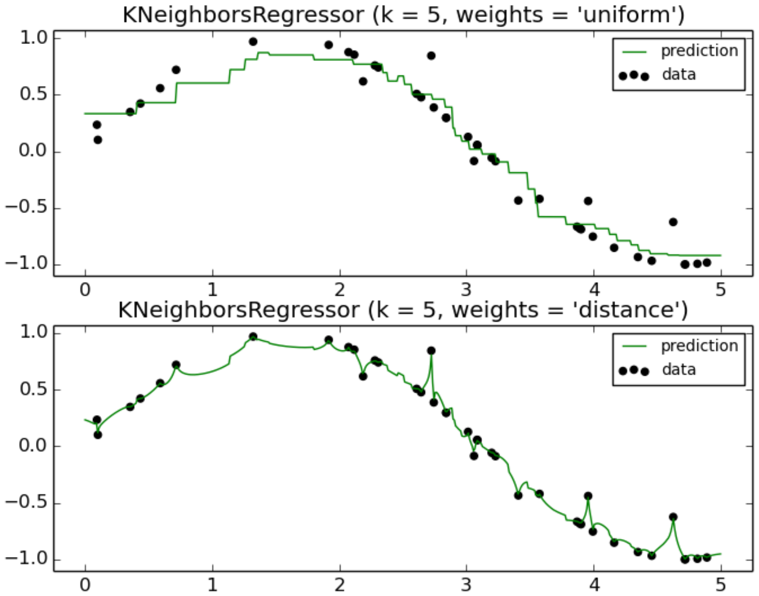
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
    - Gives linear model in each region.
  - Generative models: fit multivariate continuous distribution to (x_i, y_i).
    - E.g., multivariate Gaussian distribution (this choice still gives a linear 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.
    - Gives linear model in each region.
  - Generative models: fit multivariate continuous distribution to (x_i,y_i).
    - E.g., multivariate Gaussian distribution (this choice still gives a linear model).
  - Non-parametric models:
    - Take mean y_i value among k-nearest neighbours.
    - Variation on KNN: weight y_i values by distance. (Closest points get highest weight.)

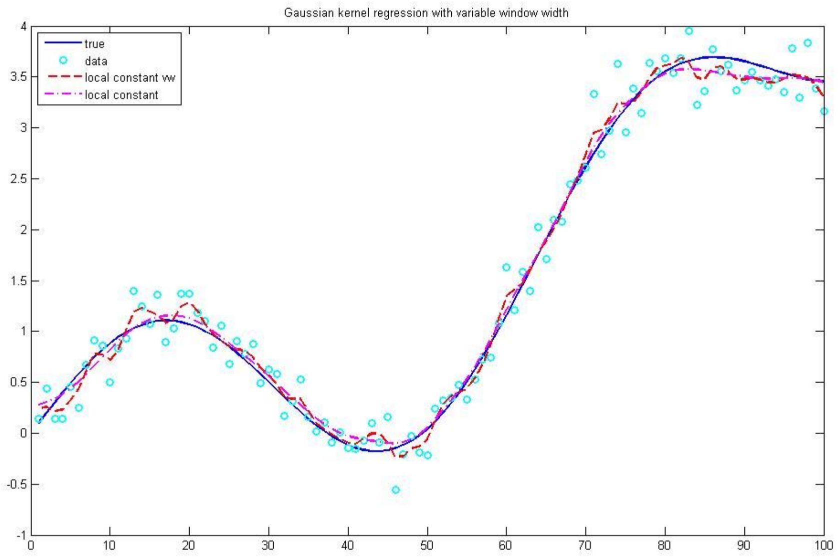


http://scikit-learn.org/stable/modules/neighbors.html

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.
    - Gives linear model in each region.
  - Generative models: fit multivariate continuous distribution to (x_i,y_i).
    - E.g., multivariate Gaussian distribution (this choice still gives a linear model).
  - Non-parametric models:
    - Take mean y_i value among k-nearest neighbours.
    - Variation on KNN: weight y_i values by distance.
    - 'Nadaraya-Waston': weight *all* y_i by distance to x_i.

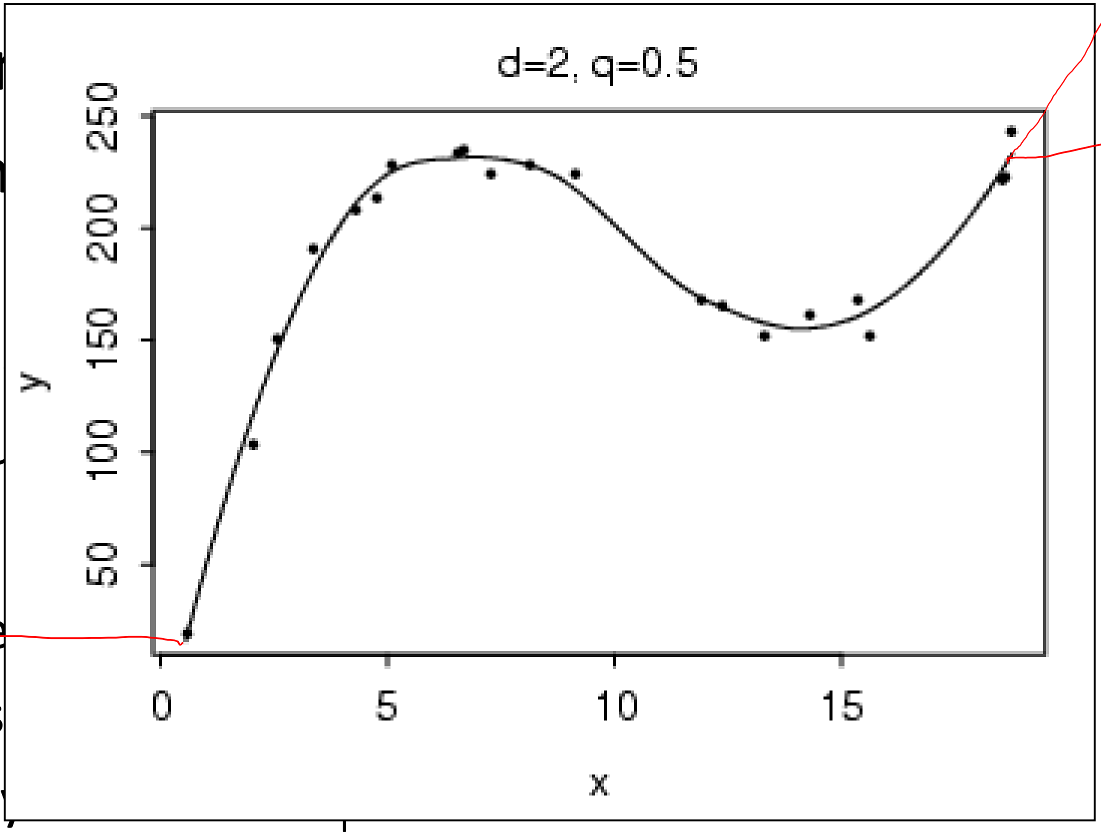
$$\hat{y}_i = \frac{\sum_{j=1}^{n} K(x_i, x_j) y_j}{\sum_{j=1}^{n} K(x_i, x_j)}$$



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

Adapting Counting/Distance-Based Methods

- We can adapt our classificatio
  - Regression tree: tree with mean
    - Gives linear model in each region.
  - Generative models: fit multivari
    - E.g., multivariate Gaussian distribu
  - Non-parametric models:
    - Take mean y_i value among k-neare
    - Variation on KNN: weight y_i values
    - 'Nadaraya-Waston': weight *all* y_i by
    - 'Locally linear regression': for given x, fit least squares with errors weighted by distance from x_i to x. (Better behaviour than KNN and NW at boundaries.)



locally linear becomes linear

KNN will be weird at boundaries.

Change of Basis

- What if instead of a linear function, we want a quadratic function?

$$y_i = \underbrace{w_0}_{\beta} + \underbrace{w_1}_{w} x_i + \underline{w_2 x_i^2}.$$

- We can do this by changing X (change of basis):

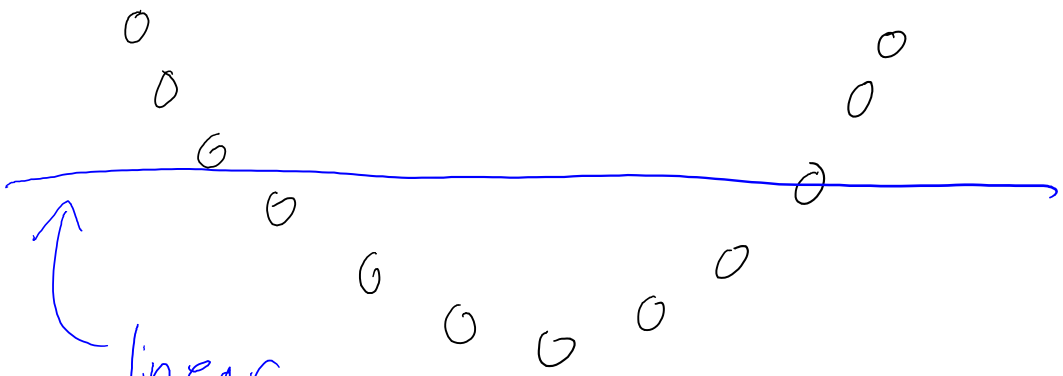
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad X_{poly} = \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix}$$

- Now fit least squares with this matrix:

$$w = (X_{poly}^T X_{poly})^{-1} X_{poly}^T y$$

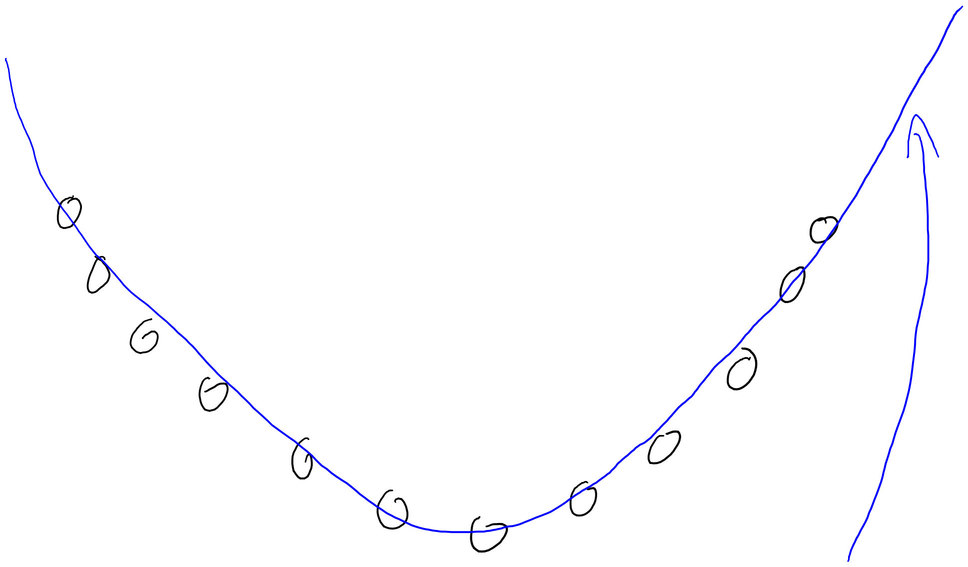
- It's a linear function of w, but a quadratic function of x.

Change of Basis



linear least squares

$y_i = w_1 x_i + w_0$



linear least squares with quadratic basis

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

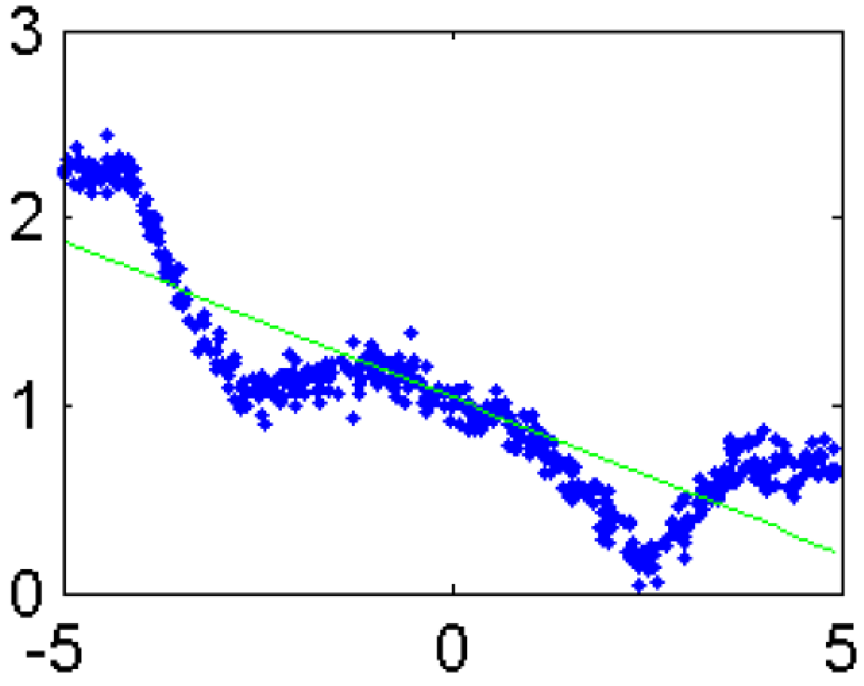
General Polynomial Basis

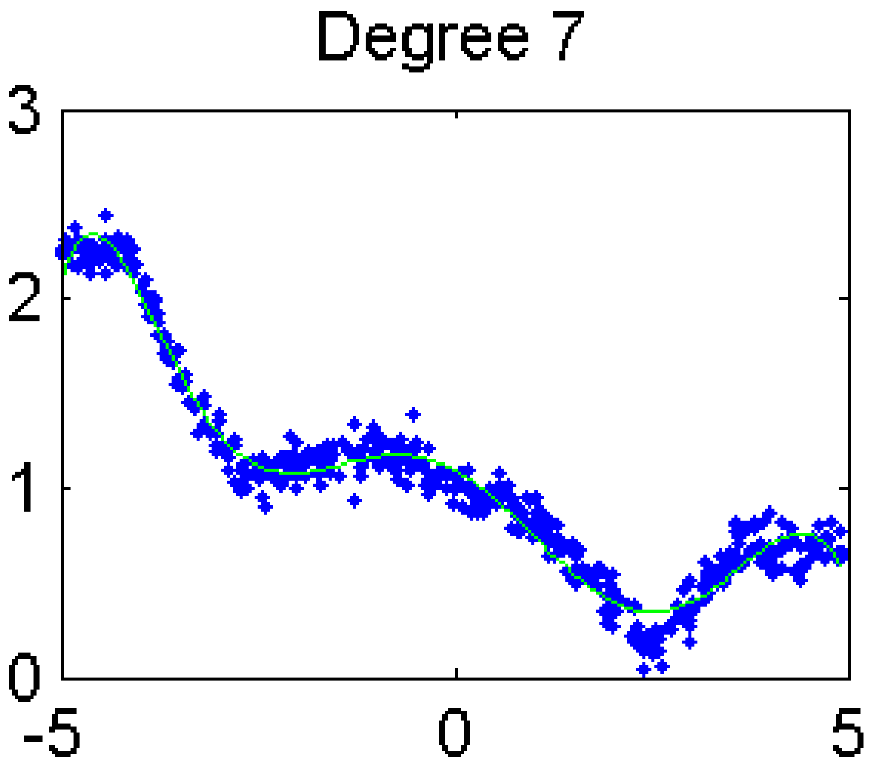
- We can have a polynomial of degree of ‘d’ by using a basis:

$$X_{poly} = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^d \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^d \end{bmatrix}$$

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

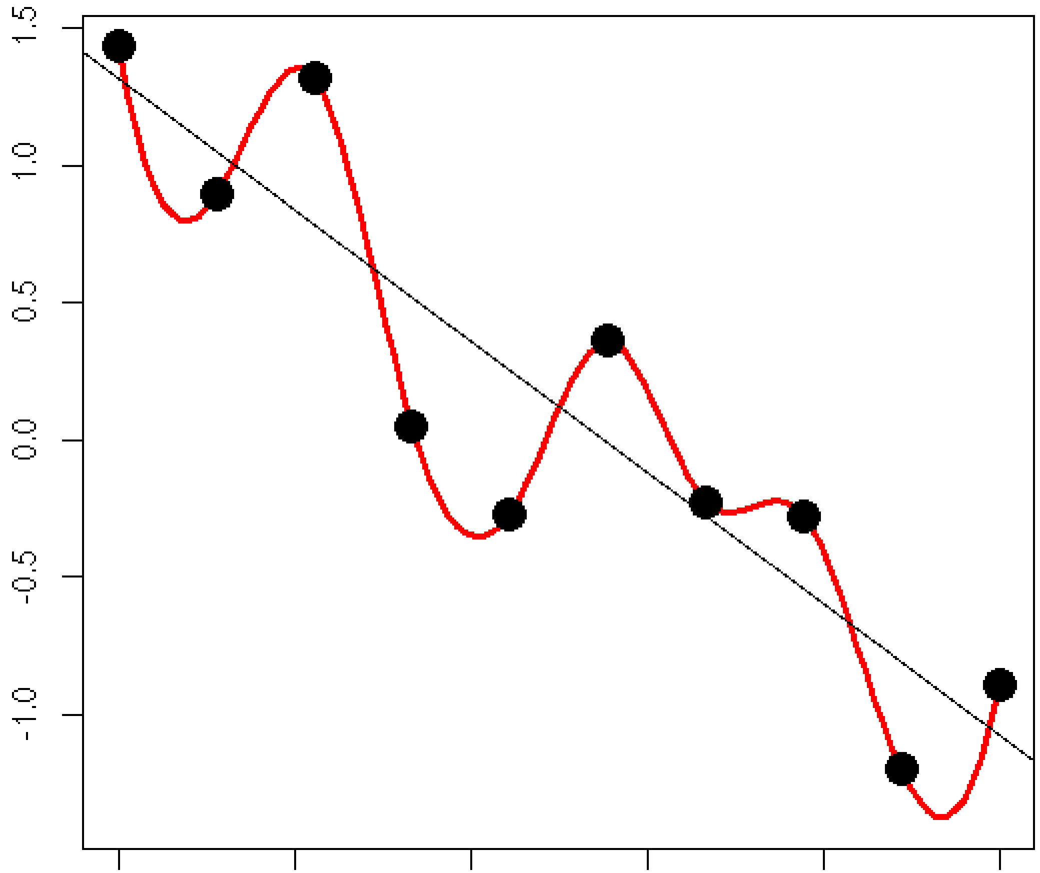
General Polynomial Basis





Degree of Polynomial and Fundamental Trade-Off

- As degree increases:
  - Training error goes down.
  - Training error becomes worse approximation of test error.



- Usual approach to selecting degree:
  - Validation or cross-validation.

http://zoonek2.free.fr/UNIX/48_R/11.html

Bias-Variance Decomposition

- Explicit form of fundamental trade-off for test set squared error:

Assume $y_i = f(x_i) + \varepsilon$, for some function f, and random error ε with mean of 0 and variance σ^2.

Assume we have some way to take a training $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, and produce a model $\hat{y}_i = \hat{f}(x_i)$.

Squared error for test point x_i is

$$E[(y_i - \hat{f}(x_i))^2] = \text{Bias}[\hat{f}(x_i)]^2 + \text{Var}[\hat{f}(x_i)] + \sigma^2$$

Where $\text{Bias}[\hat{f}(x_i)] = E[\hat{f}(x_i)] - f(x_i)$,

$\text{Var}[\hat{f}(x_i)] = E[(\hat{f}(x_i) - E[\hat{f}(x_i)])^2]$,

and expectations are with respect to training data.

- Bias: how closely expected model approximates f(x) (part 1).
- Variance: how sensitive model is to the training set (part 2).
- Irreducible error σ^2: randomness in y_i that no method can predict.

Summary

- Normal equations give solution to linear least squares problem.
- Tree/generative/non-parametric methods exist for regression.
- Change of basis allows linear models to model non-linear data:
  - Discussed polynomial and radial basis functions.
- Bias-variance trade-off is example of fundamental trade-off.

- Next time:
  - Predicting the future, and fixing more problems with least squares.