CS-UY 4563: Lecture 4
Finish Linear Regression, Model Selection

NYU Tandon School of Engineering, Prof. Christopher Musco

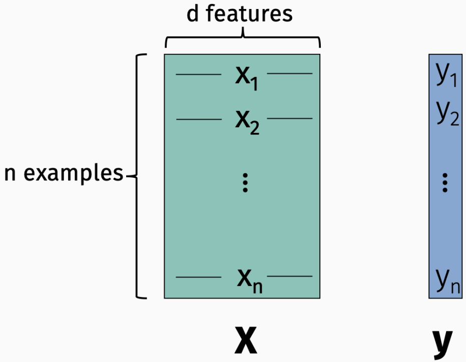
- First written assignment due **Thursday, by midnight.**
- Second lab posted `lab_robot_partial.ipynb` due **next Tuesday 2/11, by midnight.**

Target variable:

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

Predictor variables:

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

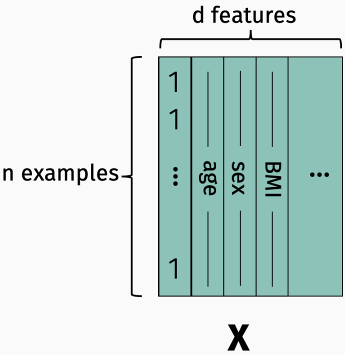


Motivating example: Predict diabetes progression in patients after 1 year based on health metrics. (Measured via numerical score.)

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`.

Predictor variables:



Linear Least-Squares Regression.

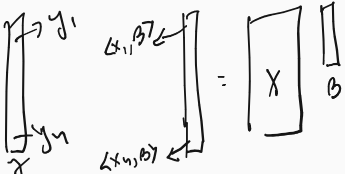
- Model:

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

- Model Parameters:

$$\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_d]$$

- Loss Function:



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

Machine learning goal: minimize the loss function $L(\boldsymbol{\beta}) : \mathbb{R}^d \rightarrow \mathbb{R}$.

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

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

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

Function: $\mathbb{R}^d$ input, $\mathbb{R}$ output

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

$$f(z) = \sum_{i=1}^{d} a_i z_i$$

Gradient: $\nabla f(z) =$

$$\nabla f(z) = \begin{bmatrix} \partial f/\partial z_1 \\ \vdots \\ \partial f/\partial z_d \end{bmatrix} \begin{matrix} \to a_1 \\ \\ \to a_d \end{matrix} = \boxed{\vec{a}} \in \mathbb{R}^d$$

Function:

$$f(\mathbf{z}) = \|\mathbf{z}\|_2^2 = \sum_{i=1}^{d} z_i^2$$

Gradient:

$$\frac{\partial f}{\partial z_i} = 2z_i$$

$$\nabla f(z) = 2z.$$

Function:

∇g

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

$(n \times d)\,(d \times 1)$

$A \quad z$

Gradient:

$\vec{w} = Az \qquad \vec{w} \in \mathbb{R}^n$

Multivariate Chain Rule

$$\frac{\partial}{\partial z_i} g(w) = \sum_{j=1}^{n} \frac{\partial g}{\partial w_j} \cdot \left(\frac{\partial w_j}{\partial z_i}\right) \longrightarrow A_{ji}$$

$$\frac{\partial}{\partial z_i} g(w) = \sum_{j=1}^{n} \frac{\partial g}{\partial w_j} A_{ji}$$

$[z] \to z_i + \delta \longrightarrow [w_1 + \delta_1, \ldots, w_n + \delta_n]$

ith entry: $[\,\cdot\,] = [A^T] [\nabla g(w)]$

Loss function:

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

$$\frac{\partial}{\partial z_i} g(w) = \text{ith entry of } A^T \nabla g(w)$$

$$\nabla_z g(w) = \nabla_z f(Az) = A^T \nabla g(w) = A^T \nabla g(Az)$$

$$L(\beta) = \|y\|_2^2 + \|X\beta\|_2^2 - 2\langle y, X\beta \rangle$$

$$\nabla L(\beta) = \nabla \|y\|_2^2 + \nabla \|X\beta\|_2^2 - 2\nabla \langle X^T y, \beta \rangle$$

$$0 + X^T \cdot 2X\beta - 2X^T y$$

Loss function: $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$.

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

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

Solve for optimal $\boldsymbol{\beta}^*$:

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

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?**

$$\begin{bmatrix} \texttt{ford} \\ \texttt{ford} \\ \texttt{honda} \\ \texttt{bmw} \\ \texttt{honda} \\ \texttt{ford} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Better approach: One Hot Encoding.

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

ford honda bmw

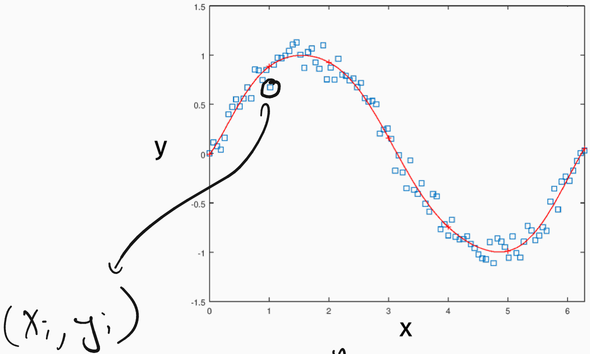
- Create a separate feature for every category, which is 1 when the variable is in that category, zero otherwise.
- Not too hard to do by hand, but you can also use library functions like `sklearn.preprocessing.OneHotEncoder`.

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:

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



$$L = \min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3\right)^2$$

Transform into a multiple linear regression problem:

$$\min_{\beta} \|y - X\beta\|_2^2$$

$$\tilde{L} = L$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^1 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta \\ \beta_3 \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

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

Could have:

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

$$[X\beta]_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3$$

Transformations can also be for multivariate data.

Example: Multinomial model.

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

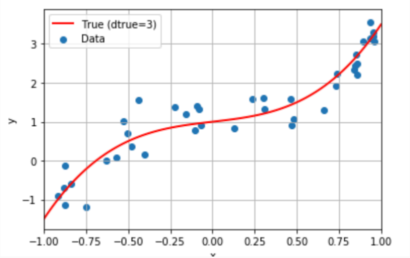
$$\begin{bmatrix} 1 & x_1 & x_1^2 & z_1 & z_1^2 & x_1 z_1 \\ 1 & x_2 & x_2^2 & z_2 & z_2^2 & x_2 z_2 \\ \vdots & \vdots & & \vdots & & \\ 1 & x_n & x_n^2 & z_n & z_n^2 & x_n z_n \end{bmatrix}$$

- 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.

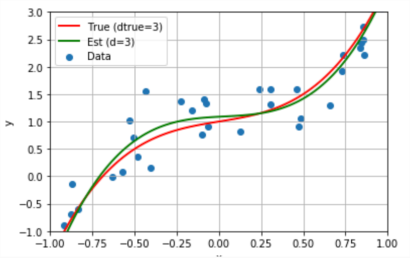
Simple experiment:

- Randomly select data points $x_1, \ldots, x_n \in [-1, 1]$.
- 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).



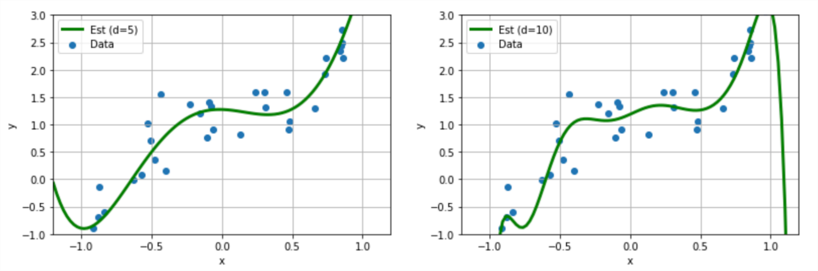
Simple experiment:

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



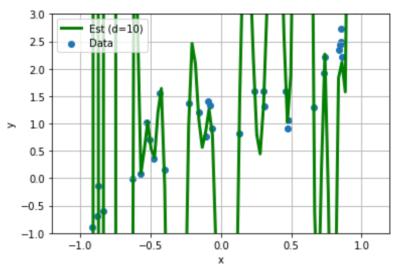
What if we fit a higher degree polynomial?

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

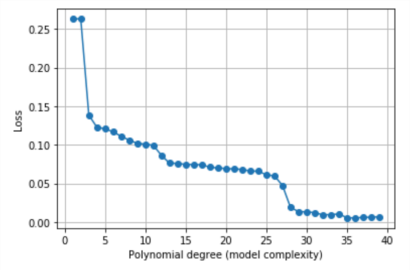


Even higher?

- Fit degree 40 polynomial under squared loss.



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

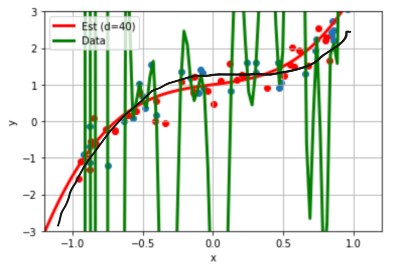


Is our model getting better and better?

Given the raw data, how do we know which model to choose?
Degree 3? Degree 5? Degree 40?

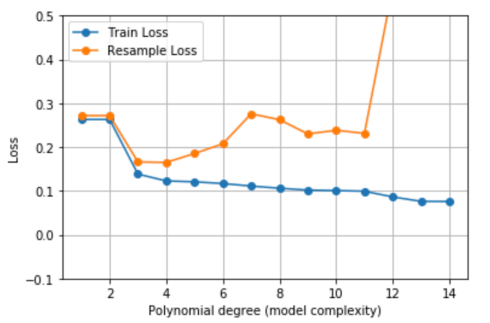
Problem: Loss alone is not informative for choosing model.

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.

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.

TRAIN-TEST PARADIGM

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 $f_1, \ldots, f_q$ by finding parameters which minimize the loss on $(\mathbf{X}_{tr}, \mathbf{y}_{tr})$.
- Evaluate loss of each trained model on $(\mathbf{X}_{ts}, \mathbf{y}_{ts})$.

50% data as train 50% as test?

80% as train, 20% test?

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 $f \in \{f_1, \ldots, f_q\}$ and parameter vector $\boldsymbol{\theta}$ to minimize the Risk:

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

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 $\boldsymbol{\theta}$, let

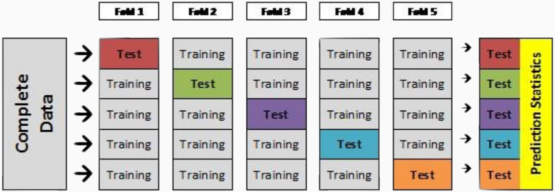
$$L_{ts}(f, \boldsymbol{\theta}) = \frac{1}{m} \sum_{\mathbf{x}, y \in (\mathbf{X}_{ts}, \mathbf{y}_{ts})} L\left(f(\mathbf{x}, \boldsymbol{\theta}) - y\right)$$

- **Claim:**[1]

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

- 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.

[1] Only true if f and $\boldsymbol{\theta}$ are chose *without looking at your test data.*



- 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?