Lecture 2: Regression, regularization and the kernel trick

Bingqing Cheng

Trinity College, the University of Cambridge

bc509@cam.ac.uk

Motivation

Why we will begin our journey from linear regression?

- One of the most widely used techniques
- Fundamental to many larger models
- Easy to interpret
- Efficient to solve
- Build understanding about what “data science” is

Have you had lunch?

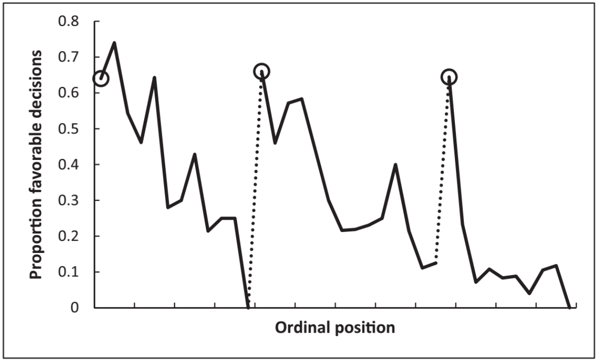


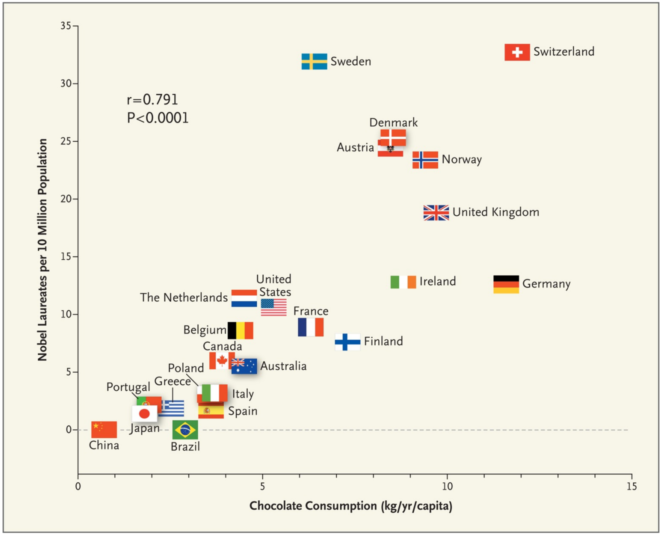
Fig. 1. Proportion of rulings in favor of the prisoners by ordinal position. Circled points indicate the first decision in each of the three decision sessions; tick marks on *x* axis denote every third case; dotted line denotes food break. Because unequal session lengths resulted in a low number of cases for some of the later ordinal positions, the graph is based on the first 95% of the data from each session.

"Extraneous Factors in Judicial Decisions" PNAS 2011

Outline

- Linear models, linear regression
- Polynomial regression
- Regularization
- Kernel regression
- A tale about model selection: Ptolemy Vs. Copernicus

Chocolate consumption & Nobel prize



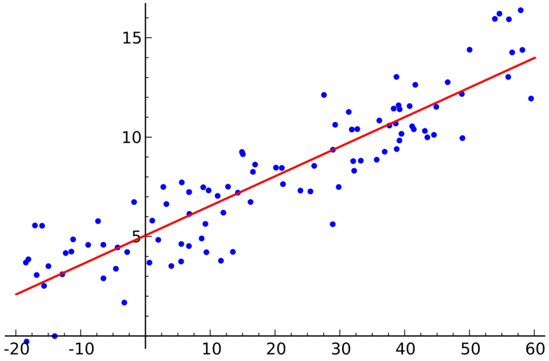
“Chocolate Consumption, Cognitive Function, and Nobel Laureates” N Engl J Med 2012

Chocolate consumption & Nobel prize

- Study Limitations: The present data are based on country averages, and the specific chocolate intake of individual Nobel laureates of the past and present remains unknown. **The cumulative dose of chocolate that is needed to sufficiently increase the odds of being asked to travel to Stockholm is uncertain.**
- Conclusions: Chocolate consumption enhances cognitive function, which is a sine qua non for winning the Nobel Prize, and it closely correlates with the number of Nobel laureates in each country. **It remains to be determined whether the consumption of chocolate is the underlying mechanism for the observed association with improved cognitive function.**

"Chocolate Consumption, Cognitive Function, and Nobel Laureates" N Engl J Med 2012

Simple linear regression



$$y = w_0 + w_1 x + \epsilon$$

Design matrix

- One sample in high dimension:

$$\mathbf{x}^d = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(d)} \end{bmatrix}$$

- N samples (design matrix $\mathbf{X}$):

$$\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)^T = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

- Output $\mathbf{y}$ of the N samples as a column vector $\mathbf{y} \equiv (y_1, \cdots, y_N)^T$

Linear regression

- A simple linear regression model in component form:

$$y = w_0 + \sum_{j=1,2,\ldots,d} w_j x^{(j)} + \epsilon$$

- In matrix form:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$$

Regression coefficients $\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_d \end{bmatrix}^T$, noise $\epsilon \approx N(0, \sigma^2)$

- $$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix}$$

Estimating the Model

Given data how can we estimate the coefficients?

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$$

- Bayesian linear regression
- Least squares (maximum likelihood)
  minimize the data dependent error $E_D(\mathbf{w})$ in the square loss form:

$$E_D(\mathbf{w}) = \frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||^2$$

This means we are trying to find $\mathbf{w}$ satisfy

$$\min_{\mathbf{w} \in \mathcal{R}^d} ||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 = \min_{\mathbf{w} \in \mathcal{R}^d} (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Estimating the Model

- We are trying to find $\mathbf{w}$ satisfy

$$\min_{\mathbf{w}\in\mathcal{R}^d} ||\mathbf{X}\mathbf{w}-\mathbf{y}||^2 = \min_{\mathbf{w}\in\mathcal{R}^d} (\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})$$

- If $\text{rank}(\mathbf{X}) = d$, then there exists unique solution to this problem:

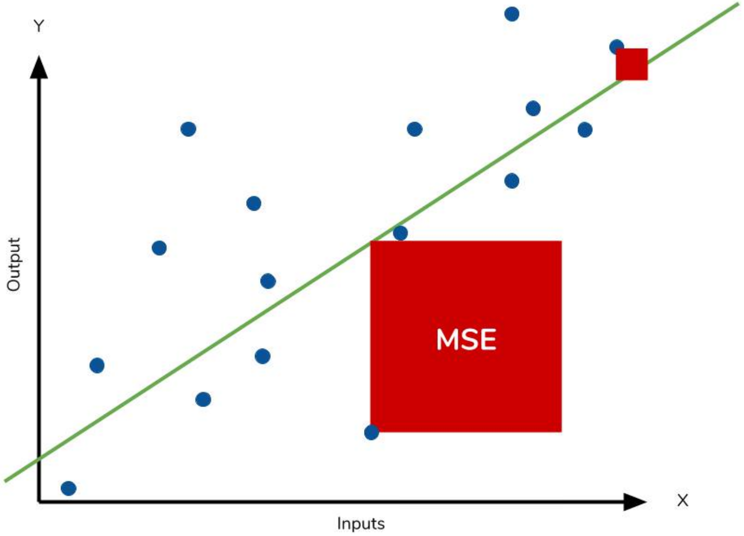
$$\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$$

- If not, one can use psudoinverse

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

- Or one can do a numerical minimization (e.g. gradient decent) to find $\mathbf{w}$.

Pictorial Interpretation of the squared Error



Linear regression in practice

- Matrix inversion using Scipy Python library
-

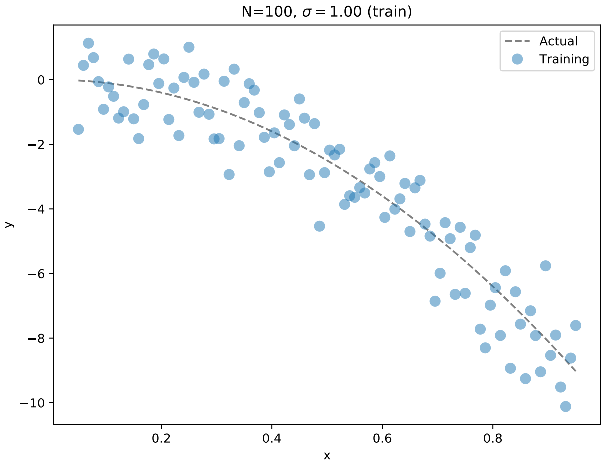
```
from sklearn import linear_model

# Linear Regression : create linear regression object
clf = linear_model.LinearRegression()

# Train the model using the training set
# Note: sklearn requires a design matrix of shape (N_train, N_features)
clf.fit(x, y)
```

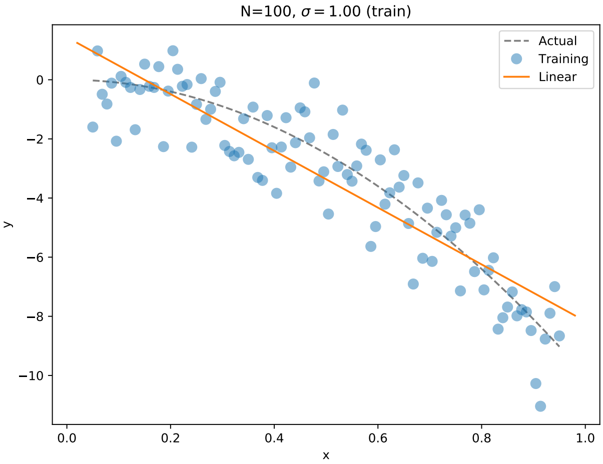
Linear regression in practice

See the Jupyter Notebook provided for the workshop



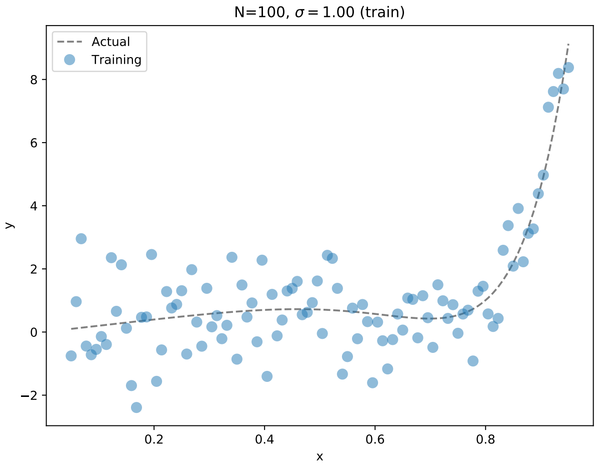
Linear regression in practice

See the Jupyter Notebook provided for the workshop



Fitting non-linear data

What if the output has a non-linear response with respect to the input?
Can we still fit a linear model to it?



Transforming the feature Space

- Design matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(d)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(d)} \\ \vdots & \vdots & \vdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \dots & x_N^{(d)} \end{bmatrix}$$

- Take a feature vector and apply non-linear transformation ϕ:

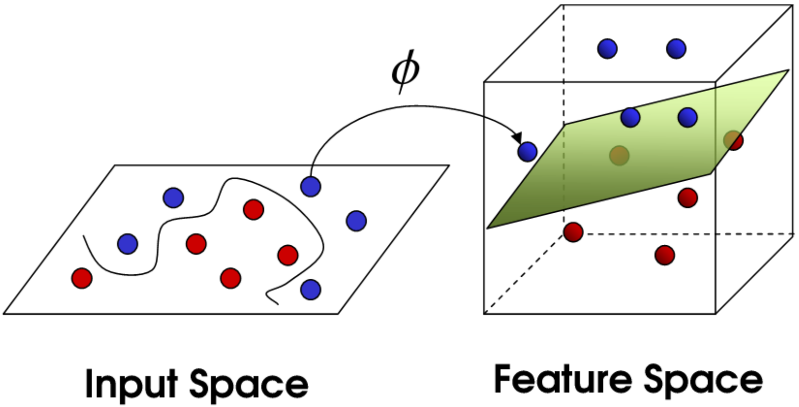
$$\phi : \mathcal{R}^d \to \mathcal{R}^k$$

- Polynomial basis:

$$\phi(x) = \begin{bmatrix} 1 & x^1 & x^2 & x^3 & x^4 \end{bmatrix}$$

- Others: splines, radial basis functions, ...

Transforming the feature Space



Polynomial regression

- Polynomial basis:

$$\phi(x) = \begin{bmatrix} 1 & x^1 & x^2 & x^3 & x^4 \end{bmatrix}$$

- Effectively,the model looks like(if we expand to the 4th order of x):

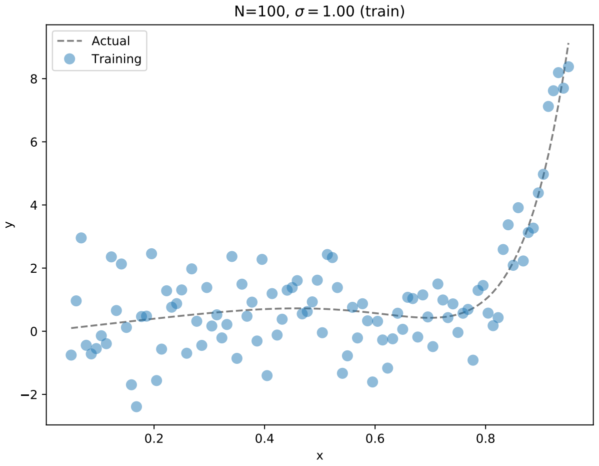
$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + \epsilon$$

- An important observation is that this is still a linear model in ϕ:

$$\phi(x) = \begin{bmatrix} \phi^{(0)} & \phi^{(1)} & \phi^{(2)} & \phi^{(3)} & \phi^{(4)} \end{bmatrix} = \begin{bmatrix} 1 & x^1 & x^2 & x^3 & x^4 \end{bmatrix}$$

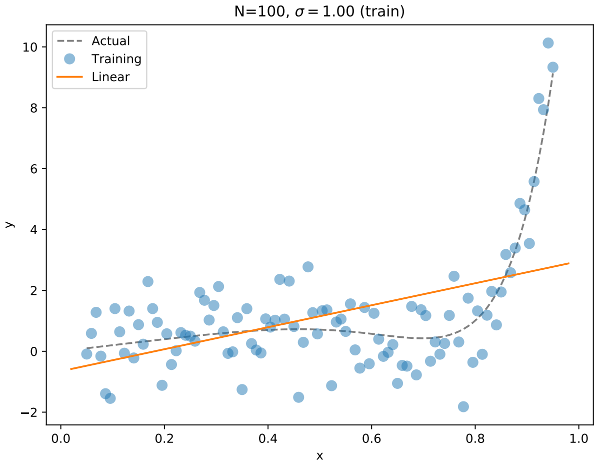
Polynomial regression at work

See the Jupyter Notebook provided for the workshop



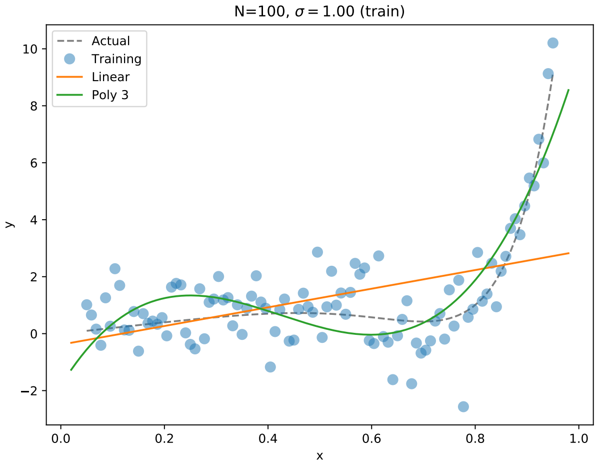
Polynomial regression at work

See the Jupyter Notebook provided for the workshop



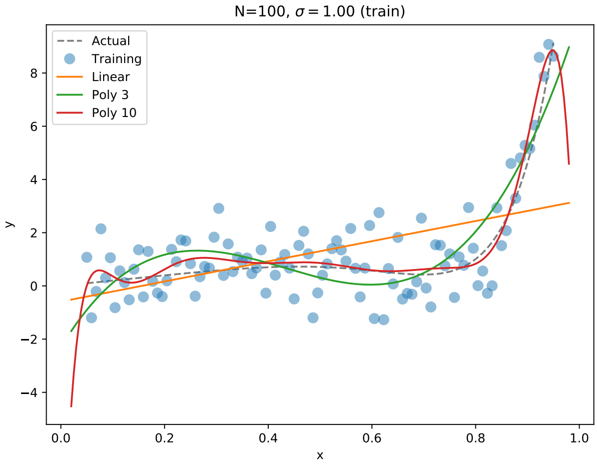
Polynomial regression at work

See the Jupyter Notebook provided for the workshop



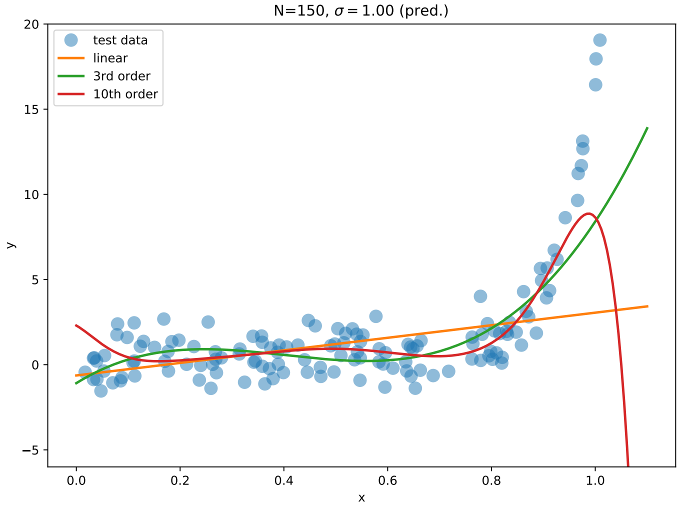
Polynomial regression at work

See the Jupyter Notebook provided for the workshop



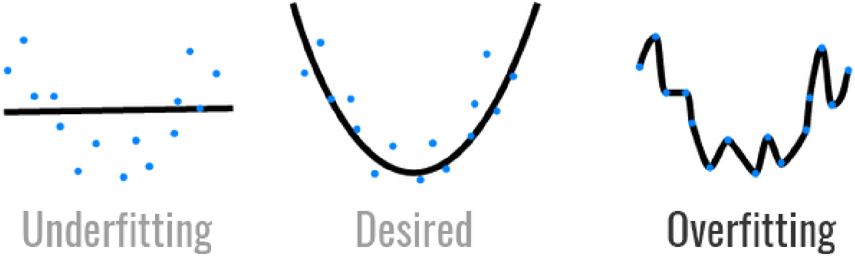
Polynomial regression at work

See the Jupyter Notebook provided for the workshop



Overfitting

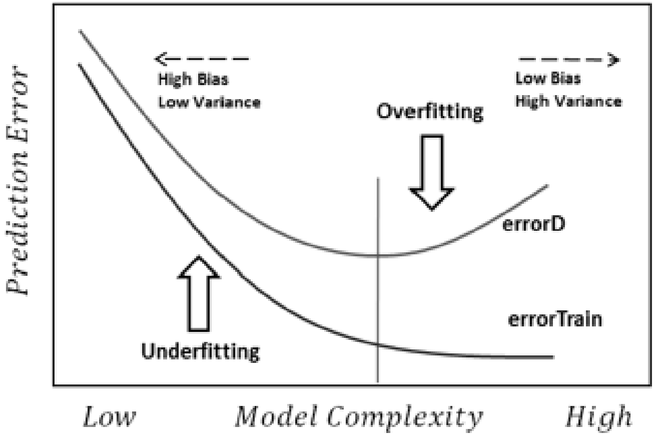
- Errors on training data are small
- But errors on new points are likely to be large



Trade-off between fitting error and model complexity

Occam's razor:

"Entities should not be multiplied without necessity."

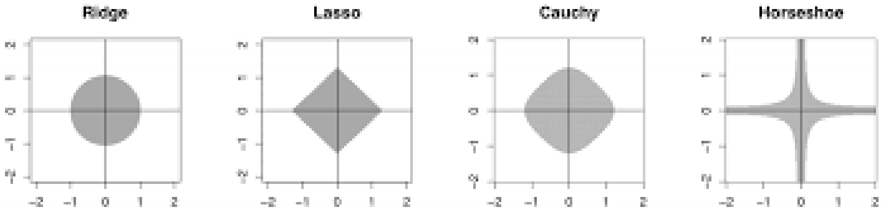


Regularization

- Ridge Regression (L2)

  $E_{ridge}(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 = \frac{\lambda}{2}\sum_{j=1}^{d} w_j w_j$
- LASSO (L1)

  $E_{LASSO}(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}|| = \frac{\lambda}{2}\sum_{j=1}^{d} |w_j|$



Regularization

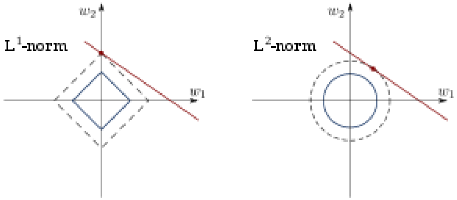
- Ridge Regression (L2)

$$\mathbf{w}_{ridge}(\lambda) = \underset{\mathbf{w}\in\mathcal{R}^d}{\operatorname{argmin}} \frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

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

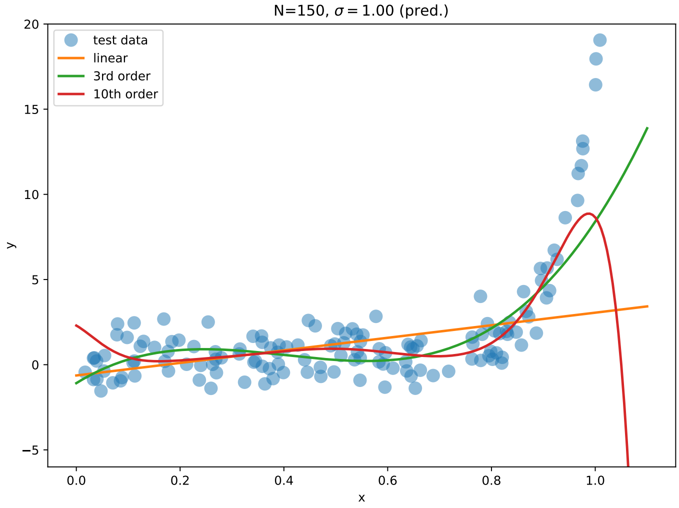
- LASSO (L1)

$$\mathbf{w}_{LASSO}(\lambda) = \underset{\mathbf{w}\in\mathcal{R}^d}{\operatorname{argmin}} \frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 + \frac{\lambda}{2}||\mathbf{w}||$$



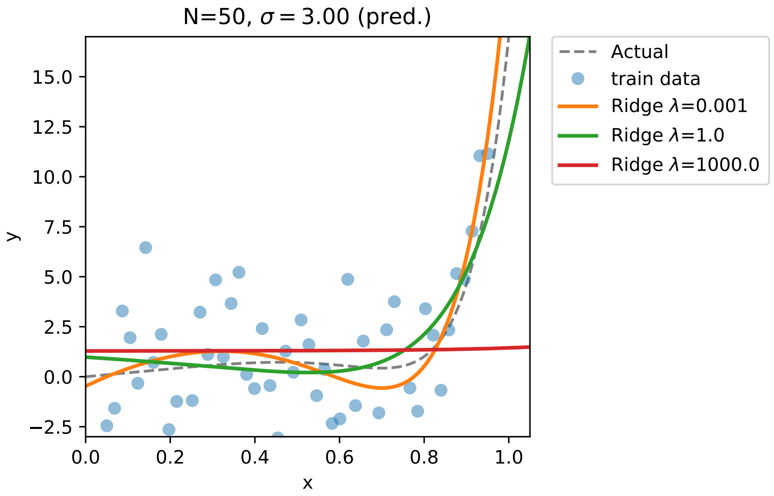
Ridge regression at work

See the Jupyter Notebook provided for the workshop



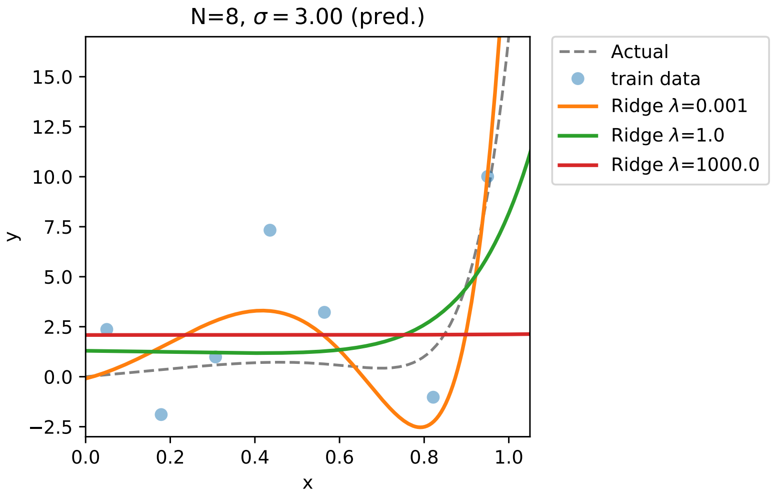
Ridge regression at work

See the Jupyter Notebook provided for the workshop



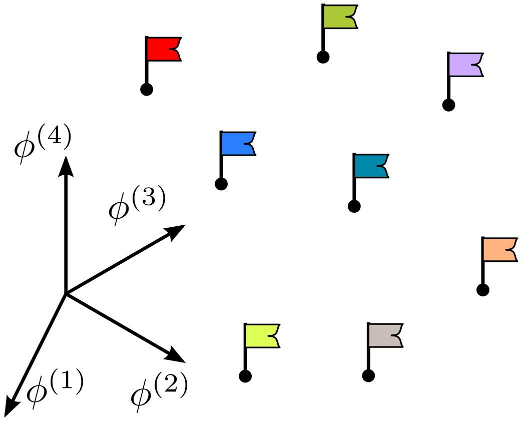
Ridge regression at work

See the Jupyter Notebook provided for the workshop



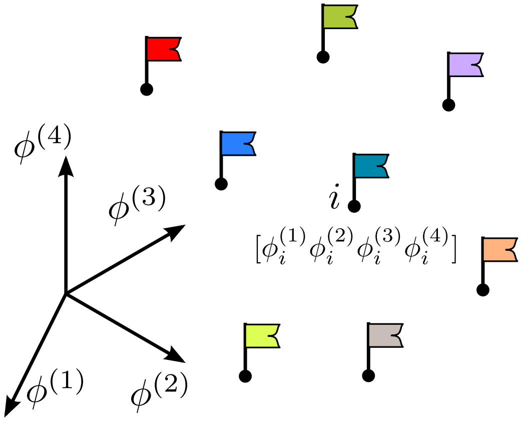
Kernel regression

- Derivation of the kernel trick in the lecture notes!
- We don't need to compute or have basis functions, just the similarities between each pair of data k_{ij} is enough.



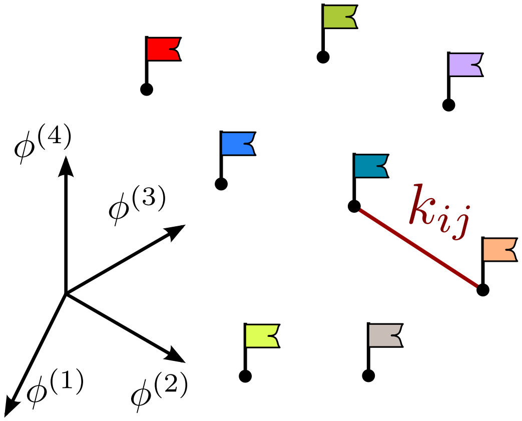
Kernel regression

- Derivation of the kernel trick in the lecture notes!
- We don't need to compute or have basis functions, just the similarities between each pair of data k_{ij} is enough.



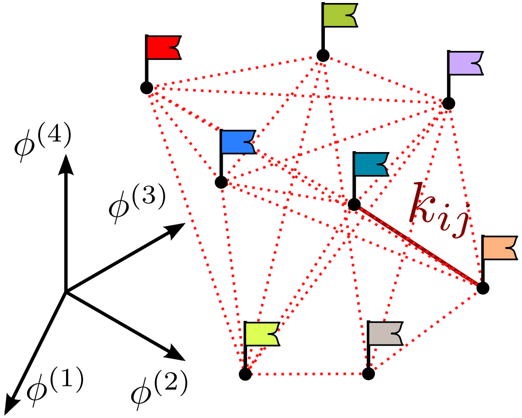
Kernel regression

- Derivation of the kernel trick in the lecture notes!
- We don't need to compute or have basis functions, just the similarities between each pair of data k_{ij} is enough.



Kernel regression

- Derivation of the kernel trick in the lecture notes!
- We don't need to compute or have basis functions, just the similarities between each pair of data k_{ij} is enough.



Ptolemy Vs. Copernicus

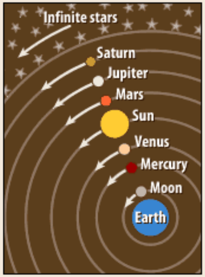
GET TO THE ROOT OF IT

Views of the universe: Ptolemy vs. Copernicus

Ptolemy's model: "Earth-centered," or "geocentric"

Ptolemy thought that all celestial objects — including the planets, Sun, Moon, and stars — orbited Earth. Earth, in the center of the universe, did not move at all.

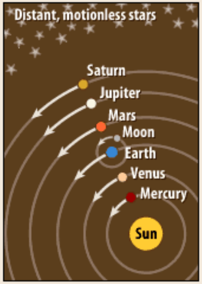
NOTE: The outer planets, like Uranus and Neptune, are missing from both charts because they had not been discovered at the time. The planets are lined up to make the charts easy to read; they never line up this way in nature.



Copernicus' model: "Sun-centered," or "heliocentric"

Copernicus thought that the planets orbited the Sun, and that the Moon orbited Earth. The Sun, in the center of the universe, did not move, nor did the stars.

Copernicus was correct about some things, but wrong about others. The Sun is not in the center of the universe, and it does move, as do the stars. Also, both Copernicus and Ptolemy thought the orbits of the planets were circular, but we now know they are elliptical.

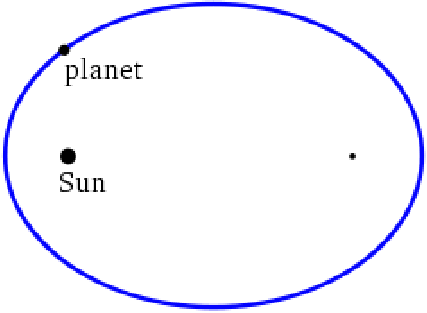


https://amazing-space.stsci.edu/resources/explorations/groundup/lesson/basic

The shortcomings of the Copernicus' model

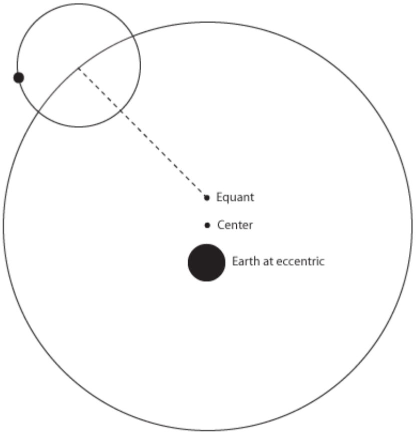
Inaccurate description of the planet motion:

- The planet orbits are elliptical

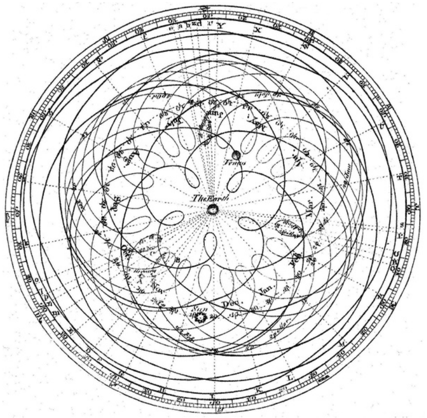


- General relativity and perturbations from other planets cause fluctuations

Ptolemy's Universe and Homer



Ptolemy's Universe and Homer



Ptolemy's Universe and Homer

One epicycle the equation of motion follows:

$$z(t) = Re^{i\omega t}$$

Two epicycles:

$$z(t) = R_1 e^{i\omega_1 t} + R_2 e^{i\omega_2 t}$$

Infinite epicycles:

$$z(t) = \int_{-\infty}^{\infty} R(\omega) e^{i\omega t} d\omega$$

Fourier Transform

https://www.youtube.com/watch?v=QVuU2YCwHjw

Ptolemy's Universe and Homer

One epicycle the equation of motion follows:

$$z(t) = Re^{i\omega t}$$

Two epicycles:

$$z(t) = R_1 e^{i\omega_1 t} + R_2 e^{i\omega_2 t}$$

Infinite epicycles:

$$z(t) = \int_{-\infty}^{\infty} R(\omega) e^{i\omega t} d\omega$$

Fourier Transform

Ptolemy's planet model is the winner from the data science point of view!

Outline

- Linear models, linear regression
- Polynomial regression
- Regularization
- Kernel regression
- A tale about model selection: Ptolemy Vs. Copernicus

Questions?

Thank you for your attention!