

Day 4: Linear Classifiers

Summer STEM: Machine Learning

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

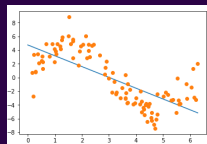
June 25, 2020

Outline

- 1 Leftovers from Day 3
- 2 Regularization
- 3 Non-linear Optimization
- 4 Logistic Regression
- 5 Lab: Diagnosing Breast Cancer
- 6 Multiclass Classification
- 7 Lab: Iris Dataset

Polynomial Fitting

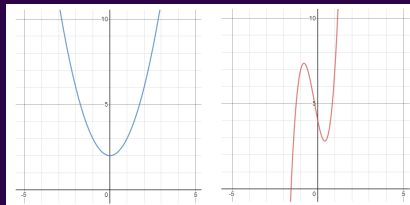
- We have been using straight lines to fit our data. But it doesn't work well every time
- Some data have more complex relation that cannot be fitted well using a straight line



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



Polynomial Fitting

- Polynomials of x : $\hat{y} = w_0 + w_1x + w_2x^2 + w_3x^3 + \cdots + w_Mx^M$
- M is called the order of the polynomial.
- The process of fitting a polynomial is similar to linearly fitting multivariate data.

Polynomial fitting

- Rewrite in matrix-vector form

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

- This can still be written as

$$Y \approx X\mathbf{w}$$

- Loss $J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- The i -th row of the design matrix X is simply a transformed feature $\phi(x_i) = (1, x_i, x_i^2, \dots, x_i^M)$

Polynomial Fitting

- Original design matrix:

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

- Design matrix after feature transformation:

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

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

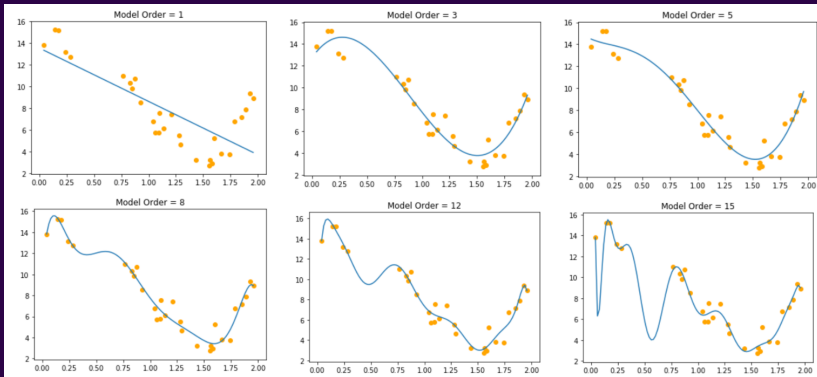
Linear Regression

- Model $\hat{y} = \mathbf{w}^T \phi(\mathbf{x})$
- Loss $J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- Find \mathbf{w} that minimizes $J(\mathbf{w})$

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?

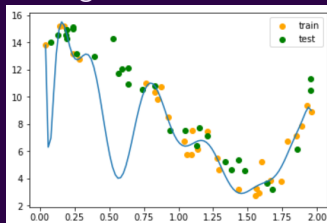
Overfitting



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

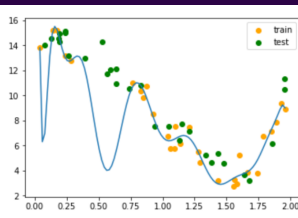
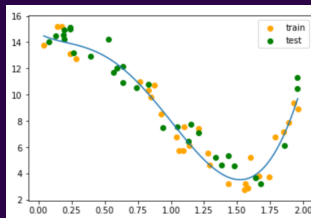
Overfitting

- 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

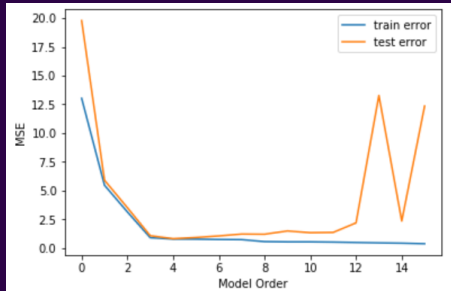
- Split the data set into a train set and a test set
- Train set will be used to train the model
- The test set will not be seen by the model during the training process
- Use test set to evaluate the model when a model is trained



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

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



Outline

- 1 Leftovers from Day 3
- 2 Regularization
- 3 Non-linear Optimization
- 4 Logistic Regression
- 5 Lab: Diagnosing Breast Cancer
- 6 Multiclass Classification
- 7 Lab: Iris Dataset

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?

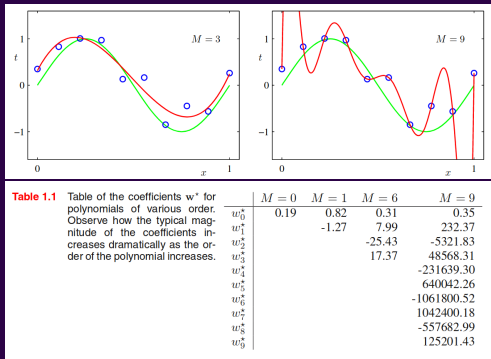
- **Regularization:** methods to prevent overfitting
 - We just covered regularization by model order selection
- Is there another way? Talk among your classmates.

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



New Cost Function

$$J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$$

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

Table 1.2 Table of the coefficients w^* 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^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	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.
 - 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)

Outline

- 1 Leftovers from Day 3
- 2 Regularization
- 3 Non-linear Optimization**
- 4 Logistic Regression
- 5 Lab: Diagnosing Breast Cancer
- 6 Multiclass Classification
- 7 Lab: Iris Dataset

Motivation

- 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

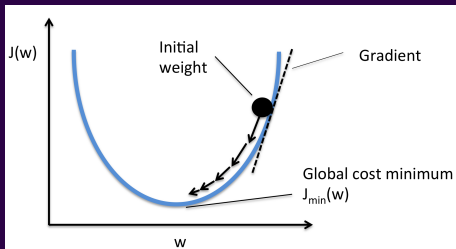
■ Update Rule

Repeat{

$$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} J$$

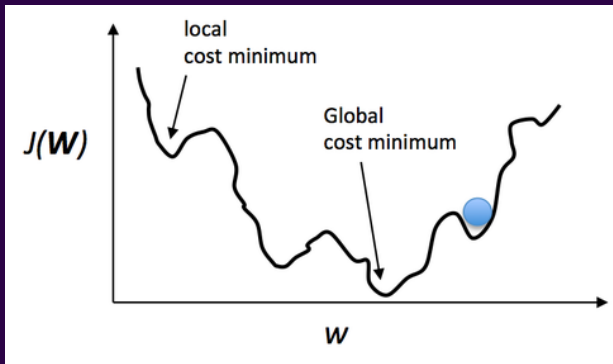
}

α is the learning rate

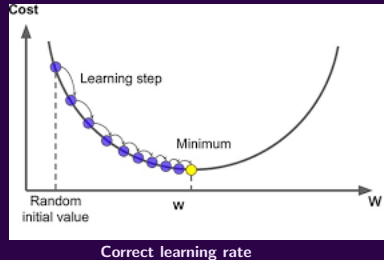
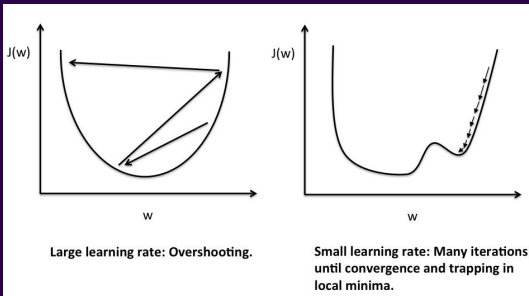


General Loss Function Contours

- 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



Some Animations

- Demonstrate gradient descent animation