Day 3: Generalization Error

Summer STEM: Machine Learning

Nikola Janjušević, Akshaj Kumar Veldanda, Jacky Yuan, Tejaishwarya Gagadam

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

July 10, 2019

NYU TANDON SCHOOL OF ENGINEERING

Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

NYU TANDON SCHOOL OF ENGINEERING

Lab: Find/Build and fit your own data

1. Find your a data set
   - Google: "[subject you're interested in] dataset"
   - `https://archive.ics.uci.edu/ml/datasets.php`
   - `https://toolbox.google.com/datasetsearch`

   – or –
2. Build your own data set
   - Only need 10+ samples

NYU TANDON SCHOOL OF ENGINEERING

Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

General Steps to Solve a Machine Learning Problem

- Load and visualize data

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = wx + b$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = wx + b$
- Choose an appropriate error function

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = wx + b$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N} (y_i - (b + wx_i))^2$

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = wx + b$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N} (y_i - (b + wx_i))^2$
- Find parameters that minimize the error function

NYU TANDON SCHOOL OF ENGINEERING

General Steps to Solve a Machine Learning Problem

- Load and visualize data
  - (x_i, y_i), $i = 1, ..., n$
- Find an appropriate model to fit the data
  - Eg: Linear model is $\hat{y} = wx + b$
- Choose an appropriate error function
  - $MSE = \sum_{i=1}^{N} (y_i - (b + wx_i))^2$
- Find parameters that minimize the error function
  - Select b, w to minimize the error function

NYU TANDON SCHOOL OF ENGINEERING

Extending the Model to Multi-variable Data

- Model: $\hat{y} = w_0 \times 1 + w_1 x_1 + w_2 x_2 + ... + w_D x_D$
- Design Matrix: Let, $X = \begin{bmatrix} 1 & x_{1_1} & \cdots & x_{1_D} \\ 1 & x_{2_1} & \cdots & x_{2_D} \\ \vdots & & \ddots & \\ 1 & x_{N_1} & \cdots & x_{N_D} \end{bmatrix}$
- We say $\mathbf{w}^\star$ solves $\mathbf{y} = X\mathbf{w}$ in the least squares sense, where

$$\mathbf{w}^\star = X^\dagger \mathbf{y}$$

- This $\mathbf{w}^\star$ is the unique set of parameters that minimize the squared error

Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

Robot Arm Calibration

- Let's train a model based on the given data.
- In this lab we're going to:
  - Predict the *current* drawn
  - Predictors, X: Robot arm's joint angles, velocity, acceleration, strain gauge readings (load measurement).

Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

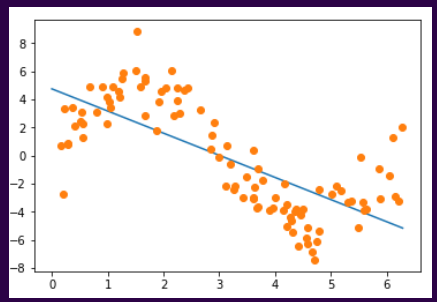
5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

Polynomial Fitting

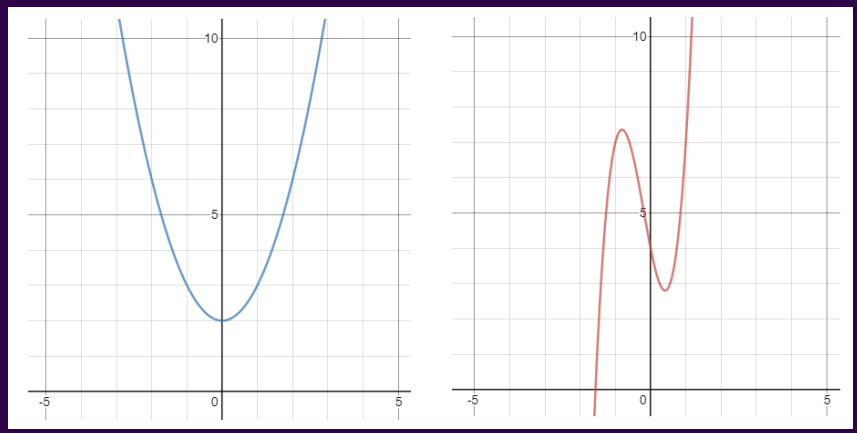
- We have been using linear model to fit our data. But it doesn't work well every time
- Some data have more complex relation that cannot be fitted well using a straight line
  - Ex: Projectile motion, Coulomb's law, Exponential growth/decay, ...



- 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!

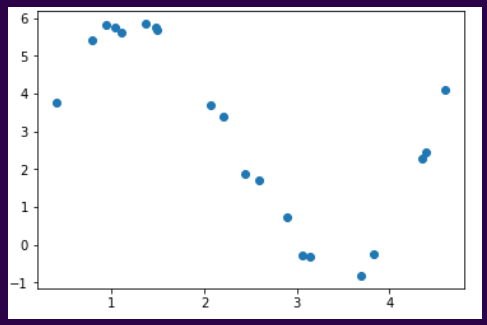
NYU TANDON SCHOOL OF ENGINEERING

Polynomial Fitting

- Design Matrix for Linear: $A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$
- Design Matrix for Polynomial: $X = \begin{bmatrix} 1 & x_1 & x_2^2 & \cdots & x_1^D \\ 1 & x_2 & x_2^2 & \cdots & x_2^D \\ \vdots & & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^D \end{bmatrix}$
- For the polynomial fitting, we just added columns of features that are powers of the original feature

Lab: 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

NYU TANDON SCHOOL OF ENGINEERING

Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

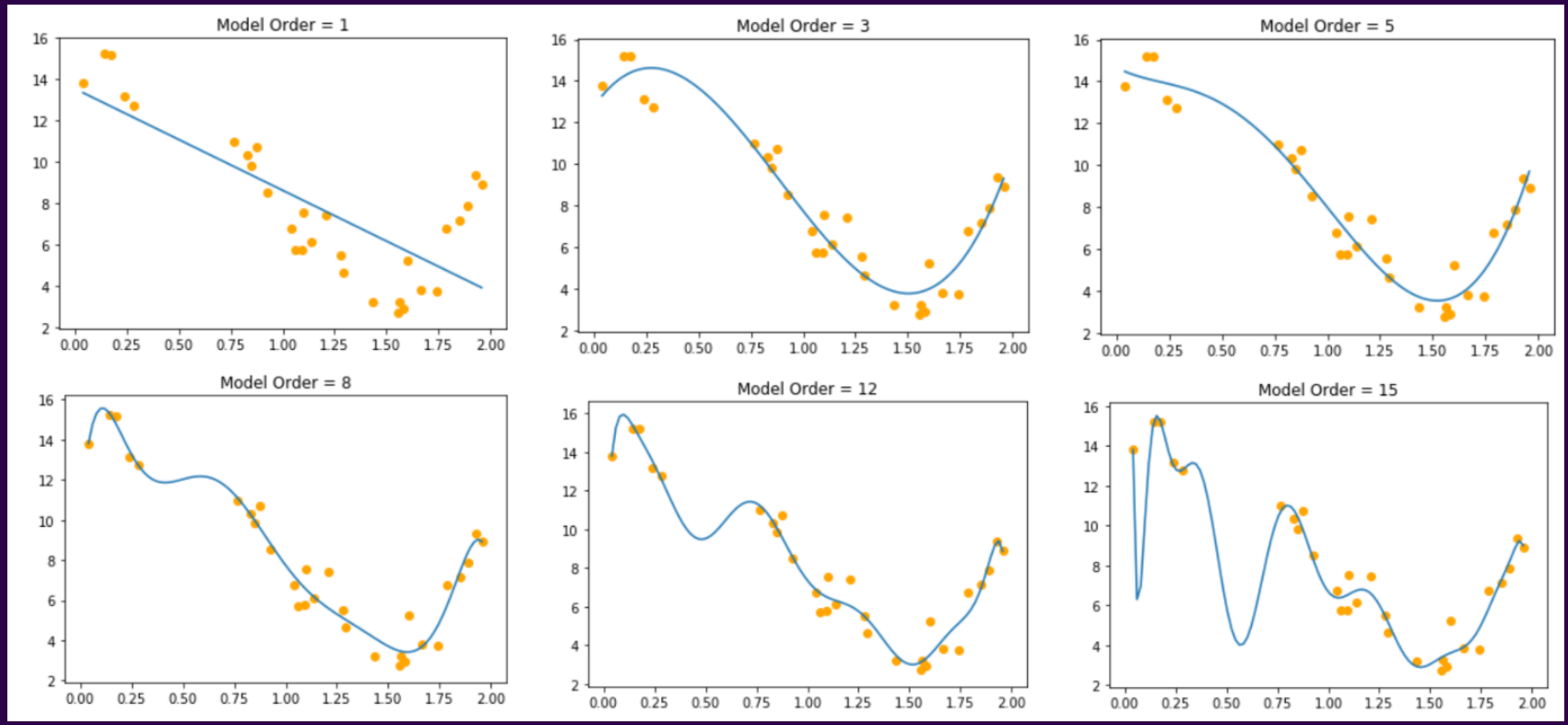
6 Regularization

7 Non-Linear Optimization

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?

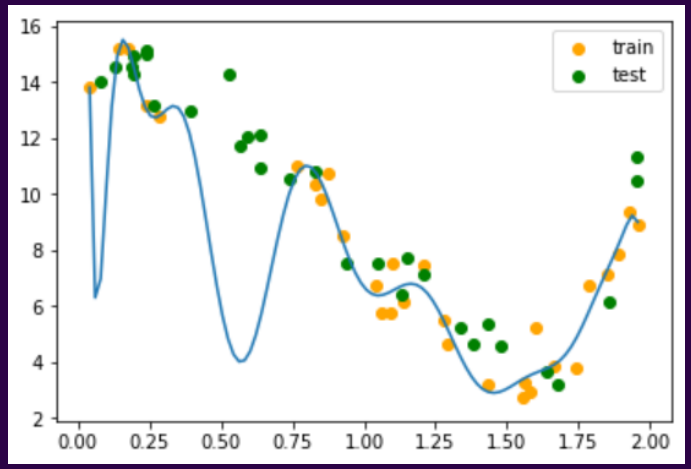
Overfitting



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

Overfitting

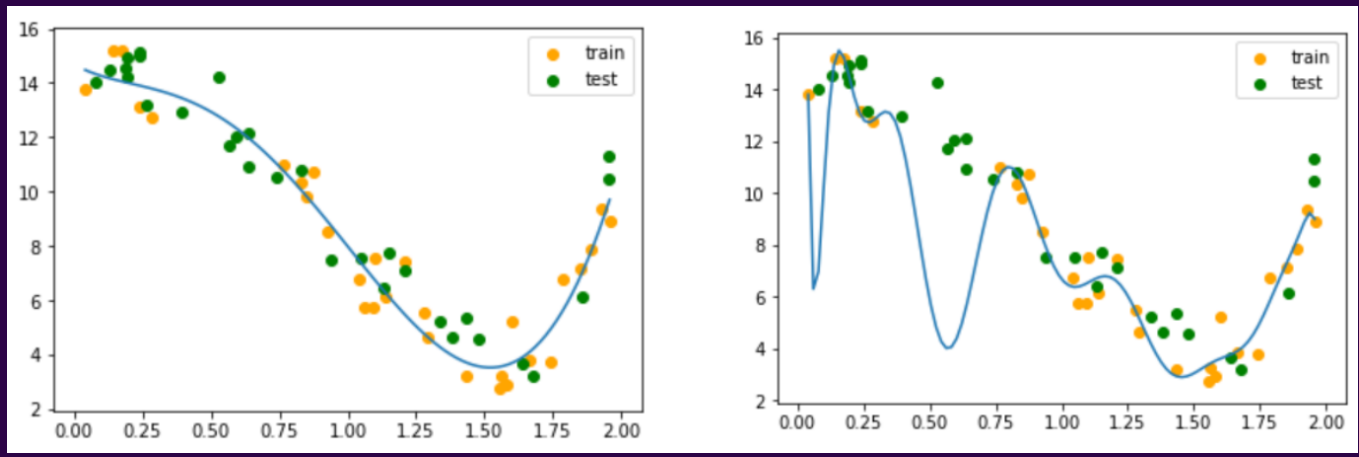
- The problem is that we are only fitting our model using data that is given
- Data usually contains noise
- When a model becomes too complex, it will start to fit the noise in the data
- What happens if we apply our model to predict some data that the model has never seen before? It will not work well.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



NYU TANDON SCHOOL OF ENGINEERING

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

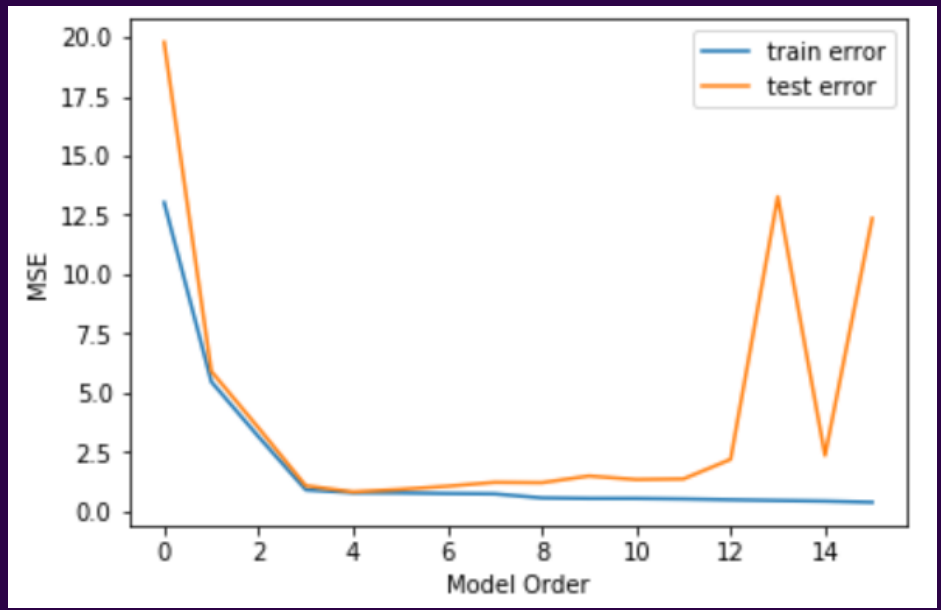


- With the training and test sets shown, which one do you think is the better model now?

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



Outline

1 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

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
- Running K-folds for cross-validation is intensive

NYU TANDON SCHOOL OF ENGINEERING

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

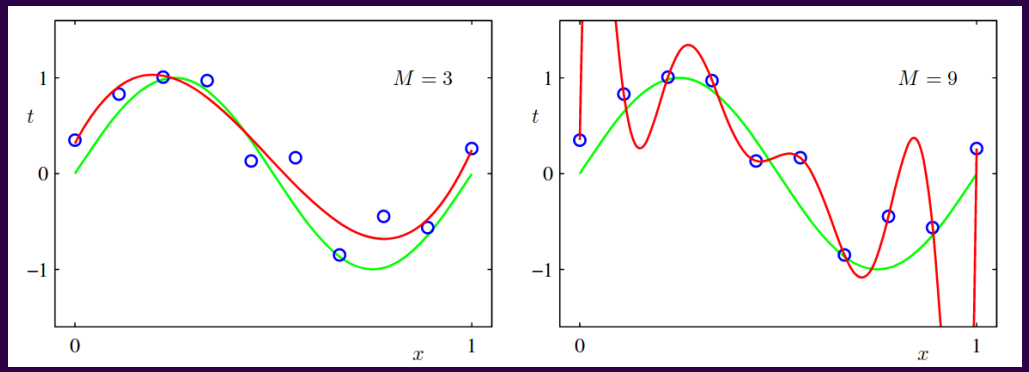
- **Regularization**: methods to prevent overfitting
  - We just covered regularization by model order selection
- Running K-folds for cross-validation is intensive
- Is there another way? Talk among your classmates.

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
- Running K-folds for cross-validation is intensive
- Is there another way? Talk among your classmates.
  - Solution: We can change our cost function.

Weight Based Regularization

- Looking back at the polynomial overfitting



Weight Based Regularization

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

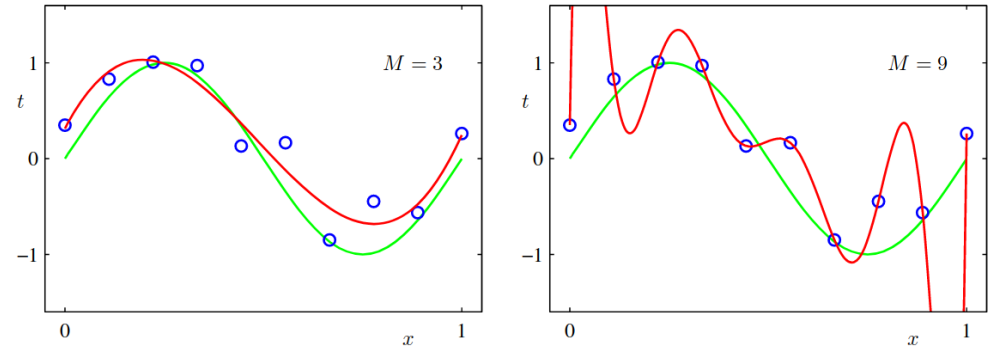


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

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

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2$$

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

- Penalize complexity by simultaneously minimizing weight values.

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

- Penalize complexity by simultaneously minimizing weight values.
- We call λ a **hyperparameter**

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

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

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

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

New Cost Function

$$J = \sum_{i=1}^{N} (y_i - y_{i\,pred})^2 + \lambda \sum_{j=1}^{D} (w_j)^2$$

- Penalize complexity by simultaneously minimizing weight values.
- We call λ a **hyperparameter**
  - 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

Cross-Validation

- Motivation: never determine a hyper-parameter based on training data

Cross-Validation

- 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.

Cross-Validation

- 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 ($\mathbf{w}$)

Cross-Validation

- 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

Cross-Validation

- 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 ($\mathbf{w}$)

Cross-Validation

- 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)

Cross-Validation

- 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 Lab: Simple Linear Regression

2 Review of Day 2

3 Lab: Robot Arm Calibration

4 Polynomial Regression

5 Train and Test Error, Overfitting

6 Regularization

7 Non-Linear Optimization

Motivation

- Cannot rely on closed form solutions

Motivation

- Cannot rely on closed form solutions
  - Computation efficiency: operations like inverting a matrix is not efficient

Motivation

- Cannot rely on closed form solutions
  - Computation efficiency: operations like inverting a matrix is not efficient
  - For more complex problems, like neural network, a closed form solution is not always available

Motivation

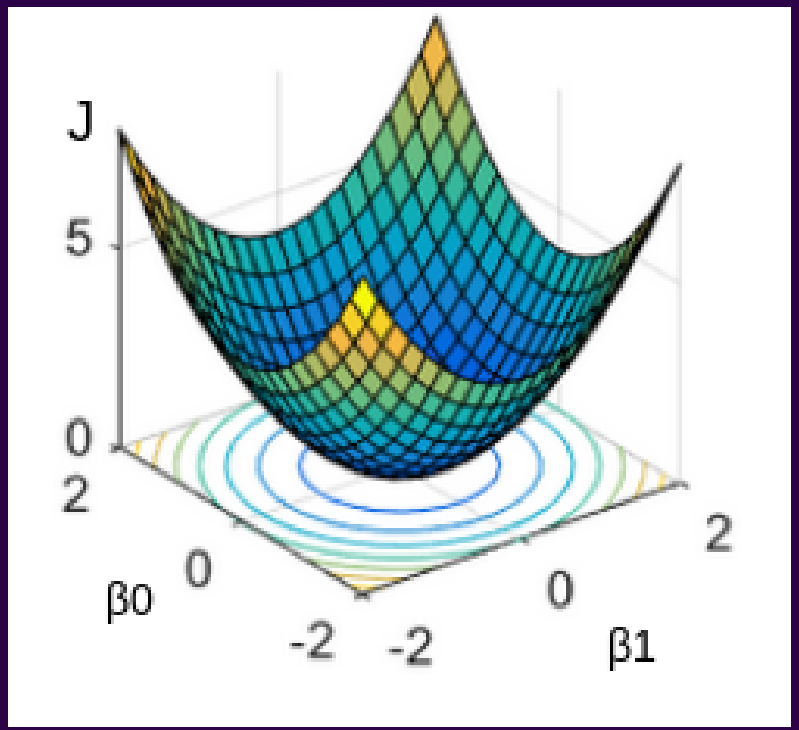
- Cannot rely on closed form solutions
  - Computation efficiency: operations like inverting a matrix is not efficient
  - For more complex problems, like neural network, a closed form solution is not always available
- Need an optimization technique to find an optimal solution

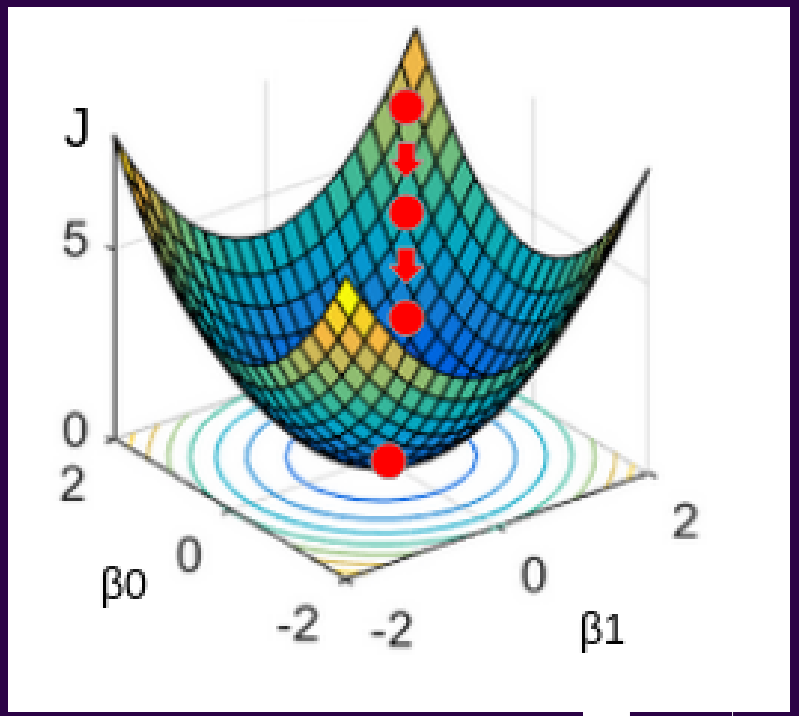
Motivation

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

Understanding Optimization

- *Recap* $\hat{y} = w_0 + w_1 x$
- *Loss*, $J = \Sigma_{i=1}^{N}(y_i - \hat{y}_i)^2 \implies J = \Sigma_{i=1}^{N}(y_i - w_0 - w_1 x_i)^2$
- Want to find $w_0 and w_1$ that minimizes J





Gradient Descent Algorithm

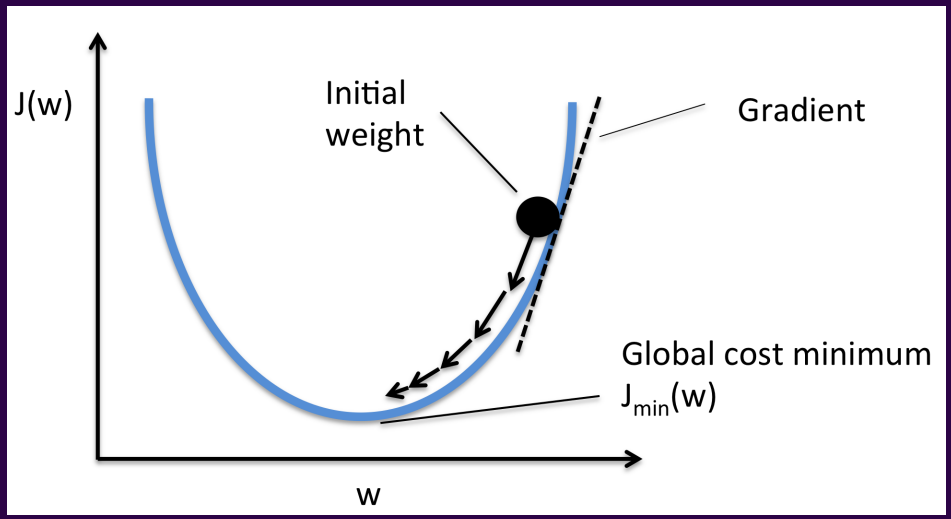
- Update Rule

  Repeat{

  $$w_{new} = w - \alpha \frac{dJ}{dw}$$

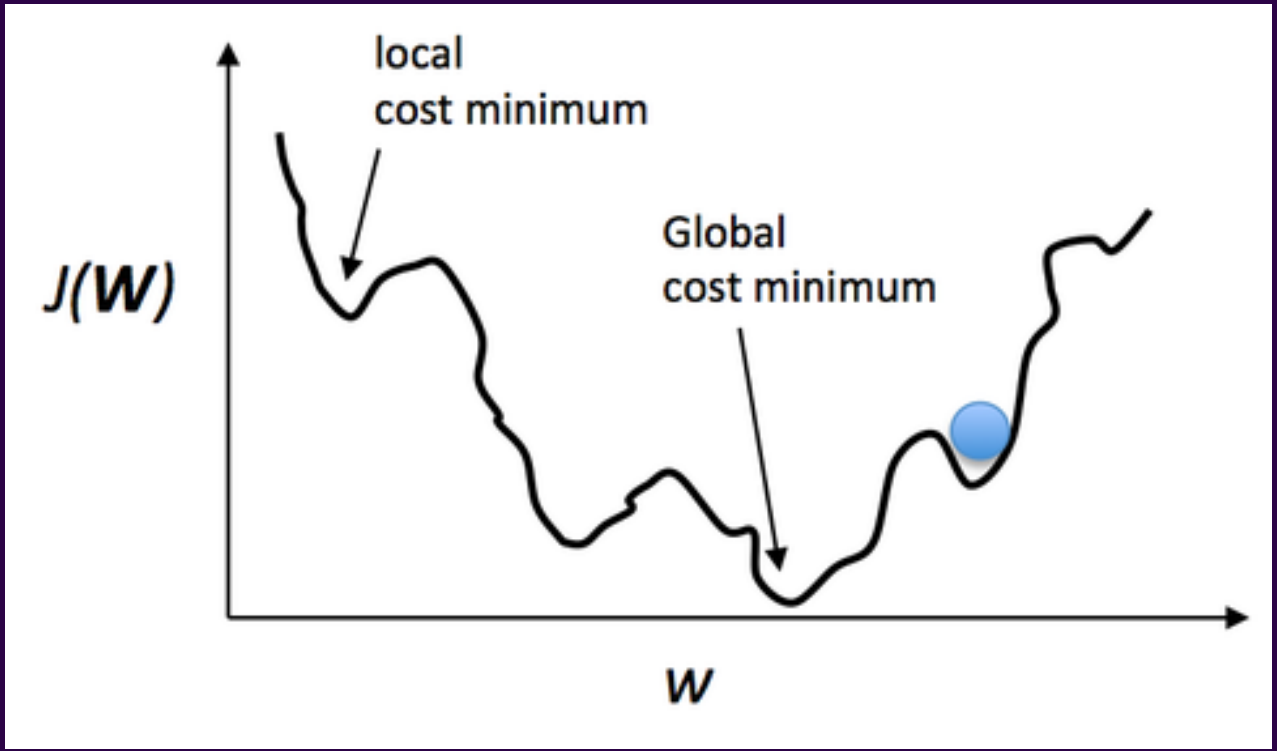
  }

  α is the learning rate



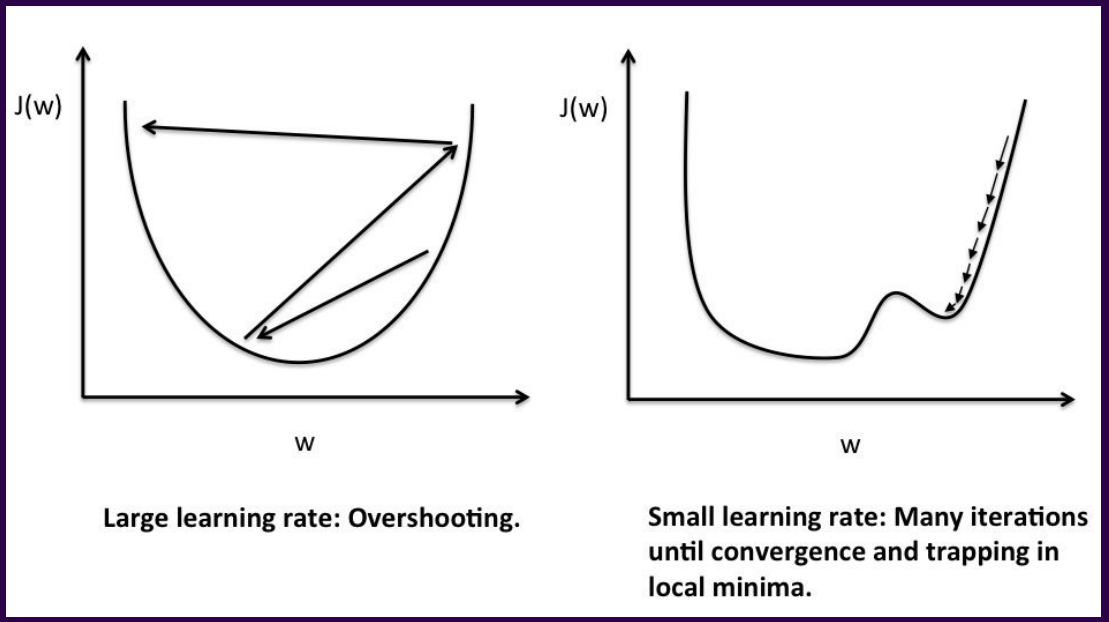
General Loss Function Contours

- 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



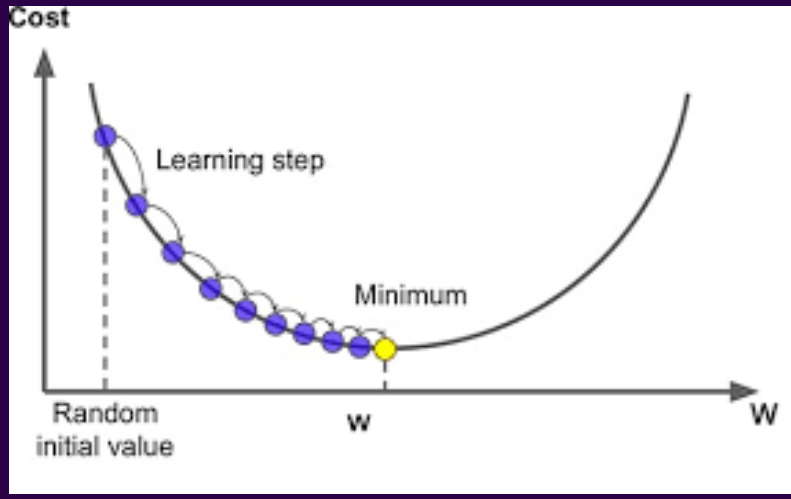
NYU TANDON SCHOOL OF ENGINEERING

Understanding Learning Rate



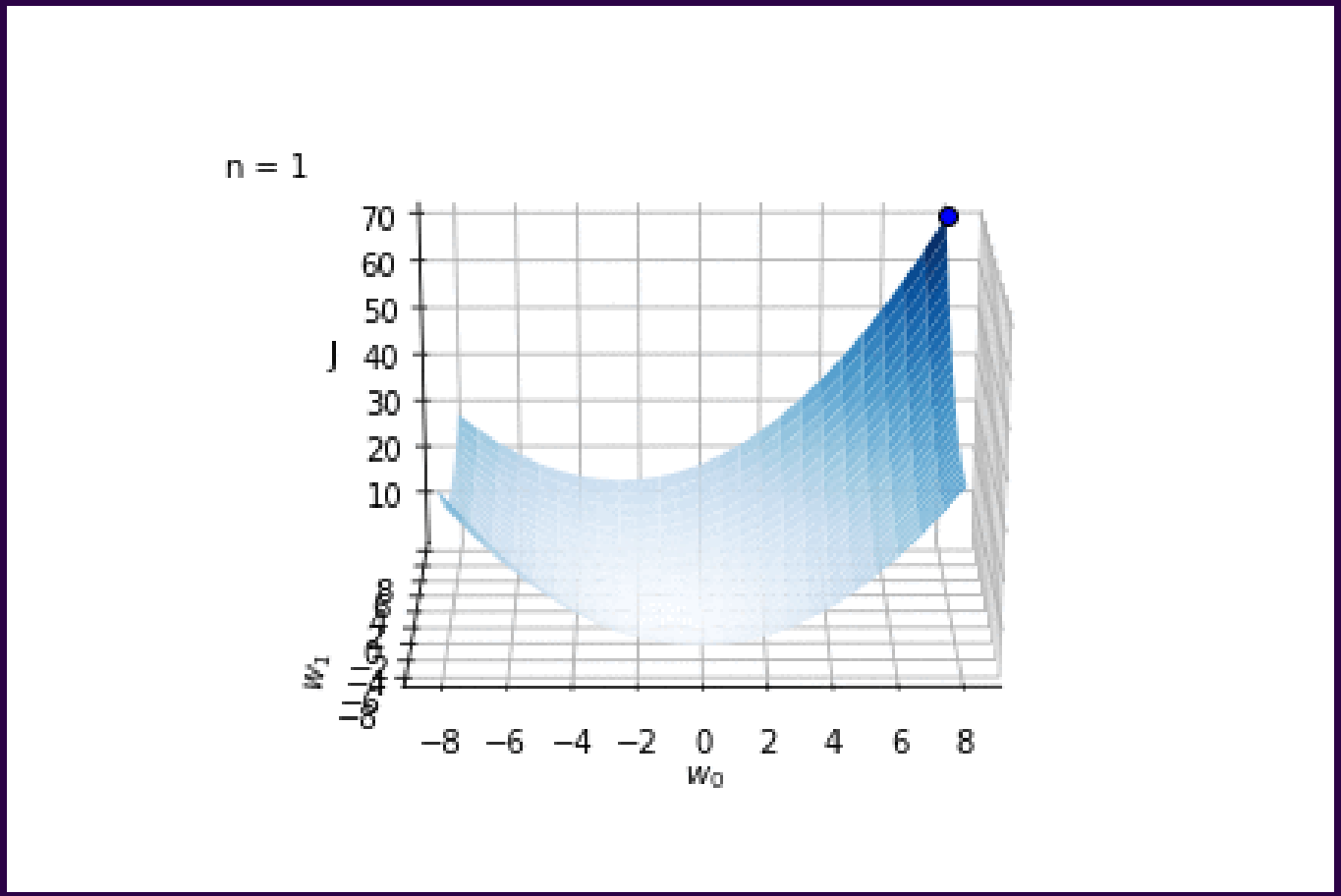
Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.



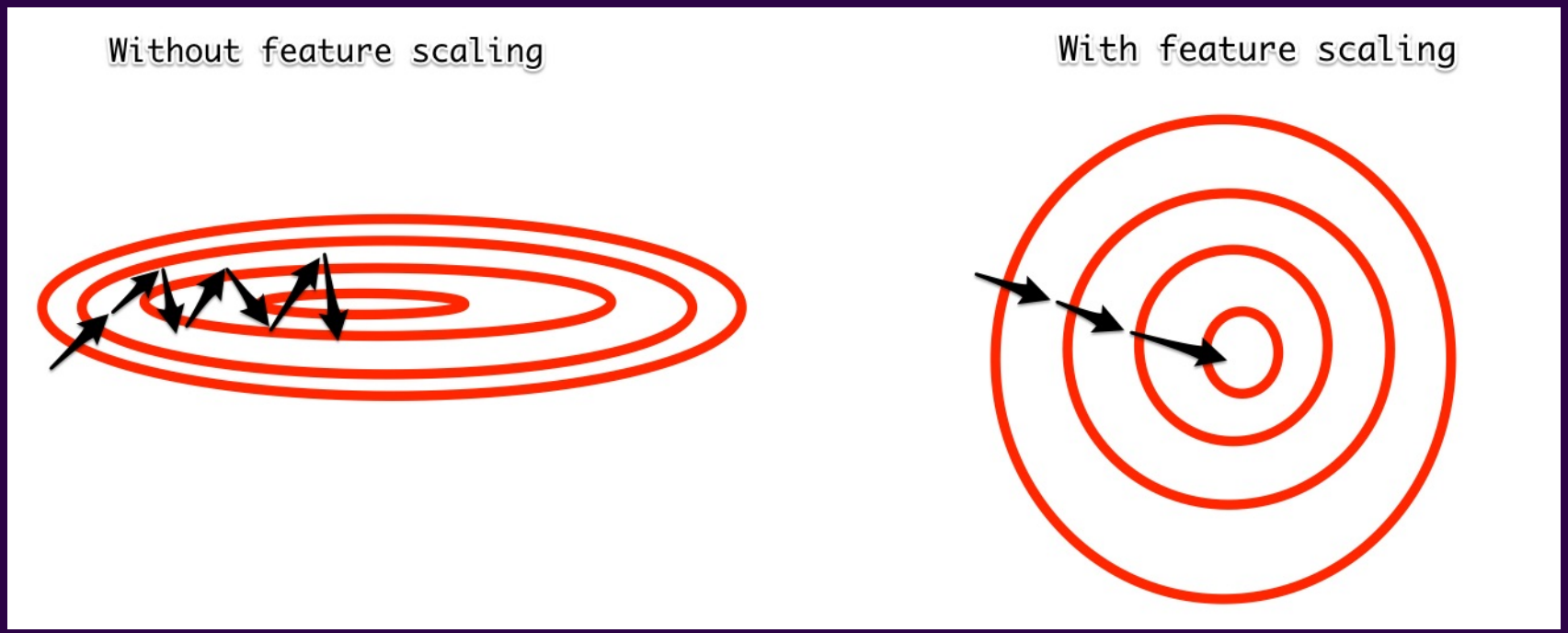
Correct learning rate

Some Animations



Importance of Feature Normalization

- Helps improve the performance of gradient based optimization



Some Gradient Based Algorithms

- Gradient descent
  - Stochastic gradient descent
  - Mini-batch gradient descent
- Gradient descent with momentum
- RMSprop
- Adam optimization algorithm

We have many frameworks that help us use these techniques in a single line of code (Eg: TensorFlow, PyTorch, Caffe, etc).

Thank You!

- Next Class: Linear Classification