CPSC 340:
Machine Learning and Data Mining

Nonlinear Regression

Fall 2017

Admin

- Assignment 2 is due tonight.
  - 1 late day to hand it in on Monday, 2 for Wednesday.
- Extra office hours
  - Day before the midterm, October 19th at 4pm (ICICS 246).
- Midterm details:
  - Posted on Piazza, with previous midterms.

Feedback from TAs…

- 1 mark out of 150 on 1 assignment is not a big deal.

- Things that will get you 0 on Assignment 2:
  - Missing name and student number on assignment.
  - Not submitting a .zip file named a2.zip (a .rar file is not a .zip file).
  - Not having a .pdf file in a2.zip called a2.pdf.
  - Using the wrong assignment number on handin.

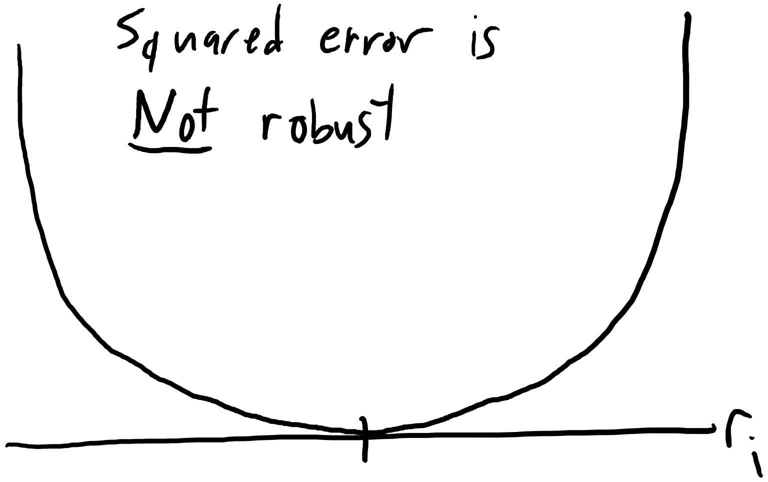
- Things that will get you 0 on individual questions:
  - Not including code in the .pdf file at the right spot.
    (Though you can just include *changed* parts of code, just say where you make changes.)

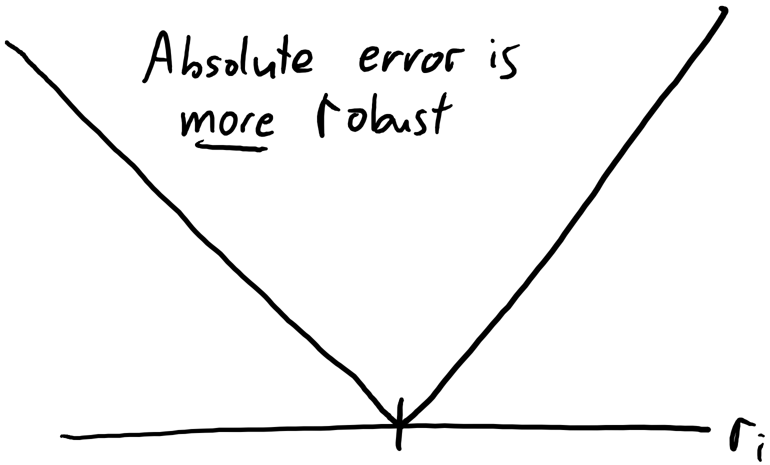
- Things that can get you 0 in the course:
  - Submitting someone else's work without citing them.

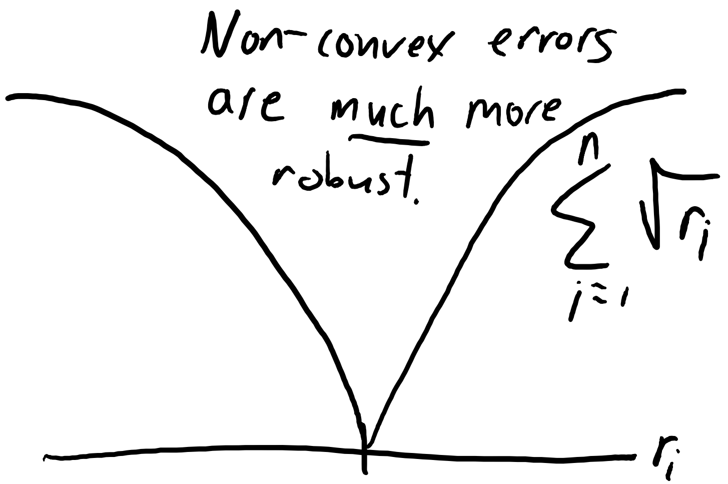
Summary of Last Lecture (Memorize This)

1. Error functions:
   - Squared error is sensitive to outliers.
   - Absolute (L_1) error and Huber error are more robust to outliers.
   - Brittle (L_∞) error is more sensitive to outliers.
2. L_1 and L_∞ error functions are convex but non-differentiable:
   - Finding 'w' minimizing these errors is harder than squared error.
3. We can approximate these with convex differentiable functions:
   - L_1 can be approximated with Huber.
   - L_∞ can be approximated with log-sum-exp.
4. Gradient descent finds stationary point of differentiable function.
   - "Stationary point" == "critical point" == "a 'w' where $\nabla f(w) = 0$".
5. For convex functions, any stationary point is a global minimum.
   - So gradient descent finds global minimum.

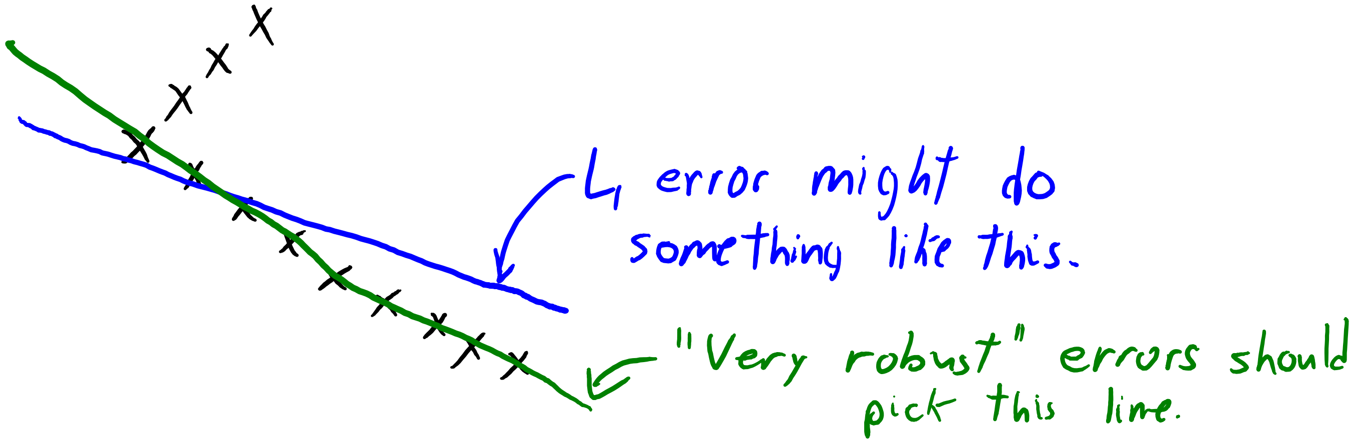
Very Robust Regression



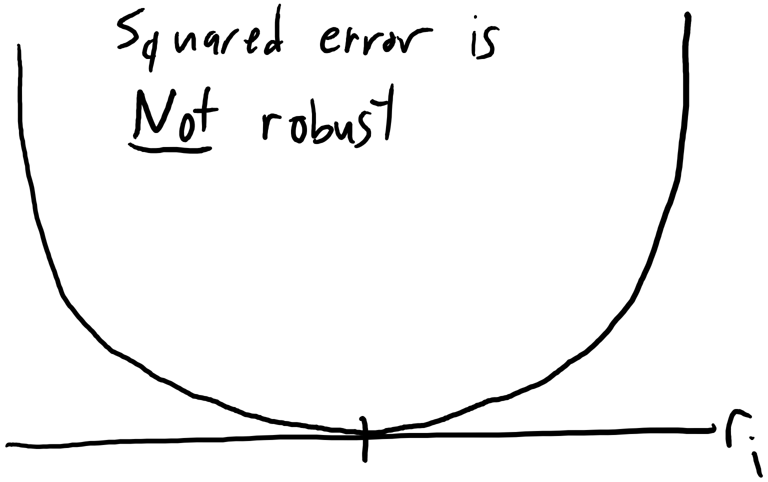


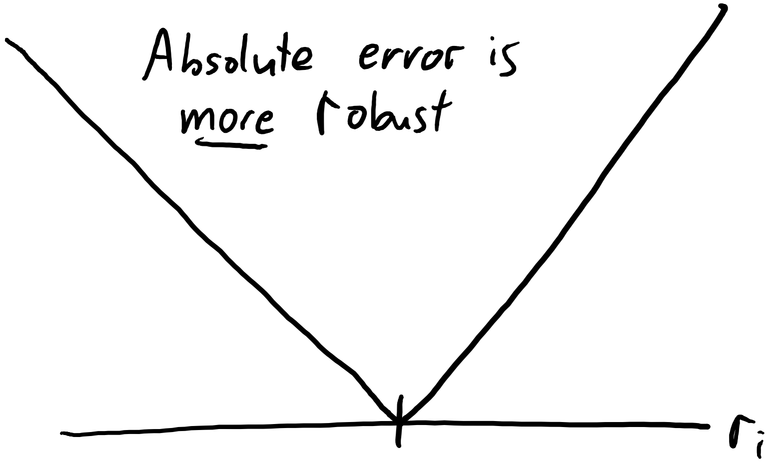


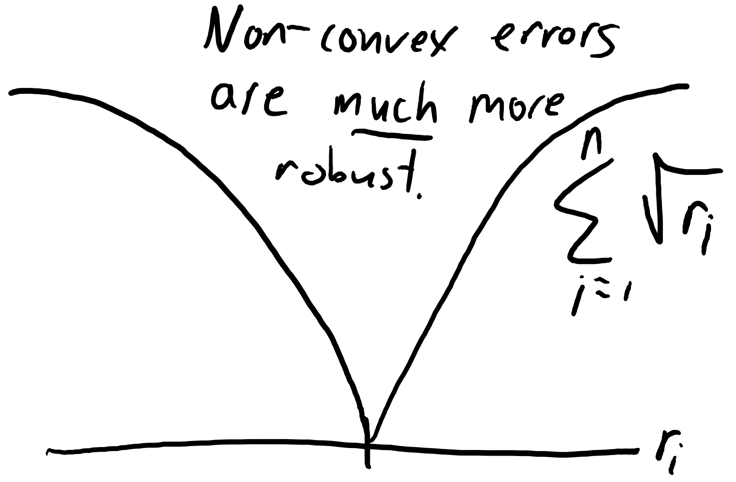
- Non-convex errors can be very robust:
  - Not influenced by outlier groups.



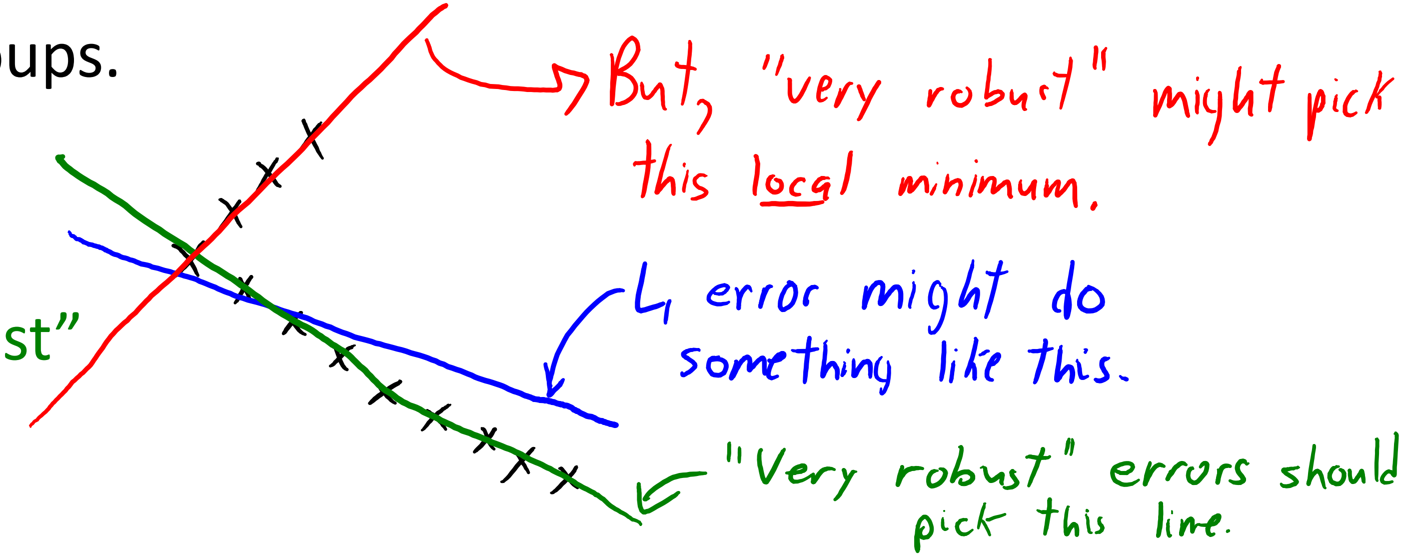
Very Robust Regression







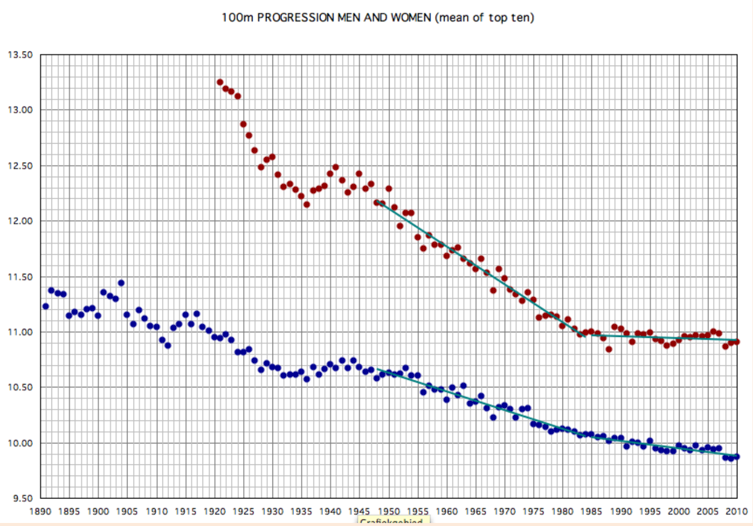
- Non-convex errors can be very robust:
  - Not influenced by outlier groups.
  - But non-convex, so finding global minimum is hard.
  - Absolute value is "most robust" convex loss function.

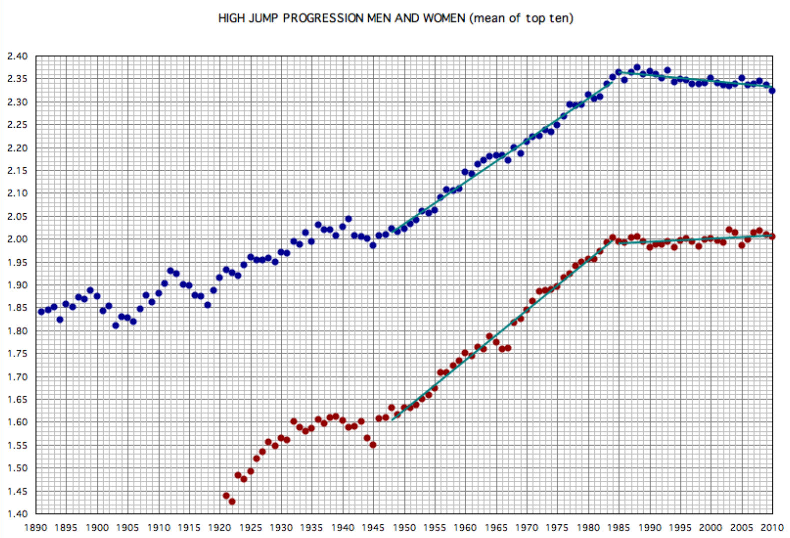


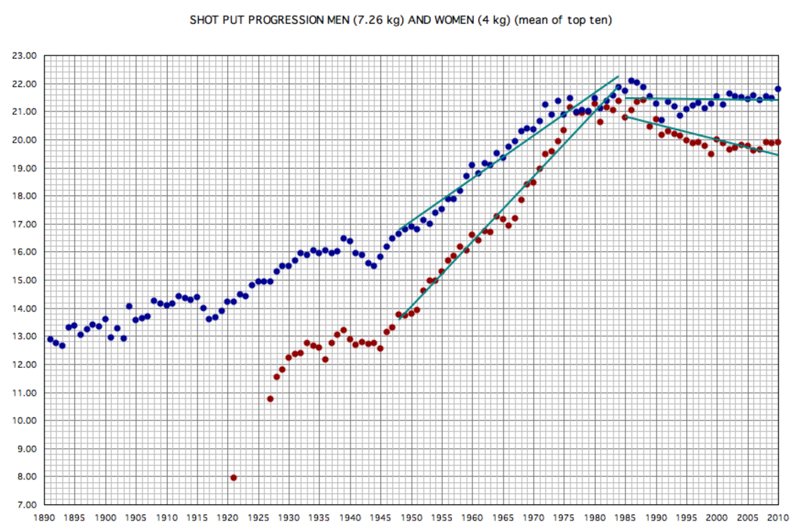
(pause)

Motivation: 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

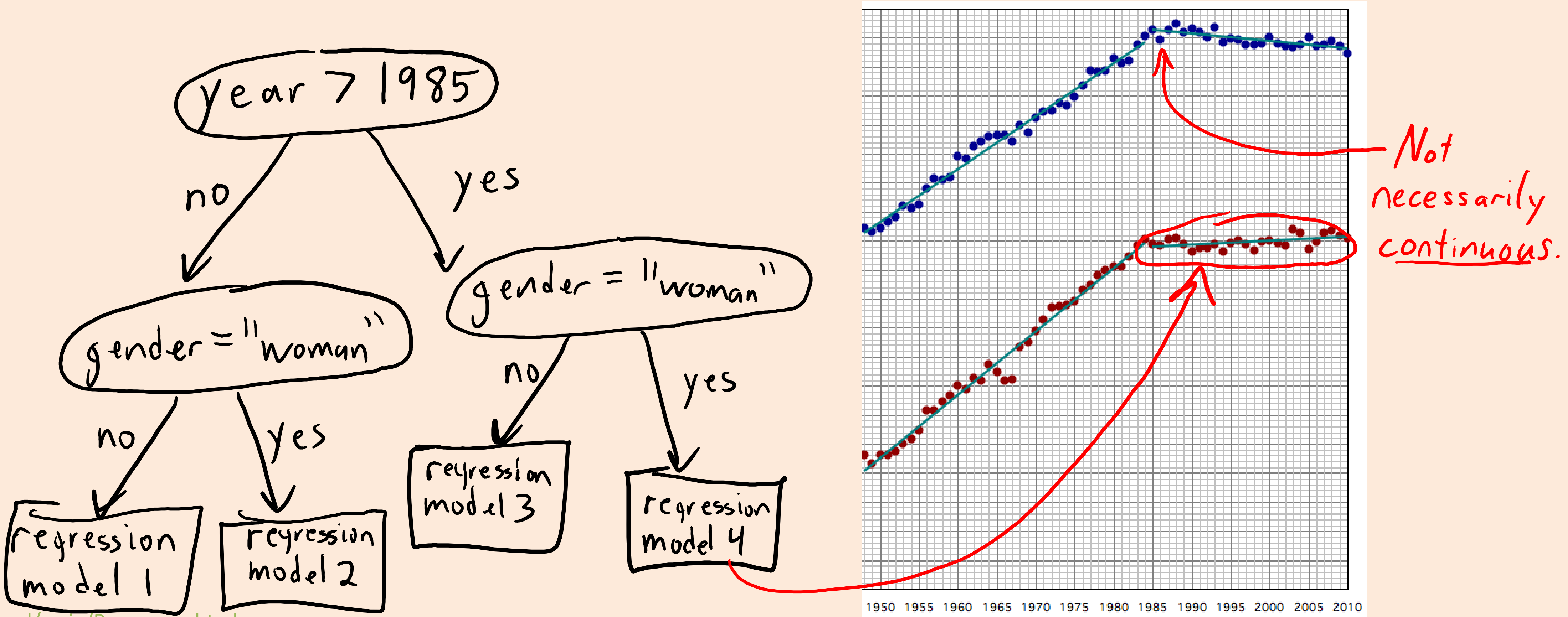
Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:

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

Adapting Counting/Distance-Based Methods

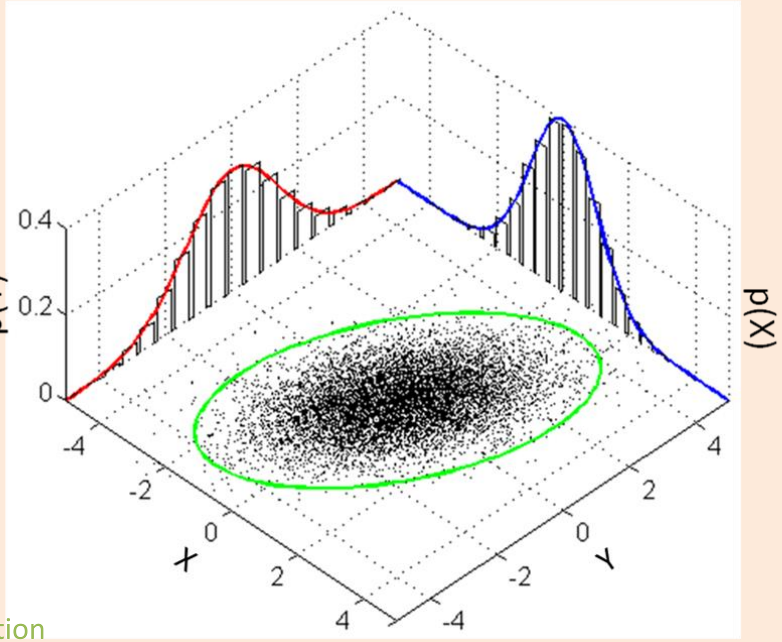
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.



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

Adapting Counting/Distance-Based Methods

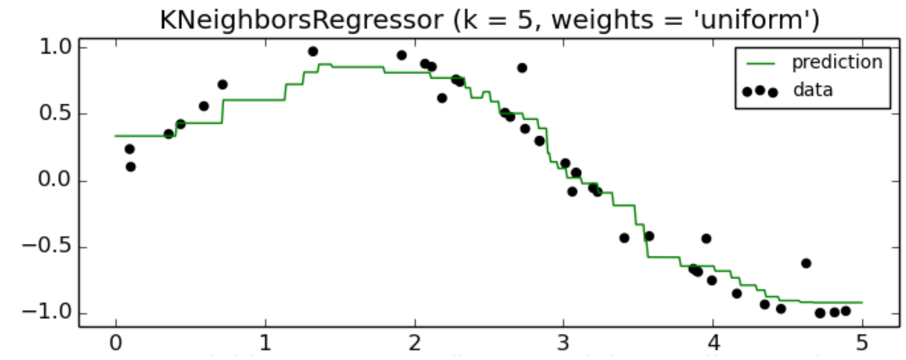
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
    - CPSC 540.



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

Adapting Counting/Distance-Based Methods

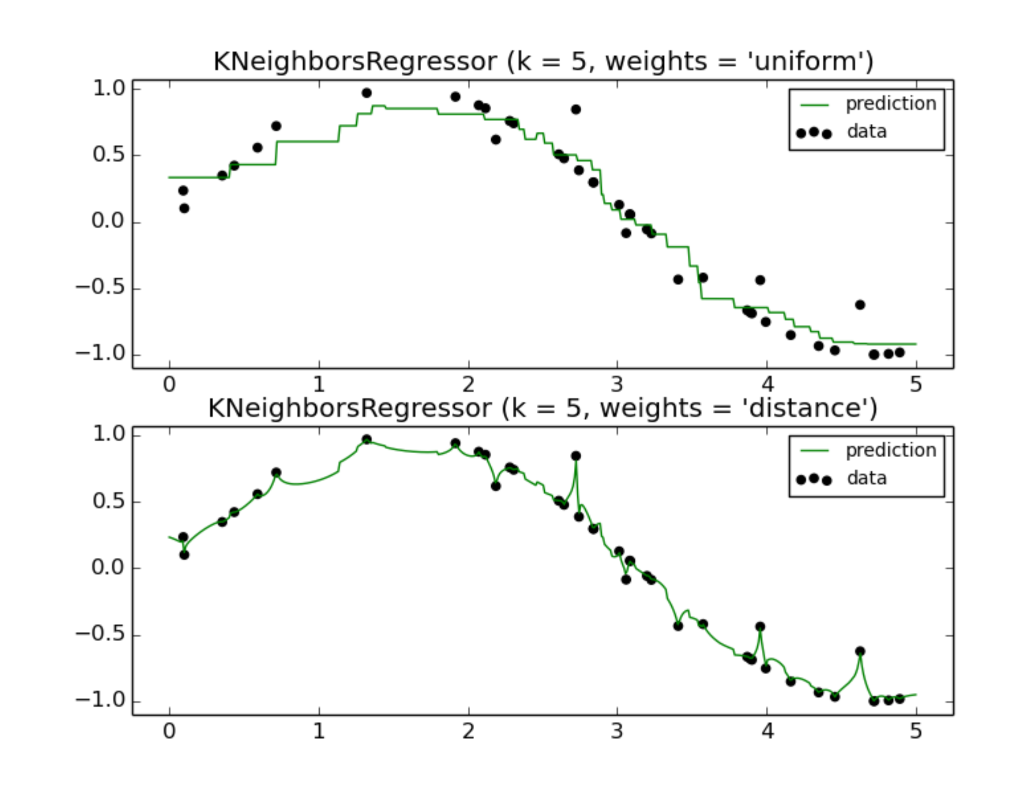
- We can adapt our classification methods to perform regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression:
      - Find 'k' nearest neighbours of $\tilde{x}_i$.
      - Return the mean of the corresponding y_i.



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.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
      - Close points 'j' get more "weight" w_{ij}.

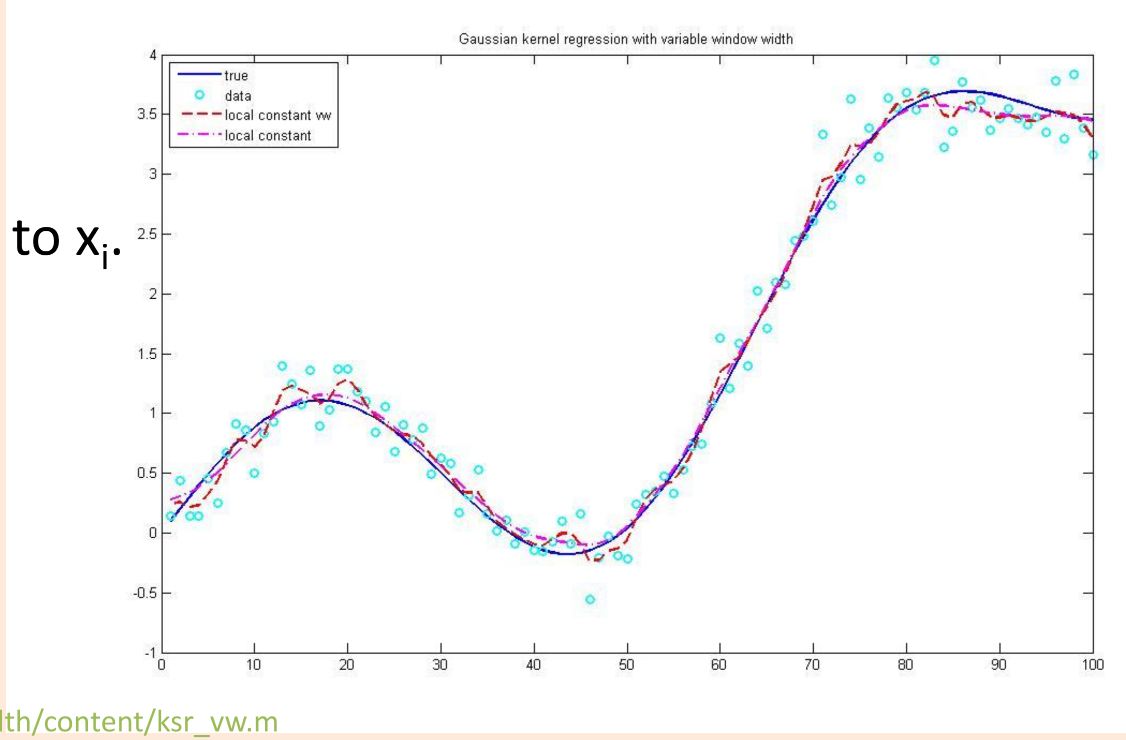


http://scikit-learn.org/stable/modules/neighbors.html

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.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Waston': weight *all* y_i by distance to x_i.

$$\hat{y}_i = \frac{\sum_{j=1}^{n} v_{ij} y_j}{\sum_{j=1}^{n} v_{ij}}$$

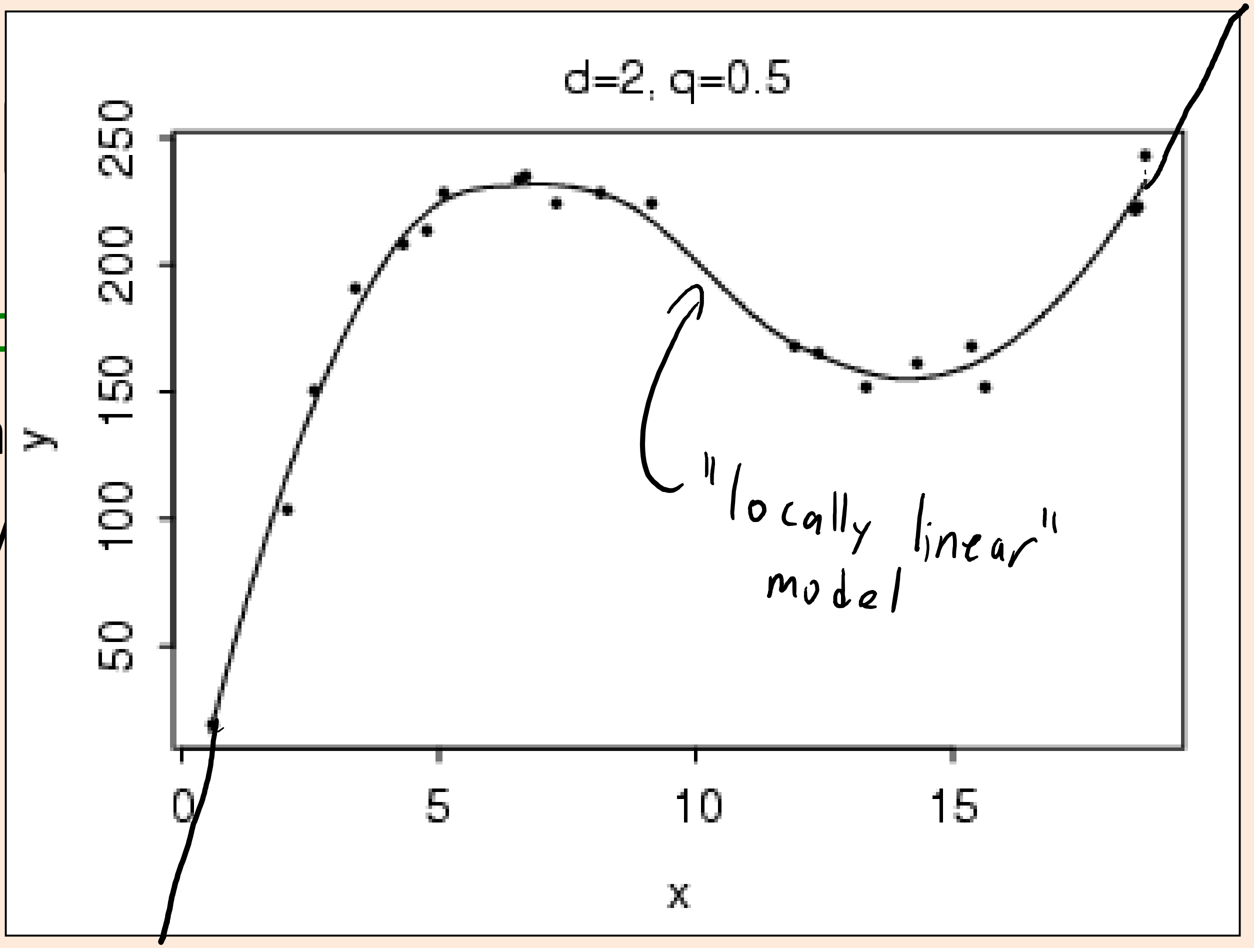


http://www.mathworks.com/matlabcentral/fileexchange/35316-kernel-regression-with-variable-window-width/content/ksr_vw.m

Adapting Counting/

- We can adapt our classificatio
  - Regression tree: tree with mea
  - Probabilistic models: fit p(x_i | y
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - ‘Nadaraya-Waston’: weight *all* y_i
    - ‘Locally linear regression’: for each x_i, fit a linear model weighted by distance.

(Better than KNN and NW at boundaries.)



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

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.
  - Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - Non-parametric models:
    - KNN regression.
    - Could be weighted by distance.
    - 'Nadaraya-Waston': weight *all* y_i by distance to x_i.
    - 'Locally linear regression': for each x_i, fit a linear model weighted by distance. (Better than KNN and NW at boundaries.)
  - Ensemble methods:
    - Can improve performance by averaging across regression models.

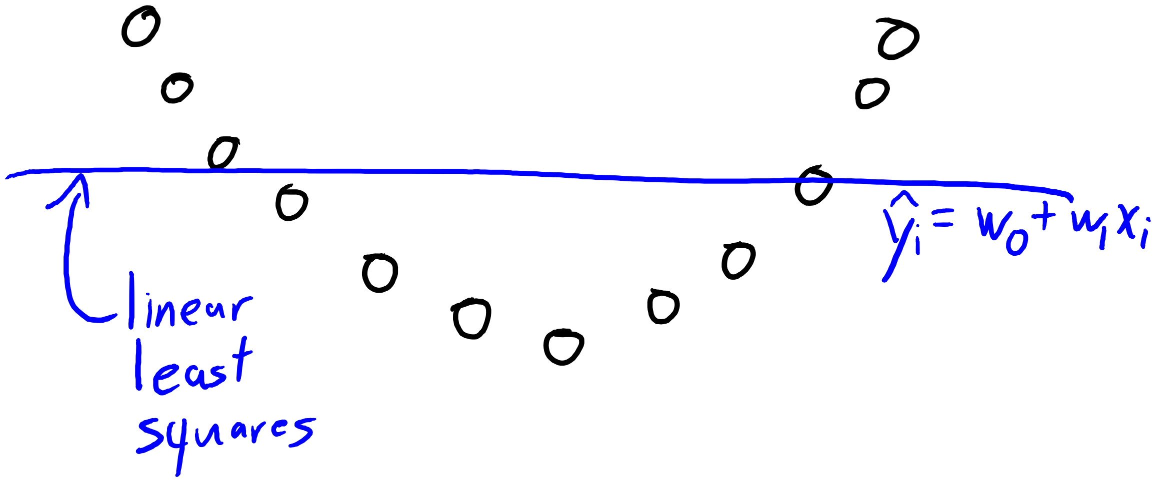
Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression.
- Applications:
  - Regression forests for fluid simulation:
    - https://www.youtube.com/watch?v=kGB7Wd9CudA
  - KNN for image completion:
    - http://graphics.cs.cmu.edu/projects/scene-completion
    - Combined with “graph cuts” and “Poisson blending”.
  - KNN regression for “voice photoshop”:
    - https://www.youtube.com/watch?v=I3l4XLZ59iw
    - Combined with “dynamic time warping” and “Poisson blending”.
- But we’ll focus on linear models with non-linear transforms.
  - These are the building blocks for more advanced methods.

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

Motivation: Limitations of Linear Models

- On many datasets, y_i is not a linear function of x_i.



- Can we use least square to fit non-linear models?

Non-Linear Feature Transforms

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

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

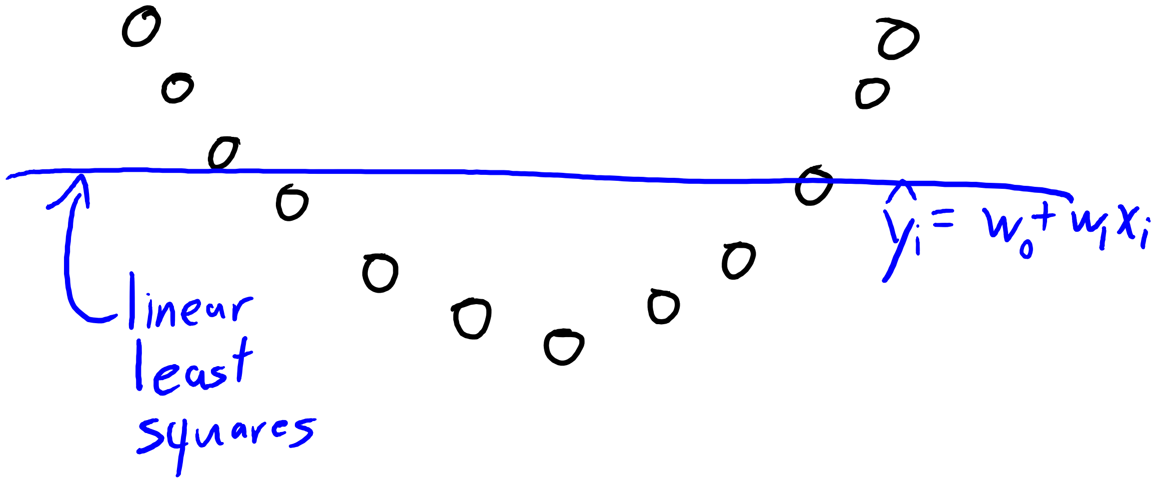
- You can do this by changing the features (change of basis):

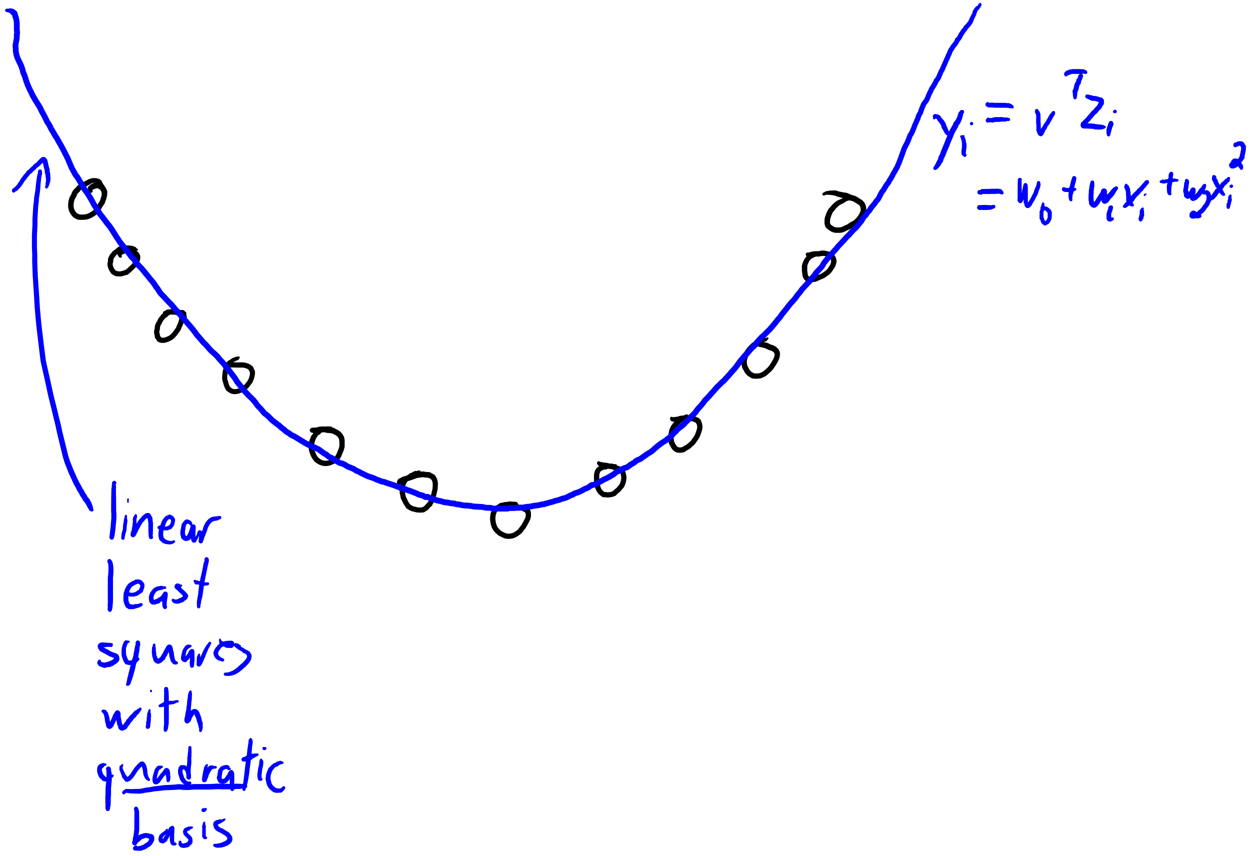
$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad Z = \begin{array}{c} \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix} \\ \begin{matrix} \text{y-int} & x & x^2 \end{matrix} \end{array}$$

- Fit new parameters 'v' under "change of basis": $v = (Z^TZ)^{-1}(Z^Ty)$.
- It's a linear function of w, but a quadratic function of x_i.

$$\hat{y}_i = v^T z_i = \underbrace{v_1}_{w_0} \underbrace{z_{i1}}_{1} + \underbrace{v_2}_{w_1} \underbrace{z_{i2}}_{x_i} + \underbrace{v_3}_{w_2} \underbrace{z_{i3}}_{x_i^2}$$

Non-Linear Feature Transforms





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

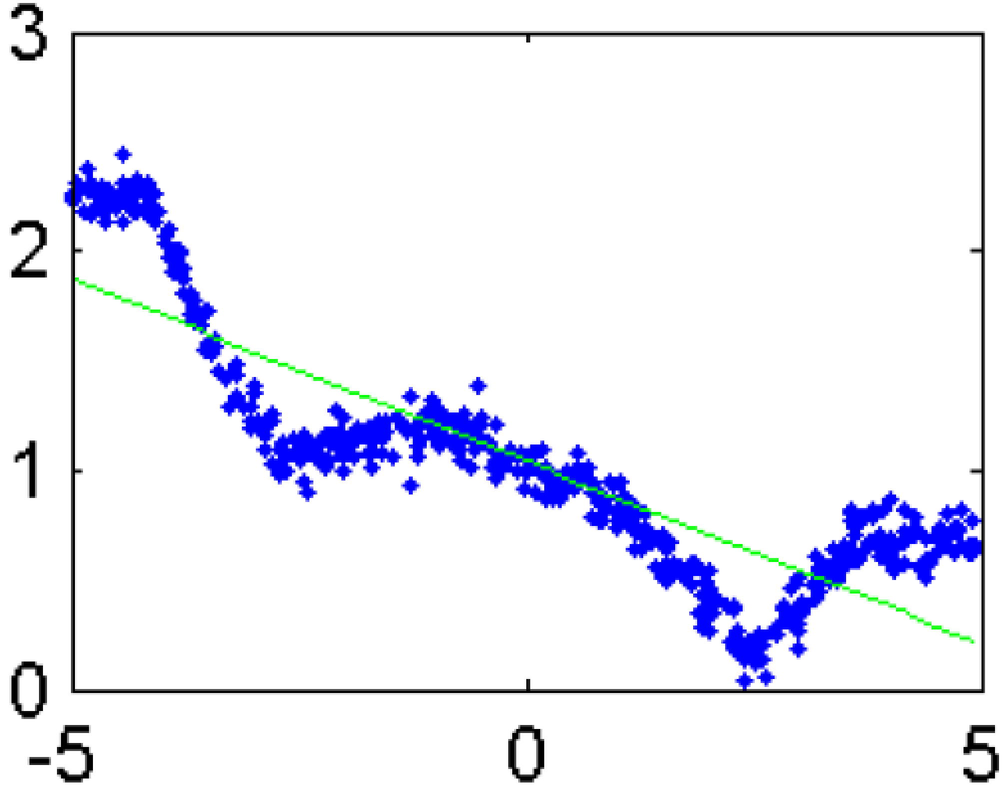
General Polynomial Features (d=1)

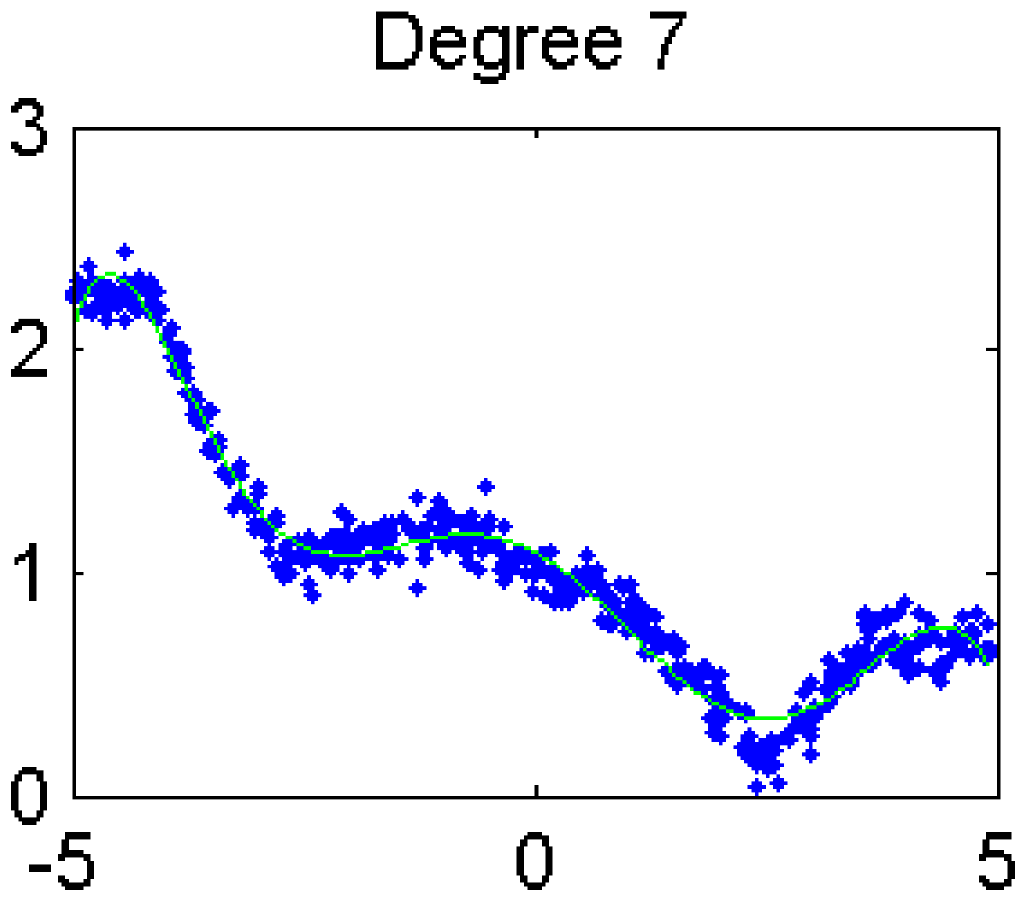
- We can have a polynomial of degree ‘p’ by using these features:

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

- There are polynomial basis functions that are numerically nicer:
  - E.g., Lagrange polynomials (see CPSC 303).

General Polynomial Features

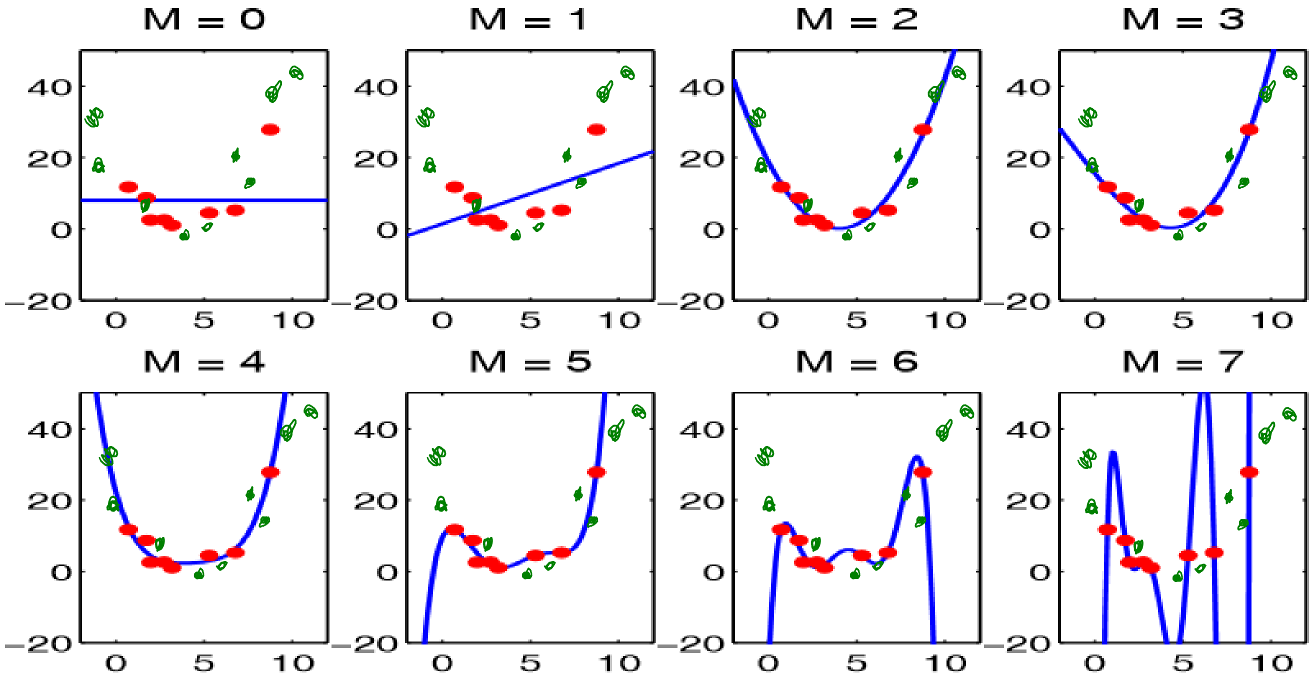




- If you have more than one feature, you include interactions:
  - With p=2, in addition to $(x_{i1})^2$ and $(x_{i2})^2$ you would include $x_{i1}x_{i2}$.

Degree of Polynomial and Fundamental Trade-Off

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

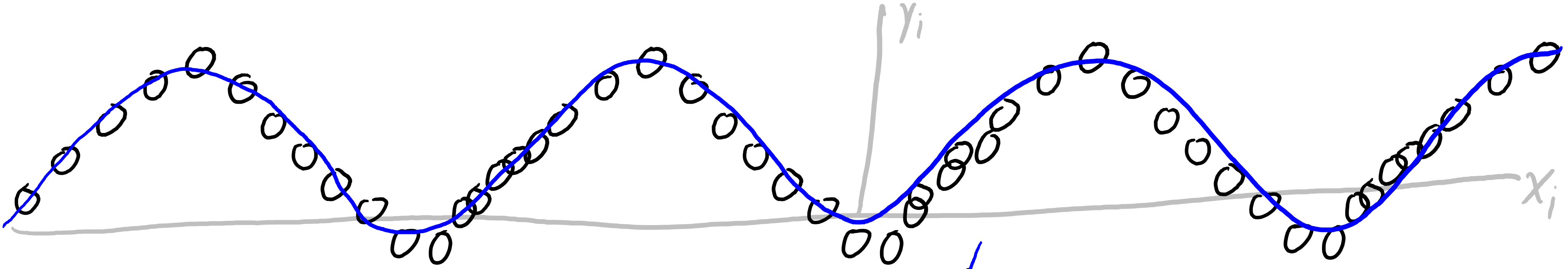


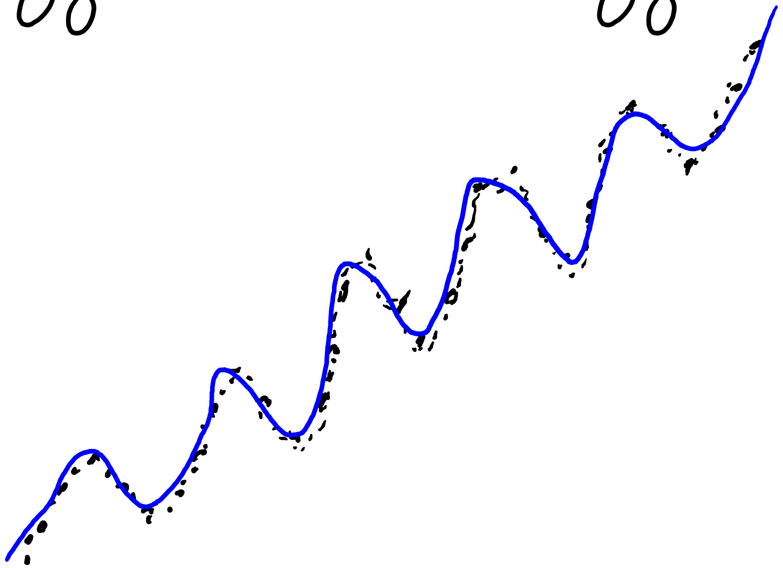
- But approximation error goes up: we start overfitting with large 'p'.
- Usual approach to selecting degree: validation or cross-validation.

http://www.cs.ubc.ca/~arnaud/stat535/slides5_revised.pdf

Beyond Polynomial Transformations

- Polynomials are not the only possible transformation:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right non-linear transform will vastly improve performance.





For periodic data we might use

$$Z = \begin{bmatrix} \sin(x_1) \\ \sin(x_2) \\ \vdots \\ \sin(x_n) \end{bmatrix}$$

$$\hat{y}_i = v^T z_i$$
$$= w_1 \sin(x_i)$$

You can have different types of bases

$$Z = \begin{bmatrix} \underbrace{\begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix}}_{\text{linear}} & \underbrace{\begin{matrix} \sin(6x_1) \\ \sin(6x_2) \\ \vdots \\ \sin(6x_n) \end{matrix}}_{\text{periodic}} \end{bmatrix}$$

End of Scope for Midterm Material.

Finding the "True" Model

- What if our goal is find the "true" model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial 'p'.

- Should we choose the 'p' with the lowest training error?
  - No, this will pick a 'p' that is way too large.
    (training error always decreases as you increase 'p')

Finding the “True” Model

- What if our goal is find the “true” model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial ‘p’.

- Should we choose the ‘p’ with the lowest validation error?
  - This will also often choose a ‘p’ that is too large.

  - Even if true model has p=2, this is a special case of a degree-3 polynomial.
  - If ‘p’ is too big then we overfit, but might still get a lower validation error.
    - Another example of optimization bias.

Complexity Penalties

- There are a lot of "scores" people use to find the "true" model.
- Basic idea behind them: put a penalty on the model complexity.
  - Want to **fit the data and have a simple model**.
- For example, minimize training error plus the degree of polynomial.

Let $Z_p = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ 1 & x_3 & (x_3)^2 & \cdots & (x_3)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$

Find 'p' that minimizes:

$$\text{score}(p) = \underbrace{\frac{1}{2}\|Z_p v - y\|^2}_{\text{train error for best 'v' with this basis.}} + \underbrace{p}_{\text{degree of polynomial}}$$

  - If we use p=4, use "training error plus 4" as error.
- If two 'p' values have similar error, this prefers the smaller 'p'.

Summary

- Tree/probabilistic/non-parametric/ensemble regression methods.
- Non-linear transforms:
  - Allow us to model non-linear relationships with linear models.
- Complexity penalties can counter optimization bias.
  - When we want to find the “true” model.

- Next time:
  - Can we find the “true” features?