Day 3: Overfitting and Generalization

Summer STEM: Machine Learning

Department of Electrical Engineering
NYU Tandon School of Engineering
Brooklyn, New York

August 5, 2020

Linear Regression

- What if we have multivariate data with $\mathbf{x}$ being a vector?
- Ex: $\mathbf{x}_i = [x_{i1}, x_{i2}]^T$

$$\hat{y}_1 = w_0 + w_1 x_{11} + w_2 x_{12}$$
$$\hat{y}_2 = w_0 + w_1 x_{21} + w_2 x_{22}$$
$$\vdots$$
$$\hat{y}_N = w_0 + w_1 x_{N1} + w_2 x_{N2}$$

- The model can be written as $\hat{y}_i = \begin{bmatrix} 1 & x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$

NYU TANDON SCHOOL OF ENGINEERING

Multilinear Regression

- In matrix-vector form

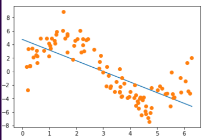
$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_{n2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

- Solution remains the same $\mathbf{w} = (X^T X)^{-1} X^T Y$
- Exercise: open `demo_multilinear.ipynb`

Outline

Polynomial Fitting

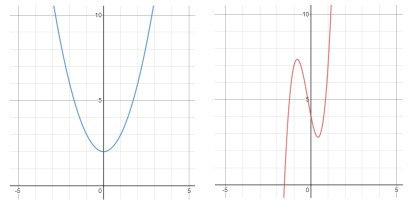
- We have been using linear model to fit our data. But it doesn't work well every time
- Some data have more complex relation that cannot be fitted well using a straight line
  - Ex: Projectile motion, Coulomb's law, Exponential growth/decay, ...



- Linear model does not look like a good fit for this data
- Can we use some other model to fit this data?

Polynomial Fitting

- Can we use a polynomial to fit our data?
- Polynomial: A sum of different powers of a variable
  - Examples: $y = x^2 + 2$, $y = 5x^3 - 3x^2 + 4$



- New model: $f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots$

Polynomial Fitting

- The process of fitting a polynomial is similar to linearly fitting multivariate data

- Recall the multivariable linear model
- $f(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + ...$
  - Where x_1, x_2, x_3... are different features

- If we treat x^2 as our second feature, x^3 as our third feature, x^4 as our fourth feature.... We can use the same procedure in multivariate regression.

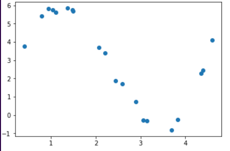
Polynomial Fitting

- Design matrix with the original feature: $$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$
- Design matrix with augmented polynomial features: $$\Phi(X) = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^D \\ 1 & x_2 & x_2^2 & \cdots & x_2^D \\ \vdots & & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^D \end{bmatrix}$$
- For the polynomial fitting, we just added columns of features that are powers of the original feature

NYU TANDON SCHOOL OF ENGINEERING

Demo: Fit a polynomial

- You are given the data set below with x and y values



- Try to fit the data using a polynomial with a certain degree
- Calculate mean square error between the sample y and your predicted y
- Try different polynomial degree and see if you can improve the mse
- Plot your polynomial over the data points

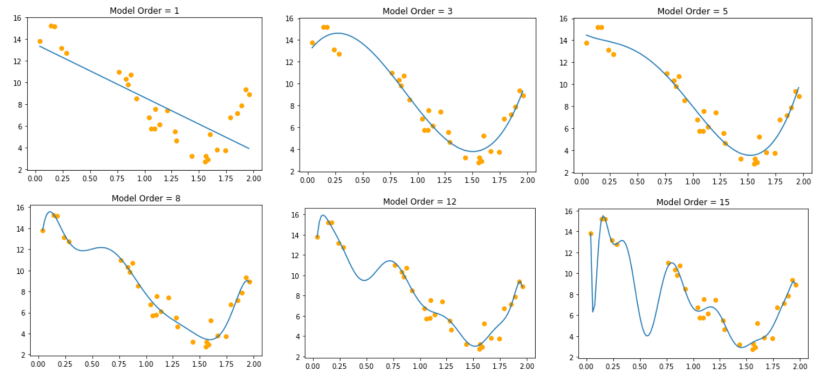
Outline

Overfitting

- We learned how to fit our data using polynomials of different order
- With a higher model order, we can fit the data with increasing accuracy
- As you increase the model order, at certain point it is possible find a model that fits your data perfectly (ie. zero error)
- What could be the problem?

NYU TANDON SCHOOL OF ENGINEERING

Overfitting



- Which of these model do you think is the best? Why?

Overfitting

- The problem is that we are only fitting our model using data that is given
- Data usually contains noise
- When a model becomes too complex, it will start to fit the noise in the data
- What happens if we apply our model to predict some data that the model has never seen before? It will not work well.
- This is called over-fitting

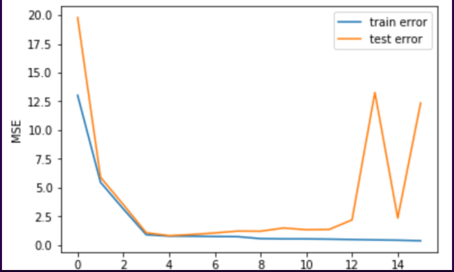
Tuning Hyper-parameters

- Motivation: never determine a **hyper-parameter** using the training set
- **Hyper-Parameter**: a parameter of the algorithm that is not a model-parameter solved for in optimization.
  - Ex: The order of the polynomials M
- Solution: split dataset into three
  - **Training set**: to compute the model-parameters ($\mathbf{w}$)
  - **Validation set**: to tune hyper-parameters (M)
  - **Test set**: to compute the performance of the algorithm (MSE)

Training and Test Error

We use the training set to find the model parameters and the validation set to tune the hyper-parameters. Finally, we use a **test set** to evaluate the model performance.

- As we increase the order of the polynomials M, the training error decreases.
- However, the test error first decreases but increases again at a certain point.



Outline

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection
- Is there another way? Talk among your classmates.

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection
- Is there another way? Talk among your classmates.
  - Solution: We can change our cost function.

Weight Based Regularization

- Looking back at the polynomial overfitting
- Notice that weight-size increases with overfitting

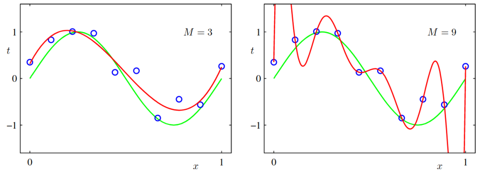


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

NYU TANDON SCHOOL OF ENGINEERING

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

- Penalize complexity by simultaneously minimizing weight values.
- λ here is also a **hyper-parameter**
  - λ determines relative importance

Table 1.2 Table of the coefficients $\mathbf{w}^\star$ for $M = 9$ polynomials with various values for the regularization parameter λ. Note that $\ln \lambda = -\infty$ corresponds to a model with no regularization, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of λ increases, the typical magnitude of the coefficients gets smaller.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01