Day 3: Overfitting and Generalization

Summer STEM: Machine Learning

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

June 23, 2020

Outline

1. Leftovers from Day 2
2. Polynomial Regression
3. Train and Test Error, Overfitting
4. Regularization

Least Square Solution

- Model:

$$f(x) = w_0 + w_1 x$$

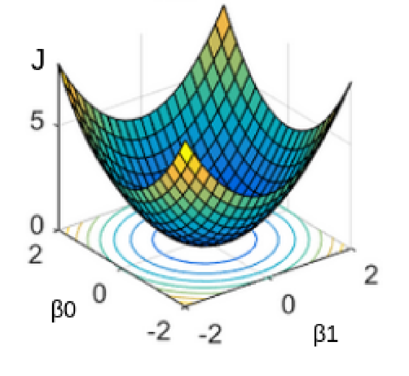
- Loss:

$$J(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - f(x_i)\|^2$$

- Optimization: find w_0, w_1 such that $J(w_0, w_1)$ is the least possible value (hence the name "least square").

Loss Landscape

Plot the loss against the parameters:



Linear Regression

- Linear models: For scalar-valued feature x, this is $f(x) = w_1 x + w_0$
- One of the simplest machine learning model, yet very powerful.
- Two ways to get the solution, we will show them later.

Least Square Solution: Using Statistics

- Solution:

$$f(x) = \bar{y} + \frac{\sigma_{xy}}{\sigma_x^2}(x - \bar{x})$$

$$w_1 = \frac{\sigma_{xy}}{\sigma_x^2}, \quad w_0 = \bar{y} - w_1\bar{x}$$

- Prediction:

$$f(x) = w_0 + w_1 x$$

Least Square Solution: Using Pseudo-Inverse

- For N data points (x_i, y_i) we have,

$$
\begin{aligned}
y_1 &\approx w_0 + w_1 x_1 \\
y_2 &\approx w_0 + w_1 x_2 \\
&\vdots \\
y_N &\approx w_0 + w_1 x_N .
\end{aligned}
$$

Linear Regression

- In matrix form we have,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

- We can write it as $Y \approx X\mathbf{w}$. We call X the design matrix.
- Exercise: verify $\|Y - X\mathbf{w}\|^2 = \sum_{i=1}^{N} \|y_i - (w_0 + w_1 x_i)\|^2$

Linear Least Square

- $\min_{\mathbf{w}} \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- Using the psuedo-inverse (only square matrices have an inverse),

$$
\begin{aligned}
Y &= X\mathbf{w} \\
X^T Y &= X^T X\mathbf{w} \\
(X^T X)^{-1} X^T Y &= (X^T X)^{-1} X^T X\mathbf{w} \\
(X^T X)^{-1} X^T Y &= \mathbf{w}.
\end{aligned}
$$

Linear Regression

- What if we have multivariate data with $\mathbf{x}$ being a vector?
- $y = \mathbf{w}^T\mathbf{x}$, here both $\mathbf{w}$ and $\mathbf{x}$ are vectors.
- Ex: $\mathbf{x}_i = [1, x_{i1}, x_{i2}]^T$ and $\mathbf{w} = [w_0, w_1, w_2]^T$

  $y_1 \approx w_0 + w_1 x_{11} + w_2 x_{12}, \cdots$
- In matrix-vector form

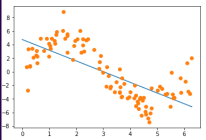
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \approx \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_{n2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

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

Outline

Polynomial Fitting

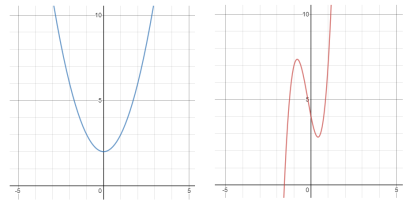
- We have been using linear model to fit our data. But it doesn't work well every time
- Some data have more complex relation that cannot be fitted well using a straight line
  - Ex: Projectile motion, Coulomb's law, Exponential growth/decay, ...



- 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: $y = x^2 + 2$, $y = 5x^3 - 3x^2 + 4$



- Polynomial Model: $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots$

Polynomial Fitting

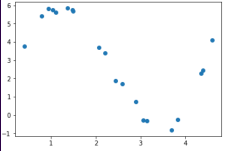
- Polynomial Model: $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + ...$
- The process of fitting a polynomial is similar to linearly fitting multivariate data
- Recall the linear model for multivariable
- $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + ...$
  - Where x_1, x_2, x_3... are different features
- If we 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 for linear fit!

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

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

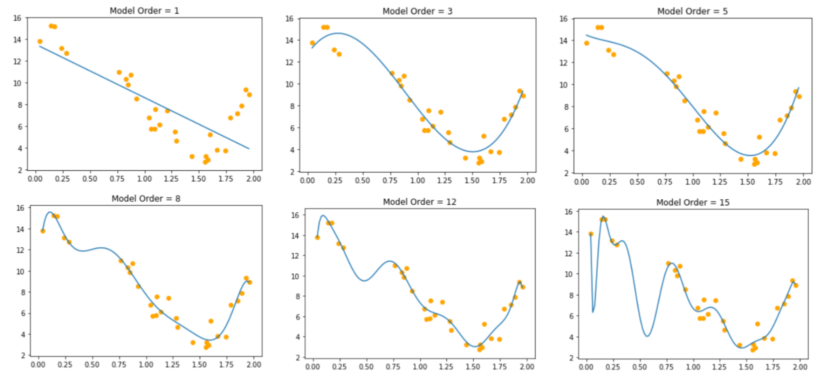
Outline

NYU TANDON SCHOOL OF ENGINEERING

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?

Overfitting

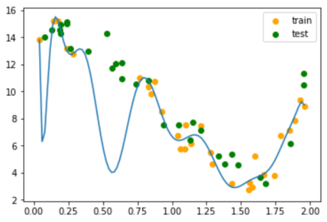


- Which of these model do you think is the best? Why?

NYU TANDON SCHOOL OF ENGINEERING

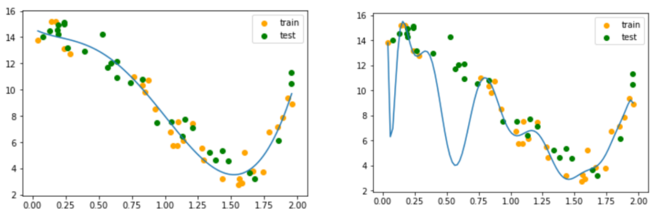
Overfitting

- 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



Overfitting

- Split the data set into a train set and a test set
- Train set will be used to train the model
- The test set will not be seen by the model during the training process
- Use test set to evaluate the model when a model is trained

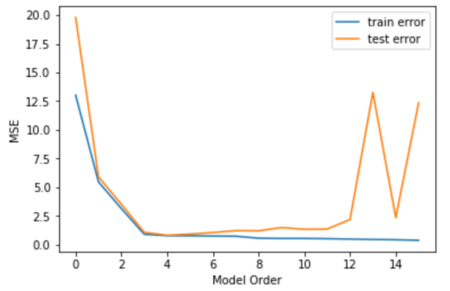


- With the training and test sets shown, which one do you think is the better model now?

NYU TANDON SCHOOL OF ENGINEERING

Train and Test Error

- Plot of train error and test error for different model order
- Initially both train and test error go down as model order increase
- But at a certain point, test error start to increase because of overfitting



NYU TANDON SCHOOL OF ENGINEERING

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

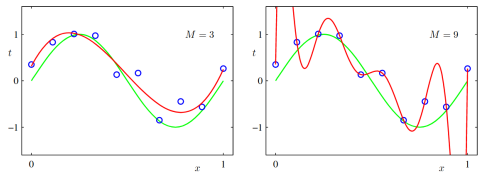


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	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

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.
- We call λ 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

NYU TANDON SCHOOL OF ENGINEERING

Tuning Hyper-parameters

- Motivation: never determine a hyper-parameter based on training data
- **Hyper-Parameter**: a parameter of the algorithm that is not a model-parameter solved for in optimization.
  - Ex: λ weight regularization value vs. model weights ($\mathbf{w}$)
- Solution: split dataset into three
  - **Training set**: to compute the model-parameters ($\mathbf{w}$)
  - **Validation set**: to tune hyper-parameters (λ)
  - **Test set**: to compute the performance of the algorithm (MSE)