

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.

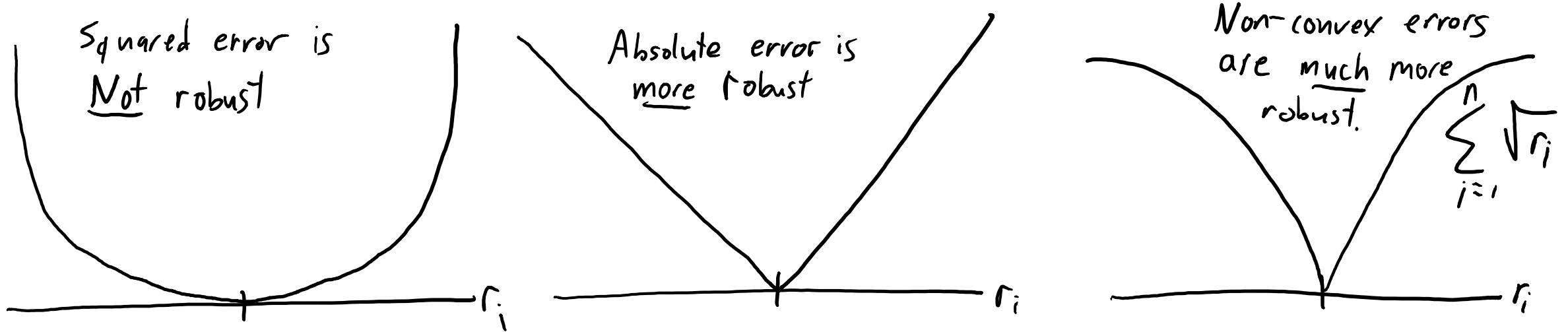
4. Gradient descent finds stationary point of differentiable function.

- “Stationary point” == “critical point” == “a 'w' where $\nabla f(w) = 0$ ”.

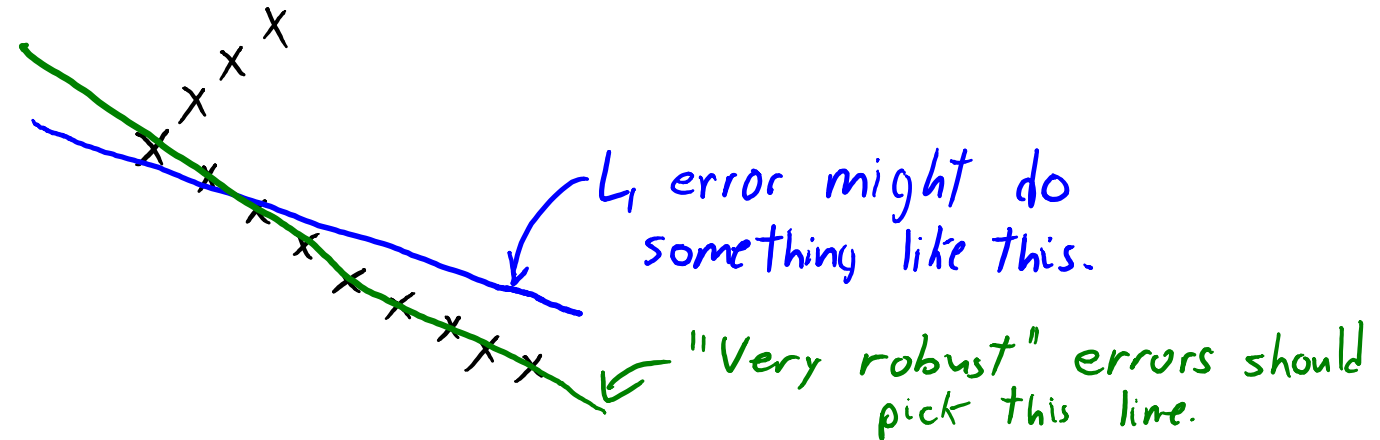
5. For convex functions, any stationary point is a global minimum.

- So gradient descent finds global minimum.

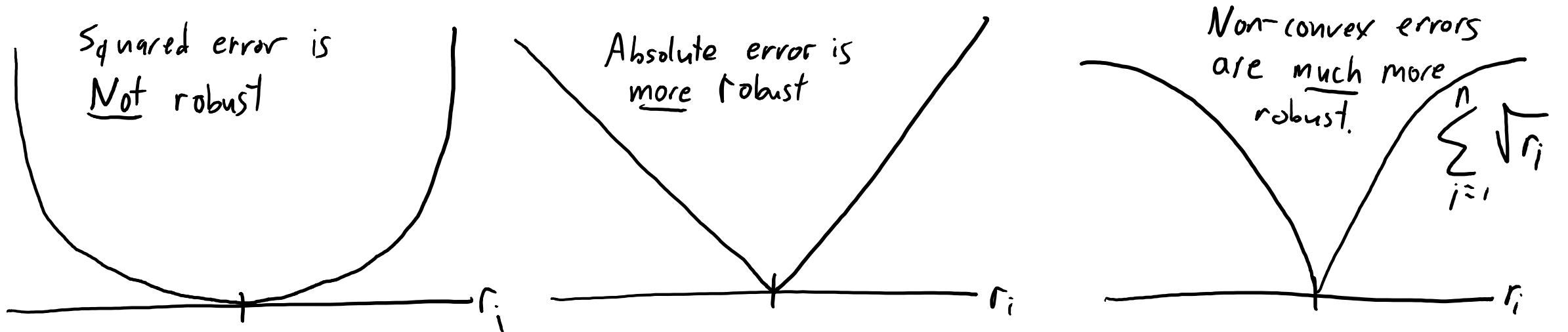
Very Robust Regression



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

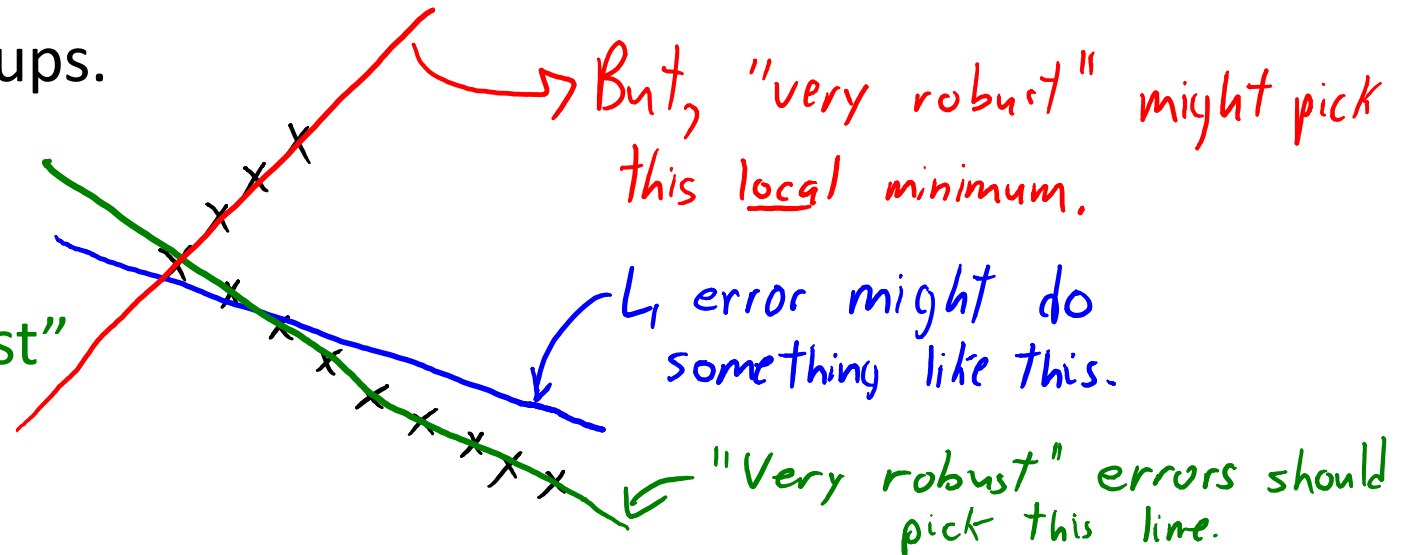


Very Robust Regression



- **Non-convex** errors can be **very robust**:

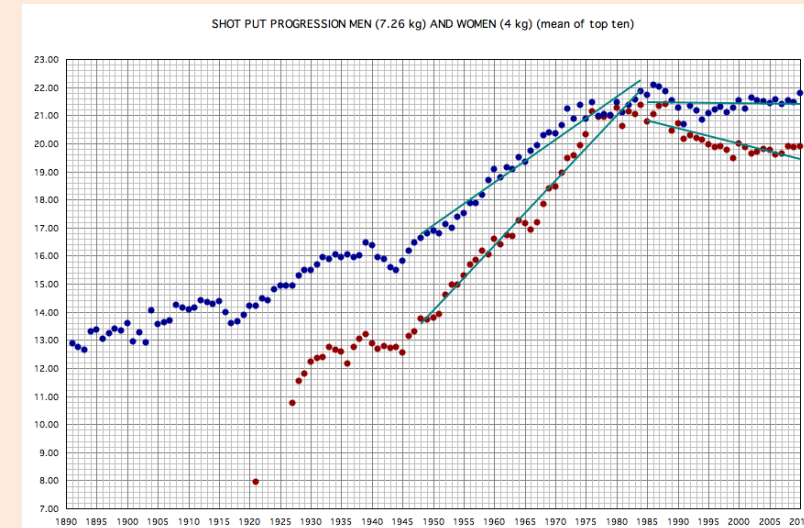
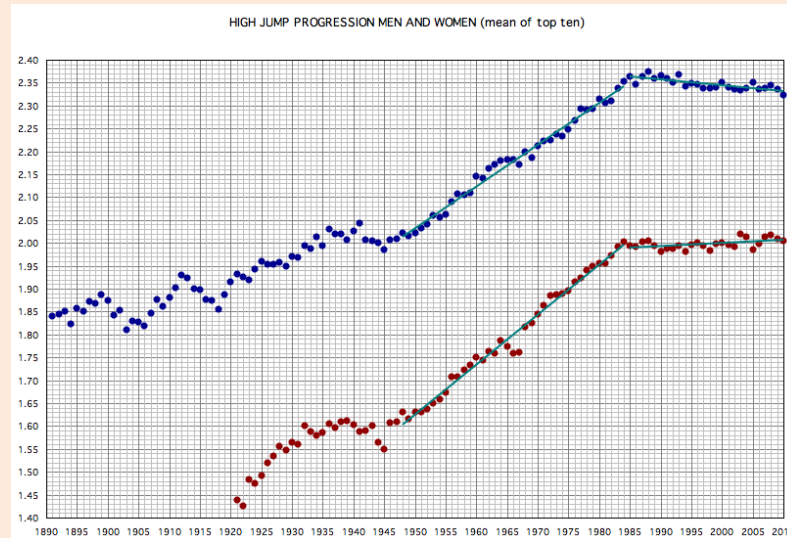
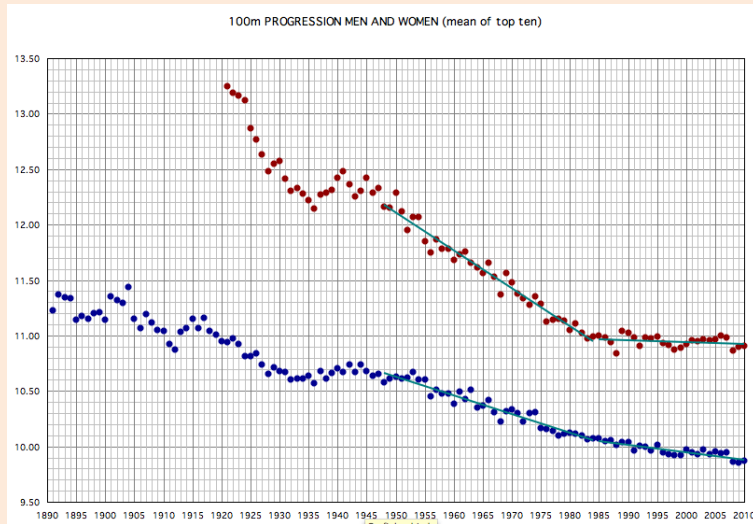
- Not influenced by outlier groups.
- But **non-convex**, so finding **global minimum** is hard.
- **Absolute value** is "most robust" convex loss function.



(pause)

Motivation: Non-Linear Progressions in Athletics

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

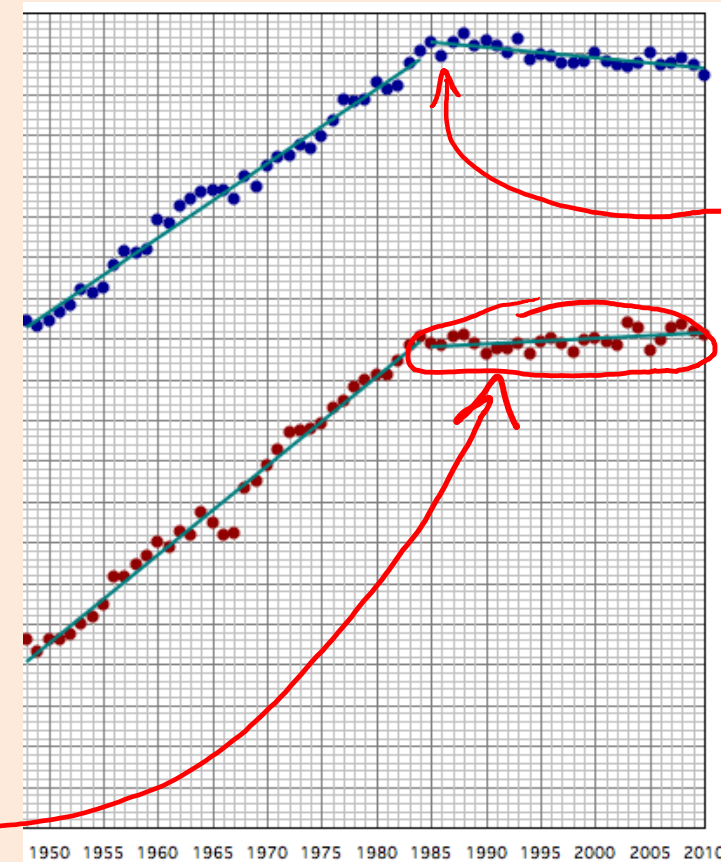
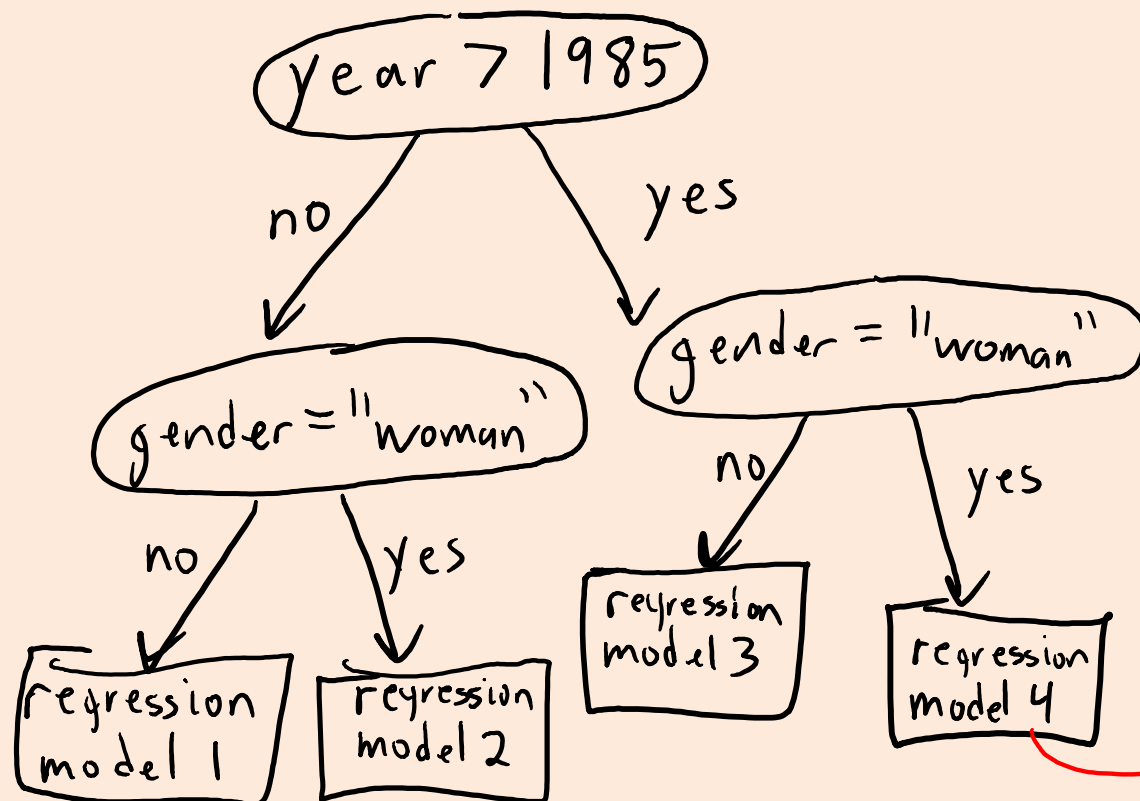


Adapting Counting/Distance-Based Methods

- We can adapt our classification methods to perform regression:

Adapting Counting/Distance-Based Methods

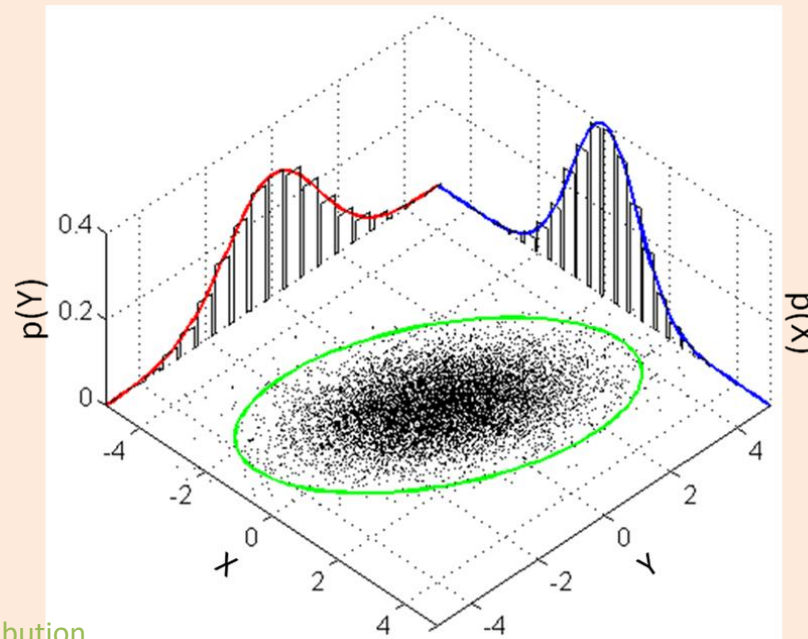
- We can **adapt our classification methods to perform regression**:
 - **Regression tree**: tree with mean value or **linear regression** at leaves.



Not necessarily continuous.

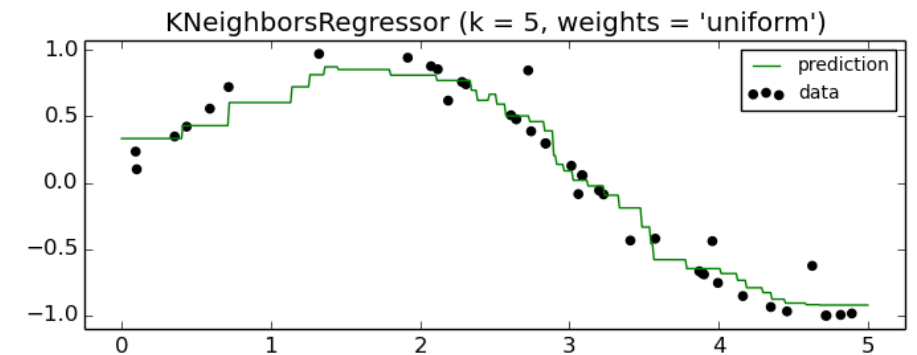
Adapting Counting/Distance-Based Methods

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



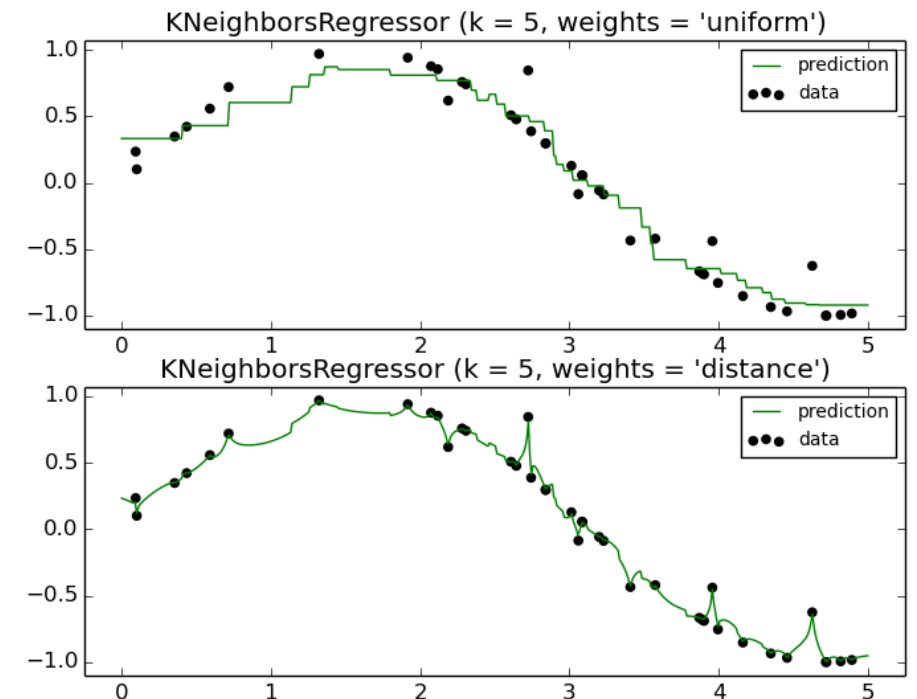
Adapting Counting/Distance-Based Methods

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



Adapting Counting/Distance-Based Methods

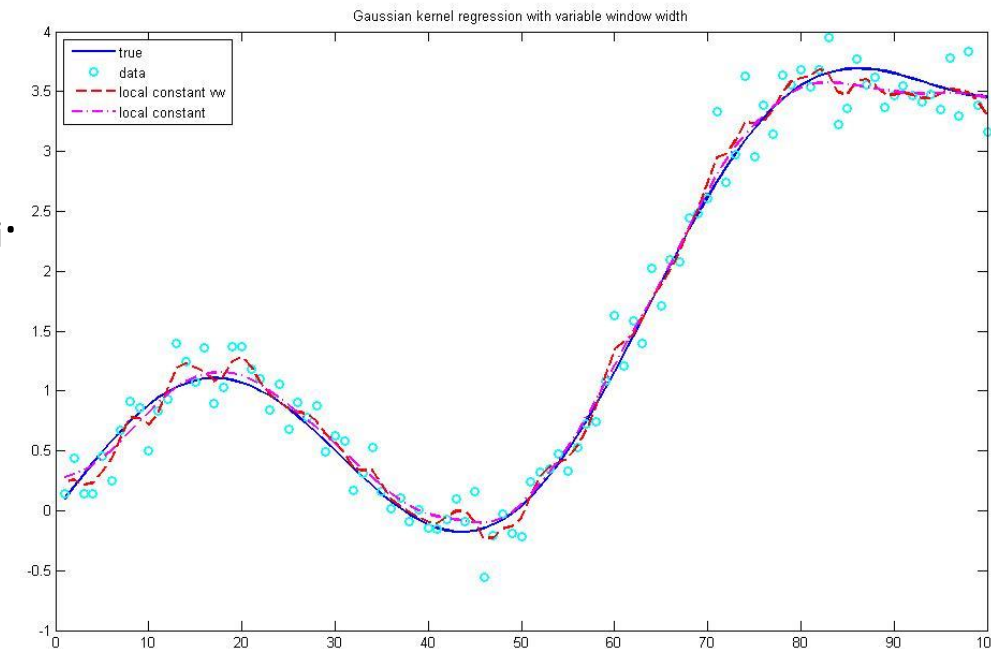
- We can **adapt our classification methods to perform regression**:
 - Regression tree: tree with mean value or linear regression at leaves.
 - **Probabilistic** models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Non-parametric models:
 - KNN regression.
 - Could be **weighted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



Adapting Counting/Distance-Based Methods

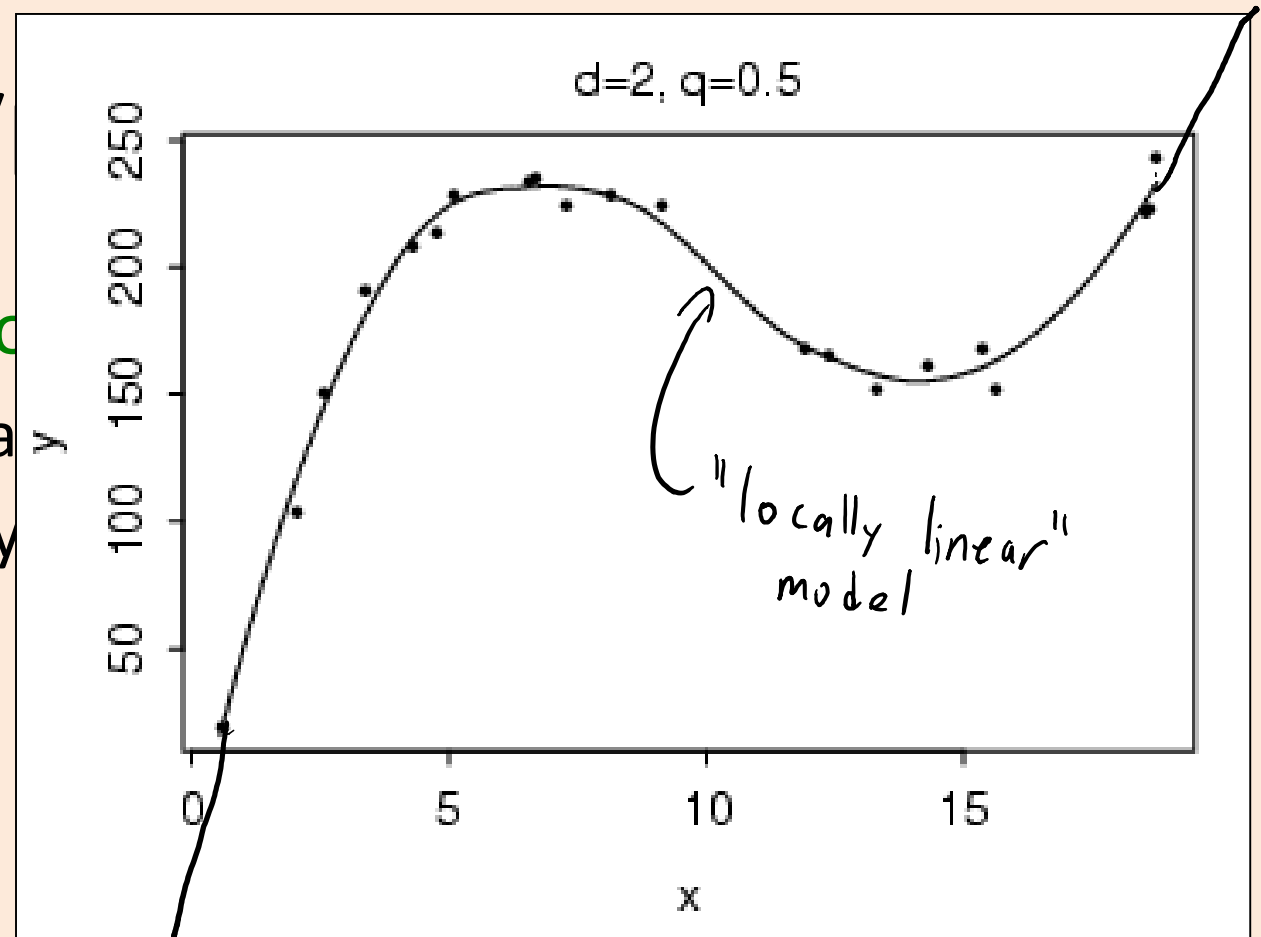
- We can **adapt our classification methods to perform regression**:
 - Regression tree: tree with mean value or linear regression at leaves.
 - **Probabilistic** models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Non-parametric models:
 - KNN regression.
 - Could be weighted by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

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



Adapting Counting/

- We can **adapt our classification**
 - Regression tree: tree with mean
 - **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.)

Adapting Counting/Distance-Based Methods

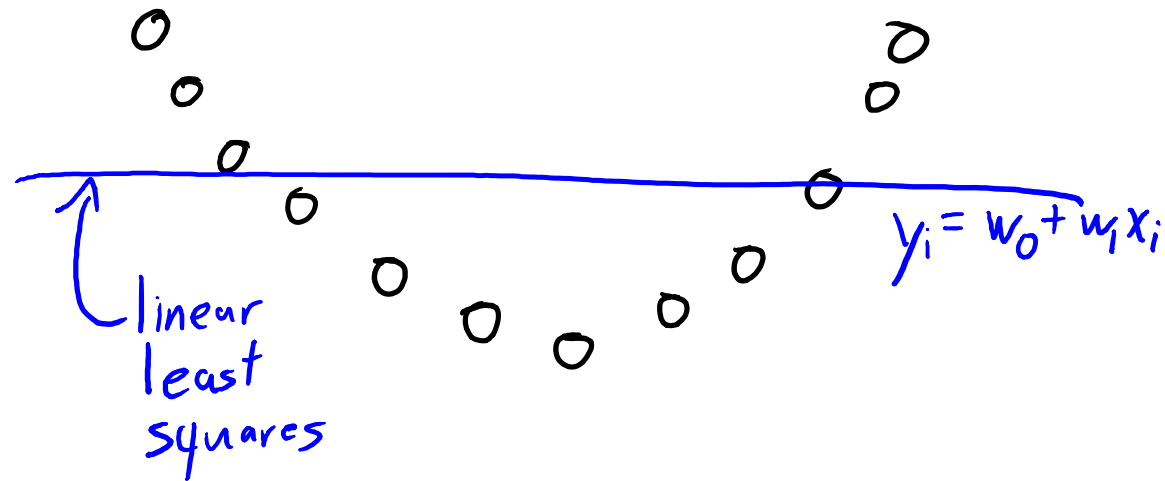
- We can adapt our classification methods to perform regression:
 - Regression tree: tree with mean value or linear regression at leaves.
 - Probabilistic models: fit $p(x_i | y_i)$ and $p(y_i)$ with Gaussian or other model.
 - Non-parametric models:
 - KNN regression.
 - Could be weighted by distance.
 - ‘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.

Motivation: Limitations of Linear Models

- On many datasets, y_i is not a 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**?

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

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

$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix}$$

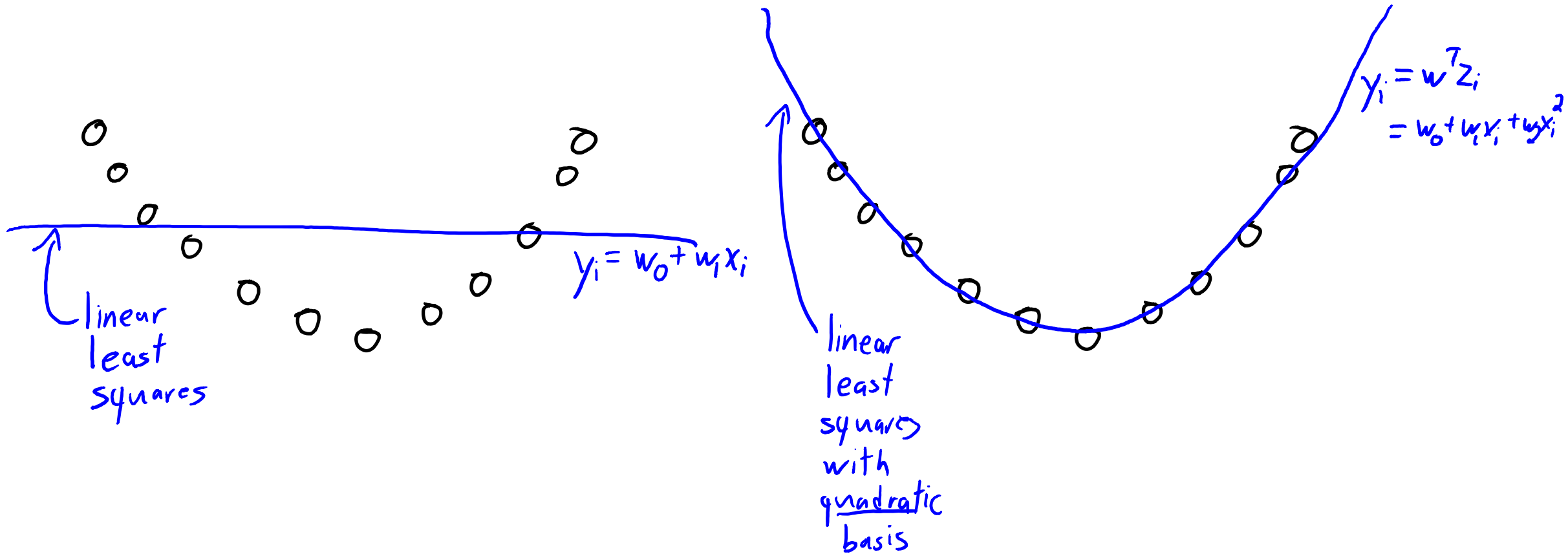
$y\text{-int}$ x x^2

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

- It's a **linear function of w** , but a **quadratic function of x_i** .
- Fit using **normal equations with Z instead of X** : $w = (Z^T Z)^{-1} (Z^T y)$

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

Non-Linear Feature Transforms



General Polynomial Features (d=1)

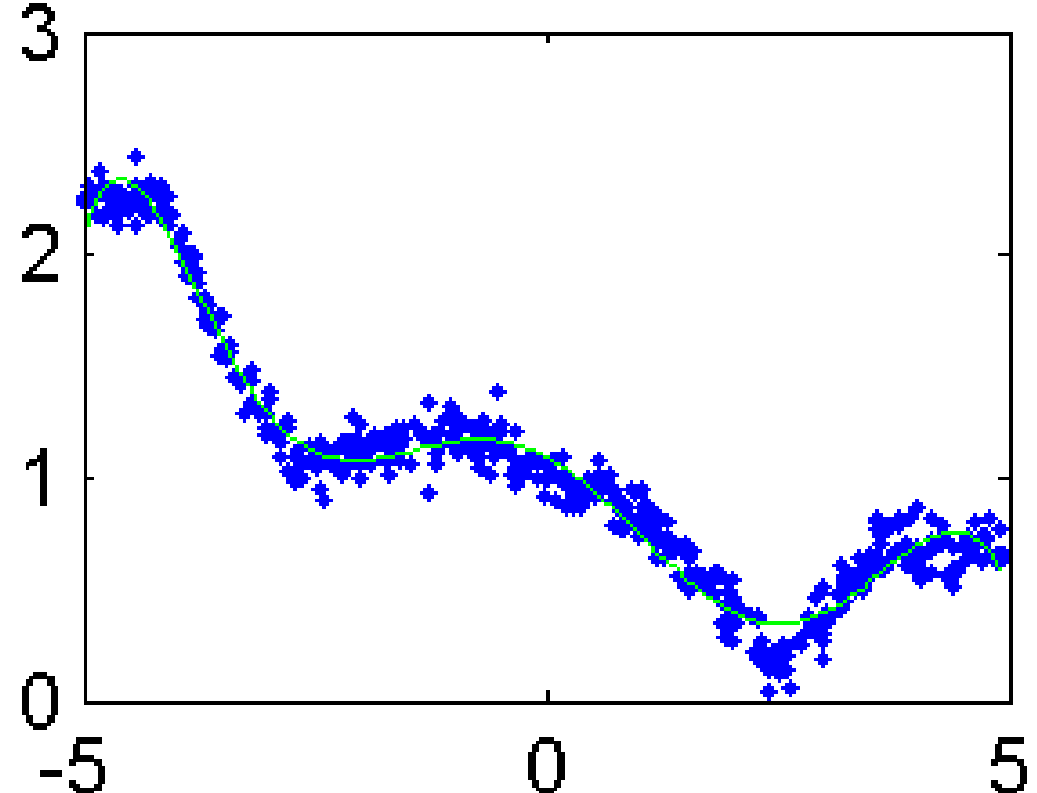
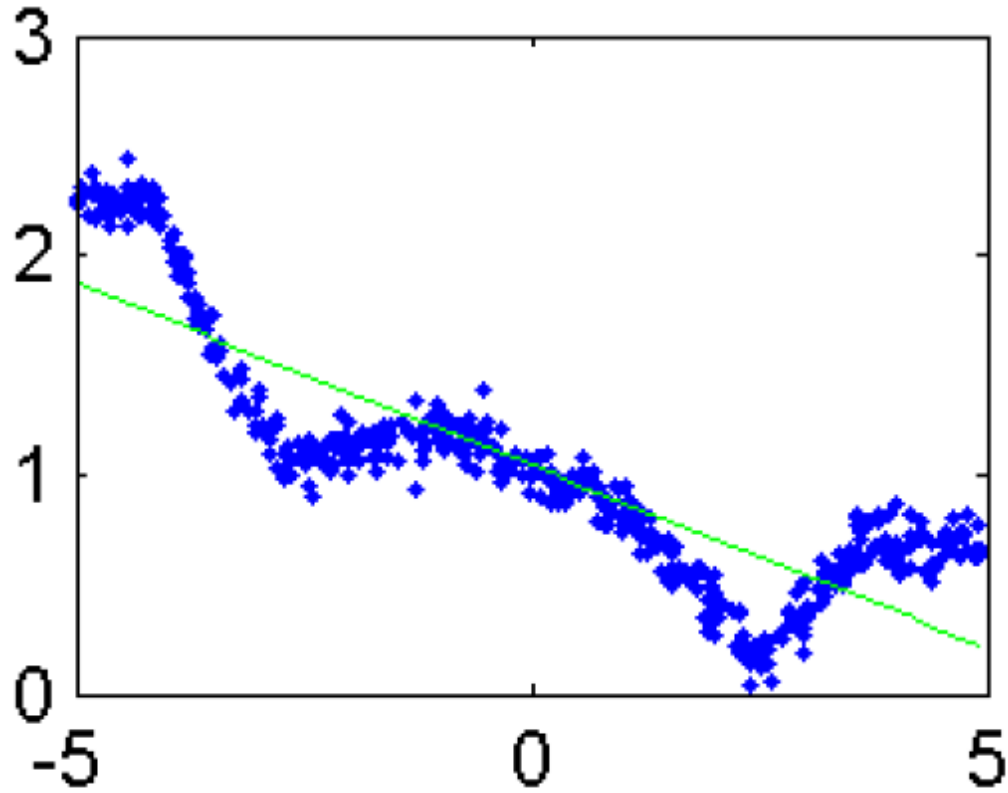
- We can have a **polynomial of degree 'p'** by using these features:

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

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

General Polynomial Features

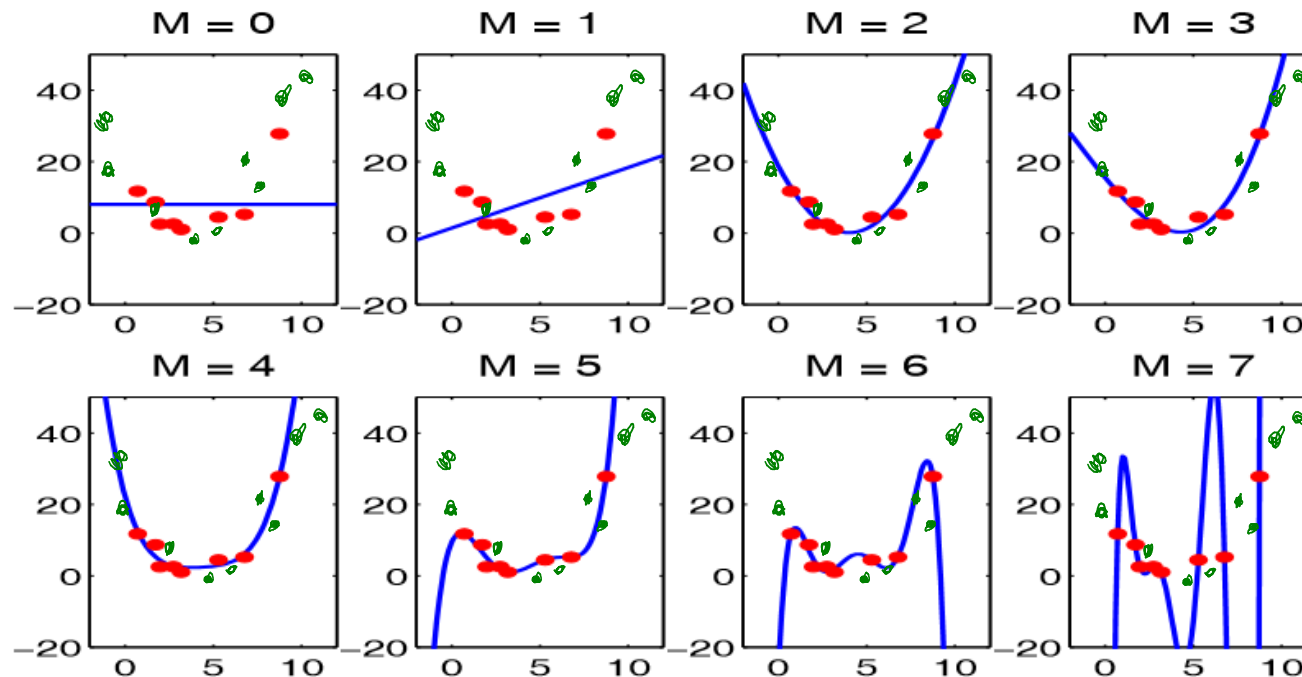
Degree 7



- 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

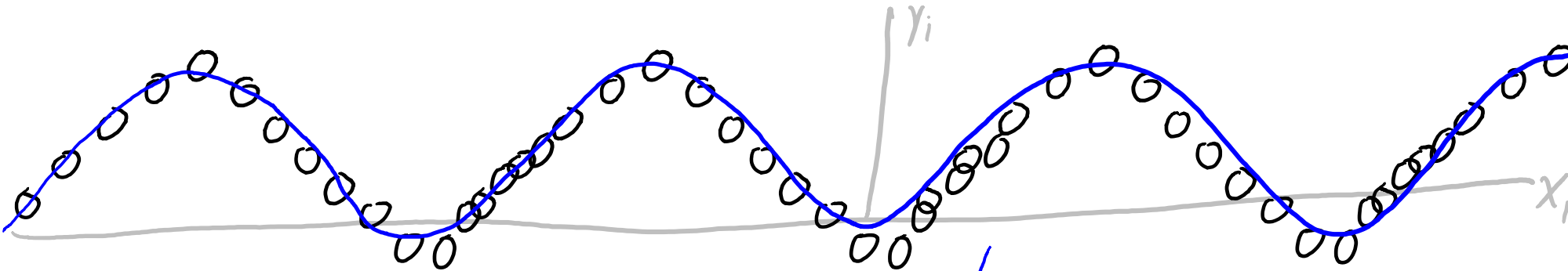
- As the polynomial degree increases, the **training error goes down**.



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

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

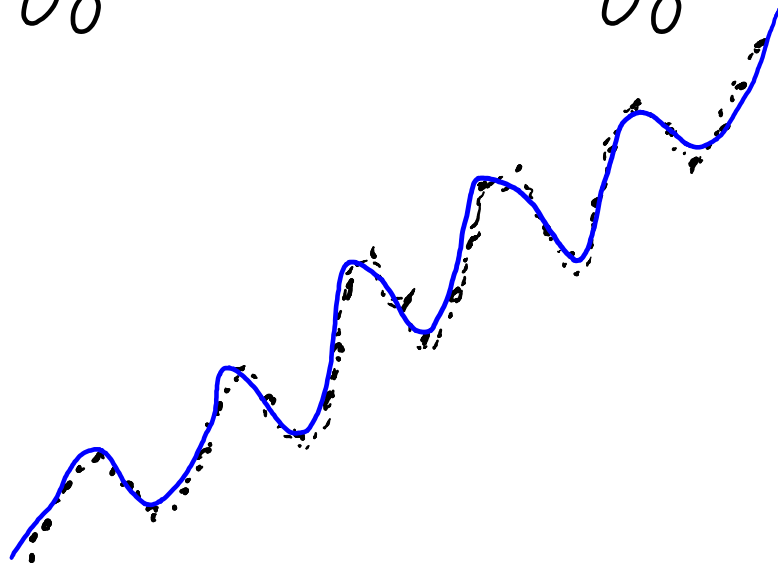
$$Z = \begin{bmatrix} \sin(x_1) \\ \sin(x_2) \\ \vdots \\ \sin(x_n) \end{bmatrix}$$

You can have different types
of bases

$$Z = \begin{bmatrix} x_1 & \sin(6x_1) \\ x_2 & \sin(6x_2) \\ \vdots & \vdots \\ x_n & \sin(6x_n) \end{bmatrix}$$

linear periodic

$$\begin{aligned} y_i &= w^T z_i \\ &= w_1 \sin(x_i) \end{aligned}$$



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.

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

Find 'w' and 'p' minimizing:

$$\text{score}(w, p) = \underbrace{\frac{1}{2} \|Z_p w - y\|^2}_{\text{usual error}} + \underbrace{p}_{\text{degree of polynomial.}}$$

- 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?