NYU

Overfitting and Generalization

NYU K12 STEM Education: Machine Learning

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

Course Details

- Course Website
- Instructors:

Rugved Mhatre
rugved.mhatre@nyu.edu

Akshath Mahajan
akshathmahajan@nyu.edu

Outline

1. Review
2. Polynomial Fitting
3. Regularization
4. Optimization

Review

- Statistics - Mean, Variance, Covariance

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression
  - Classification

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression
  - Classification
- Linear Regression

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression
  - Classification
- Linear Regression
  - Error Functions

Review

- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression
  - Classification
- Linear Regression
  - Error Functions
  - Least Square Solution

Review

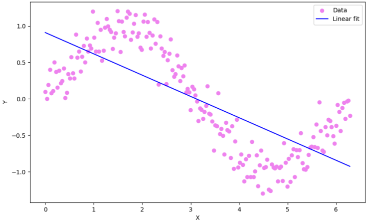
- Statistics - Mean, Variance, Covariance
- Types of Supervised Learning problems:
  - Regression
  - Classification
- Linear Regression
  - Error Functions
  - Least Square Solution
- Multivariable Linear Regression

Outline

1. Review
2. Polynomial Fitting
3. Regularization
4. Optimization

Polynomial Fitting

- We have been using straight lines 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



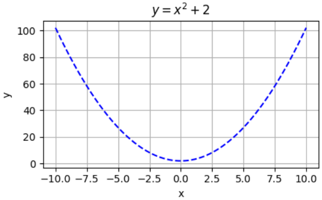
- Can we use some other model to fit this data?

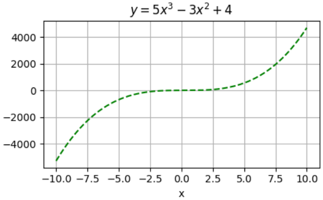
Polynomial Fitting

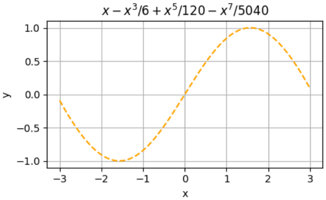
Can we use a polynomial to fit our data?

Polynomial: A sum of different powers of a variable

Example:







Polynomial Fitting

Polynomials of x: $\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$

m is called the order of the polynomial.

The process of fitting a polynomial is similar to linearly fitting multivariable data.

Polynomial Fitting

In matrix-vector form:

$$\begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_N} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

This can still be written as: $\hat{Y} = XW$

Loss:

$$J(W) = \frac{1}{N} ||Y - XW||^2$$

Polynomial Fitting

The i^{th} row of the design matrix X is simply a transformed feature:

$$\phi(x_i) = (1, x_i, x_i^2, \cdots, x_i^m)$$

Original Design Matrix:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

Polynomial Fitting

Design matrix after feature transformation:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix}$$

For the polynomial fitting, we just added columns of features that are powers of the original feature.

Linear Regression

Model:

$$\hat{y} = W^T \phi(x)$$

Loss:

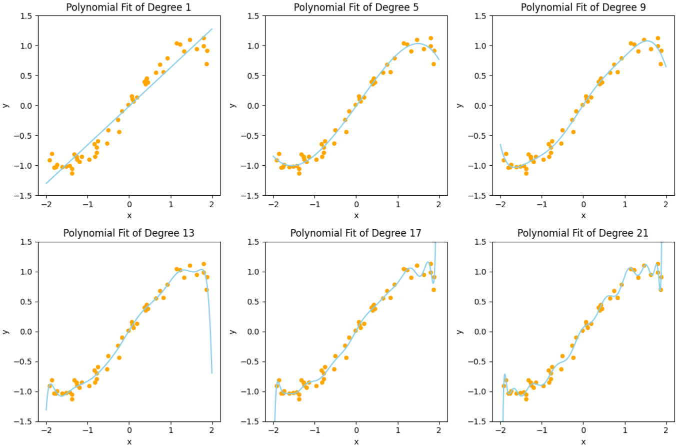
$$J(W) = \frac{1}{N} ||Y - XW||^2$$

Find W that minimizes $J(W)$

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



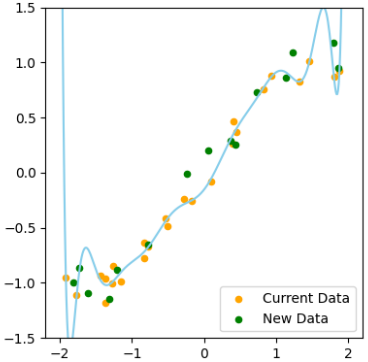
Which of these model do you think is the best? Why?

Overfitting

- Open Fit a Polynomial Demo from Course Website

Overfitting

- 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 overfitting

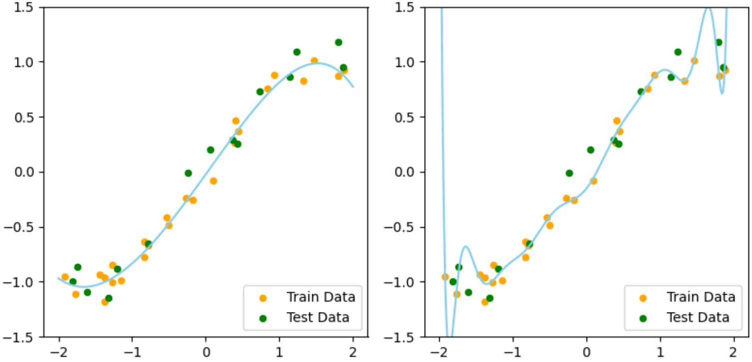


Solution to Overfitting

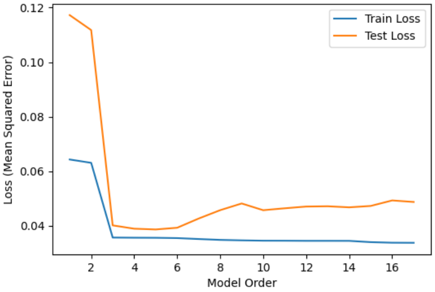
- Split the data set into a train set and a test set
- Train set will be used to train the model
- The test set will not be seen by the model during the training process
- Use test set to evaluate the model when a model is trained

Solution to Overfitting

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



Train and Test Loss



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

Outline

1. Review
2. Polynomial Fitting
3. Regularization
4. Optimization

Regularization

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

- **Regularization:** methods to prevent overfitting
- One way to regularize is by model order selection.
- Is there another way?

Regularization

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

- **Regularization:** methods to prevent overfitting
- One way to regularize is by model order selection.
- Is there another way?
- We can change the cost function

Weight Based Regularization

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

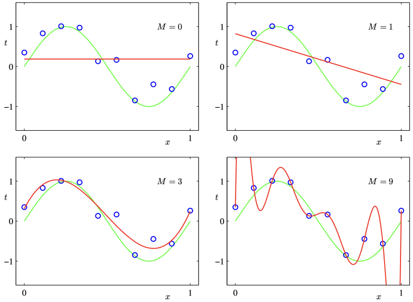


Table of the coefficients $\mathbf{w}^{\star}$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

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

Weight Based Regularization

New Cost Function:

$$J(W) = \frac{1}{N}||Y - XW||^2 + \lambda||W||^2$$

- Penalize complexity by simultaneously minimizing weight values.
- We call λ a **hyper-parameter**
  - λ determines relative importance

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

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.
  Example: λ weight regularization value vs. model weights (W)
- Solution: Split dataset into three:
  - **Training Set:** to compute the model parameters (W)
  - **Validation Set:** to tune the hyper-parameters (λ)
  - **Testing Set:** to compute the performance of the ML algorithm (MSE)

Overfitting and Regularization

- Open Overfitting, Weight Regularization Demo from Course Website

Outline

1. Review
2. Polynomial Fitting
3. Regularization
4. Optimization

Non-linear Optimization

- Cannot rely on closed form solutions
  - Computation Efficiency: operations like inverting a matrix is not efficient
  - For more complex problems such as neural networks, a closed form solution is not always available
- Need an optimization technique to find an optimal solution
  - Machine learning practitioners use **gradient** based methods

Gradient Descent Algorithm

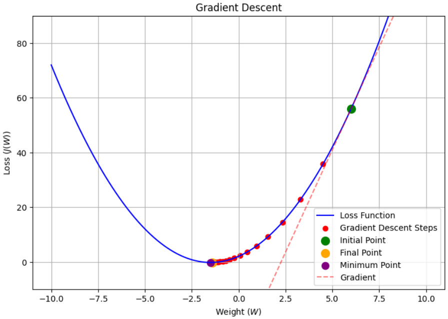
Update Rule:

Repeat{

$$W_{\text{new}} = W - \alpha \nabla J(W)$$

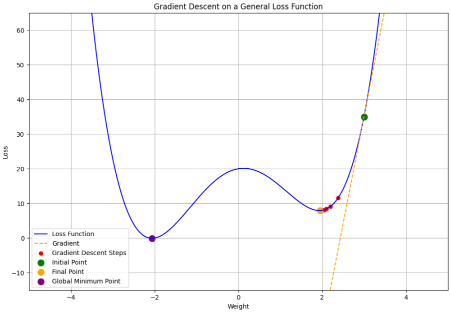
}

α is the learning rate

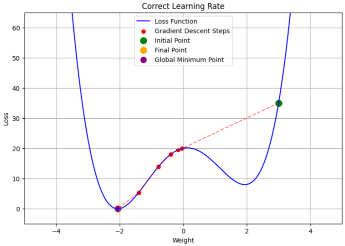


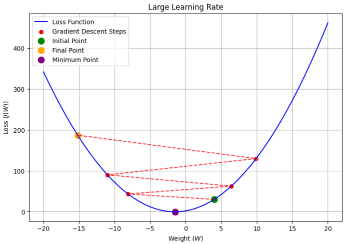
Loss Function Contours

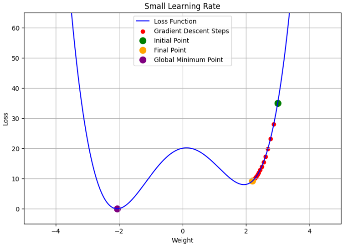
- Most loss function contours are not perfectly parabolic
- Our goal is to find a solution that is very close to global minimum by the right choice of hyper-parameters.



Understanding Learning Rate







Gradient Descent Animations