

CS-UY 4563: Lecture 4

Finish Linear Regression, Model Selection

NYU Tandon School of Engineering, Prof. Christopher Musco

- First written assignment due **Thursday**, by midnight.
- Second lab posted `lab_robot_partial.ipynb` due next Tuesday 2/11, by midnight.

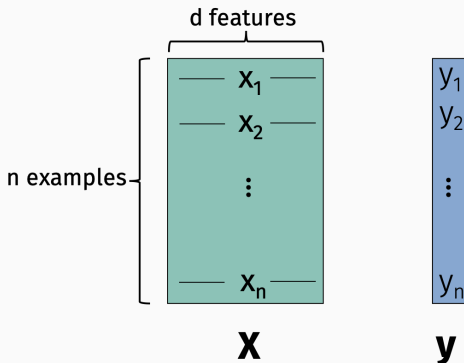
MULTIPLE PREDICTOR DATA SET

Target variable:

- Scalars y_1, \dots, y_n for n data examples (a.k.a. samples).

Predictor variables:

- d dimensional vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ for n data examples and d features



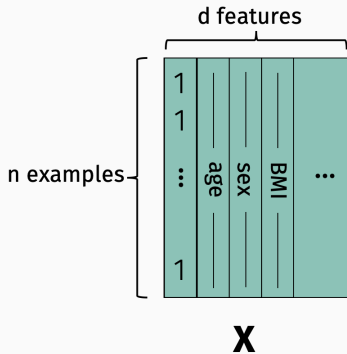
Motivating example: Predict diabetes progression in patients after 1 year based on health metrics. (Measured via numerical score.)

Features: Age, sex, body mass index, average blood pressure, six blood serum measurements (e.g. cholesterol, lipid levels, iron, etc.)

Demo in `demo1_diabetes.ipynb`.

THE DATA MATRIX

Predictor variables:



Linear Least-Squares Regression.

- Model:

$$f_{\beta}(\mathbf{x}) = \langle \mathbf{x}, \beta \rangle$$

- Model Parameters:

$$\beta = [\beta_1, \beta_2, \dots, \beta_d]$$

- Loss Function:

$$L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Machine learning goal: minimize the loss function

$$L(\boldsymbol{\beta}) : \mathbb{R}^d \rightarrow \mathbb{R}.$$

Find optimum by determining for which $\boldsymbol{\beta} = [\beta_1, \dots, \beta_d]$ the gradient is 0. I.e. when do we have:

$$\nabla L(\boldsymbol{\beta}) = \begin{bmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Function:

$$f(\mathbf{z}) = \mathbf{a}^T \mathbf{z} \text{ for some fixed column vector } \mathbf{a} \in \mathbb{R}^d$$

Gradient:

Function:

$$f(\mathbf{z}) = \|\mathbf{z}\|_2^2$$

Gradient:

Function:

$$f(\mathbf{z}) = g(\mathbf{Az}) = \text{ for fixed } \mathbf{A} \in \mathbb{R}^{n \times d} \text{ and function } g$$

Gradient:

Loss function:

$$L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Loss function: $\|y - X\beta\|_2^2$.

Goal: minimize the loss function $L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$.

$$\nabla L(\beta) = 2\mathbf{X}^T\mathbf{X}\beta - 2\mathbf{X}^T\mathbf{y} = 0$$

Solve for optimal β^* :

$$\mathbf{X}^T\mathbf{X}\beta^* = \mathbf{X}^T\mathbf{y}$$

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

What is the sign of β_1 when we run a simple linear regression using the following predictors for diabetes progression in isolation:

- Body mass index (BMI): **Positive**
- Sex (values of 1 indicates male, value of 2 indicates female): **Positive**

What is the sign of the corresponding β 's when we run a multiple linear regression using the following predictors together:

- Body mass index (BMI): **Positive**
- Sex (values of 1 indicates male, value of 2 indicates female): **Negative**

Can you explain this? Try to think of your own example of a regression problem where this phenomenon might show up.

DEALING WITH CATEGORICAL VARIABLES

The sex variable in the diabetes problem was binary.

Suppose we go back to the MPG prediction problem. What if we had a categorical predictor variable for car make with more than 2 options: e.g. Ford, BMW, Honda. **How would you encode as a numerical column?**

$$\begin{bmatrix} \text{ford} \\ \text{ford} \\ \text{honda} \\ \text{bmw} \\ \text{honda} \\ \text{ford} \end{bmatrix} \rightarrow \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

Better approach: One Hot Encoding.

$$\begin{bmatrix} \text{ford} \\ \text{ford} \\ \text{honda} \\ \text{bmw} \\ \text{honda} \\ \text{ford} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

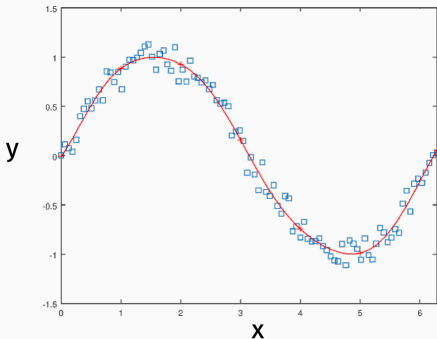
- Create a separate feature for every category, which is 1 when the variable is in that category, zero otherwise.
- Not too hard to do by hand, but you can also use library functions like `sklearn.preprocessing.OneHotEncoder`.

Avoids adding inadvertent linear relationships.

TRANSFORMED LINEAR MODELS

Suppose we have singular variate data examples (x, y) . How could we fit the non-linear model:

$$y \approx \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$



TRANSFORMED LINEAR MODELS

Transform into a multiple linear regression problem:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

Each column j is generated by a different basis function $\phi_j(x)$.

Could have:

- $\phi_j(x) = x^q$
- $\phi_j(x) = \sin(x)$
- $\phi_j(x) = \cos(10x)$
- $\phi_j(x) = 1/x$

Transformations can also be for multivariate data.

Example: Multinomial model.

- Given a dataset with target y and predictors x, z .
- For inputs $(x_1, z_1), \dots, (x_n, z_n)$ construct the data matrix:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & z_1 & z_1^2 & x_1 z_1 \\ 1 & x_2 & x_2^2 & z_2 & z_2^2 & x_2 z_2 \\ \vdots & \vdots & & \vdots & & \\ 1 & x_n & x_n^2 & z_n & z_n^2 & x_n z_n \end{bmatrix}$$

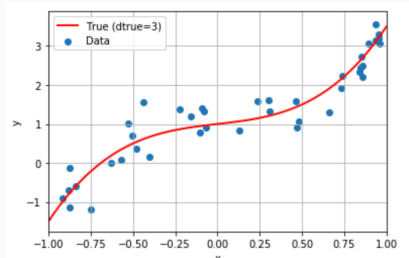
- Captures non-linear interaction between x and y .

Remainder of lecture: Learn about model selection, test/train paradigm, and cross-validation through a simple example.

S

Simple experiment:

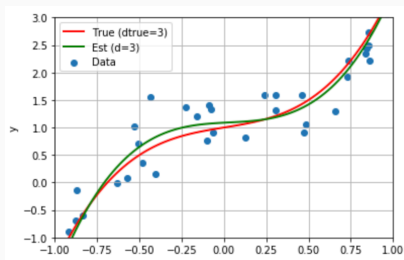
- Randomly select data points $x_1, \dots, x_n \in [-1, 1]$.
- Choose a degree 3 polynomial $p(x)$.
- Create some fake data: $y_i = p(x_i) + \eta$ where η is a random number (e.g random Gaussian).



FITTING A POLYNOMIAL

Simple experiment:

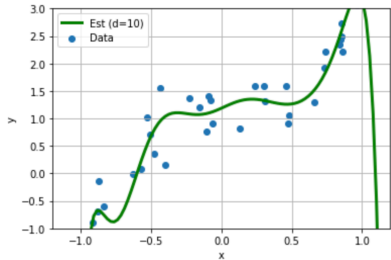
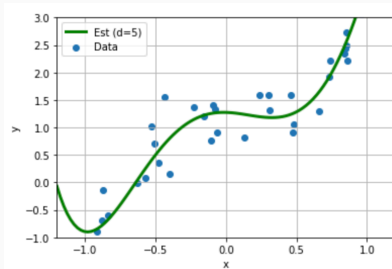
- Use multiple linear regression to fit a degree 3 polynomial.



FITTING A POLYNOMIAL

What if we fit a higher degree polynomial?

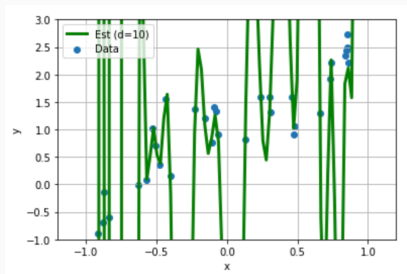
- Fit degree 5 polynomial under squared loss.
- Fit degree 10 polynomial under squared loss.



FITTING A POLYNOMIAL

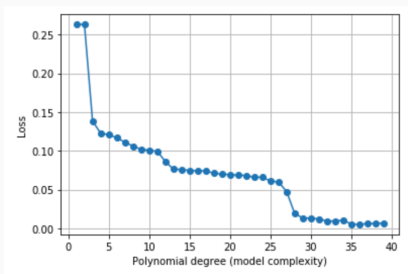
Even higher?

- Fit degree 40 polynomial under squared loss.



MODEL SELECTION

The more **complex** our model class (i.e. the higher degree we allow) the better our loss:



Is our model getting better and better?

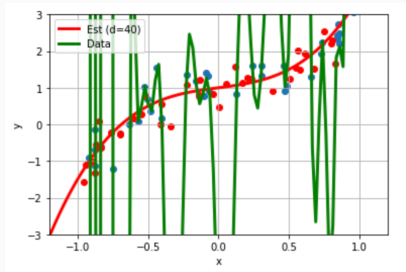
Given the raw data, how do we know which model to choose?

Degree 3? Degree 5? Degree 40?

MODEL SELECTION

Problem: Loss alone is not informative for choosing model.

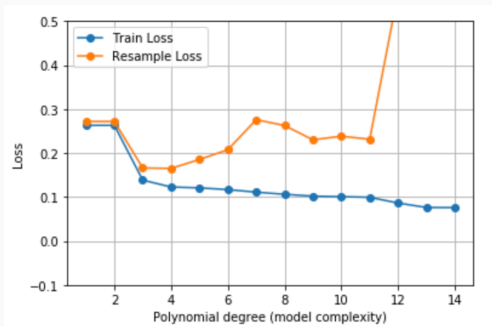
For more complex models, we get smaller loss on the training data, but don't expect to perform well on “new” data:



In other words, the model does not **generalize**.

MODEL SELECTION

Solution: Directly test model on “new data”.



- Loss continues to decrease as model complexity grows.
- Performance on new data “turns around” once our model gets too complex. Minimized around degree 4.

In most situations, we cannot simply collect or generate “new data”. Here’s an alternative:

Test/train split:

- Given data set (\mathbf{X}, \mathbf{y}) , split into two sets $(\mathbf{X}_{tr}, \mathbf{y}_{tr})$ and $(\mathbf{X}_{ts}, \mathbf{y}_{ts})$.
- Train q models f_1, \dots, f_q by finding parameters which minimize the loss on $(\mathbf{X}_{tr}, \mathbf{y}_{tr})$.
- Evaluate loss of each trained model on $(\mathbf{X}_{ts}, \mathbf{y}_{ts})$.

Justification:

- Assume each data example is randomly drawn from some distribution $(\mathbf{x}, y) \sim \mathcal{D}$: we don't care about any particulars of this distribution.
- **Goal:** Find model $f \in \{f_1, \dots, f_q\}$ and parameter vector $\boldsymbol{\theta}$ to minimize the **Risk**:

$$R(f, \boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [L(f(\mathbf{x}, \boldsymbol{\theta}) - y)]$$

where L is some loss function (e.g. $L(z) = |z|$ or $L(z) = z^2$).

Justification:

- Suppose the testing dataset $(\mathbf{X}_{ts}, \mathbf{y}_{ts})$ has m examples.
- Given any model f and parameters θ , let

$$L_{ts}(f, \theta) = \frac{1}{m} \sum_{\mathbf{x}, y \in (\mathbf{X}_{ts}, \mathbf{y}_{ts})} L(f(\mathbf{x}, \theta) - y)$$

- Claim:¹

$$\mathbb{E}[L_{ts}(f, \theta)] = R(f, \theta).$$

- So our testing error is an unbiased estimate for the true risk which measures how well a function performs on average for any “new” data point.

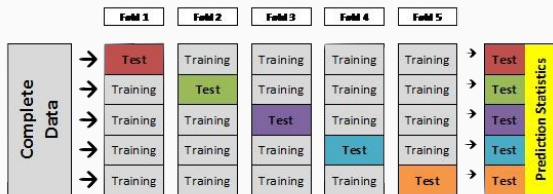
¹Only true if f and θ are chose *without looking at your test data*.

How should we divide up our data into training and test data?

- 50% train, 50% test?
- 80% train, 20% test?
- 20% train, 80% test?

There is a trade-off here. A larger test set allows for a more accurate estimate of the risk R , which helps us choose a model. However, it means we have less data to train with, so for each individual model we might get a worst set of parameters.

K-FOLD CROSS VALIDATION



- Randomly divide data in K parts.
 - Typical choice: $K = 5$ or $K = 10$.
- Use $K - 1$ parts for training, 1 for test.
- For each model, compute test loss L_{ts} for each “fold”.
- Choose model with best average loss.
- Retrain best model on entire dataset.

Leave-one-out cross validation: take $K = n$, where n is our total number of samples.

Is there any disadvantage to choosing K larger?