The National University of Computer and
Emerging Sciences

Introduction to Machine Learning

Machine Learning for Data Science

Dr. Akhtar Jamil

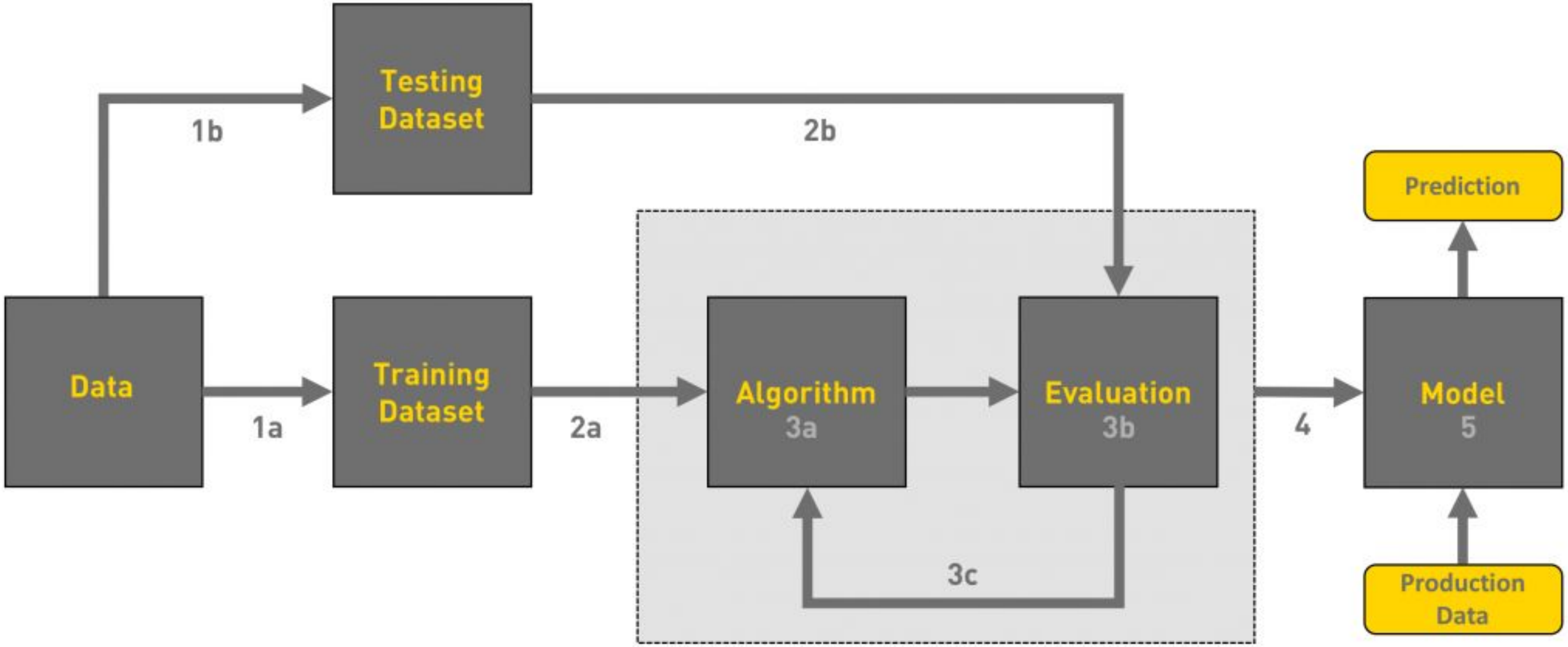
Department of Computer Science

Goals

- Review of Previous Lecture
- Today's Lecture
  - Linear Regression with multiple variables
  - Least Squares

Today’s Lecture

Workflow of ML tasks

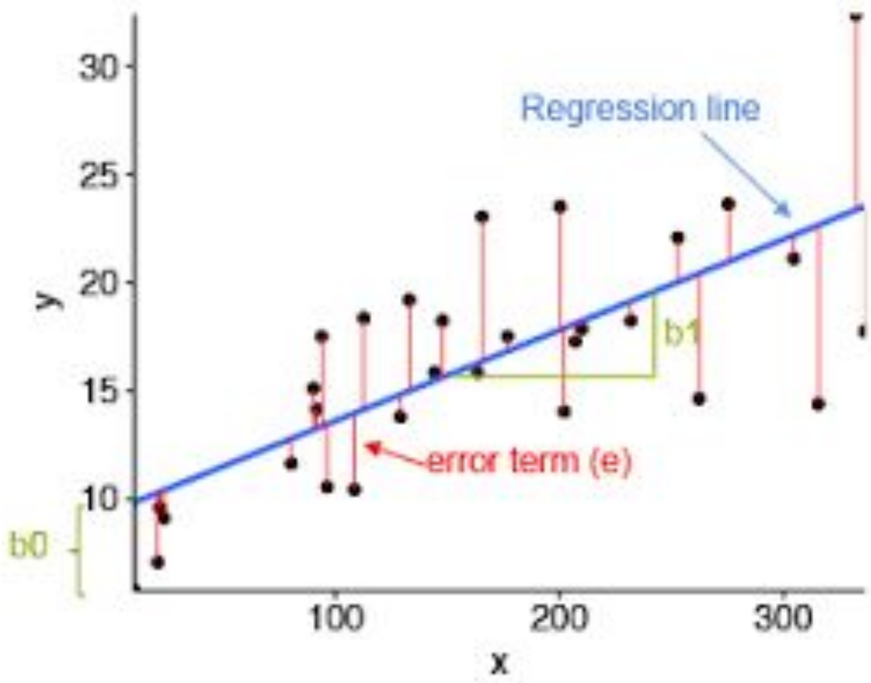


Workflow of ML Problem

- Data Preparation.
- Model Selection and Development.
- Train and Test model
- Deploy your trained model.
- Monitor and Manage models

Prediction

If you know something about X, this knowledge helps you predict something about Y.

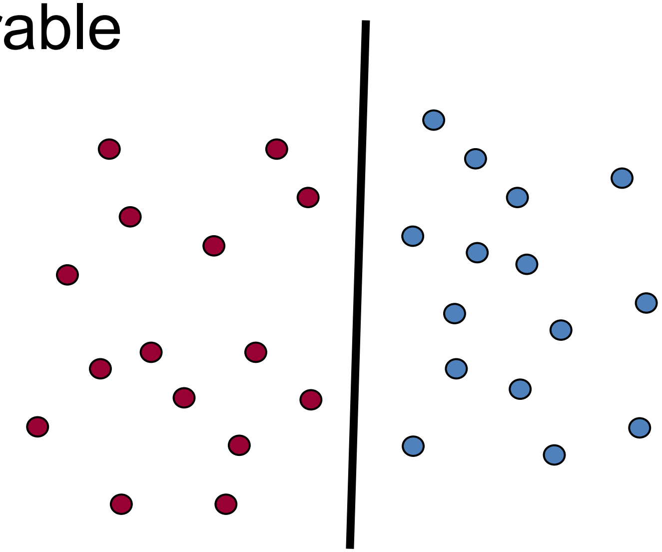


Linear models

An assumption is *linear separability*:

- in 2 dimensions, can separate classes by a line
- in higher dimensions, need hyperplanes

A *linear model* is a model that assumes the data is linearly separable

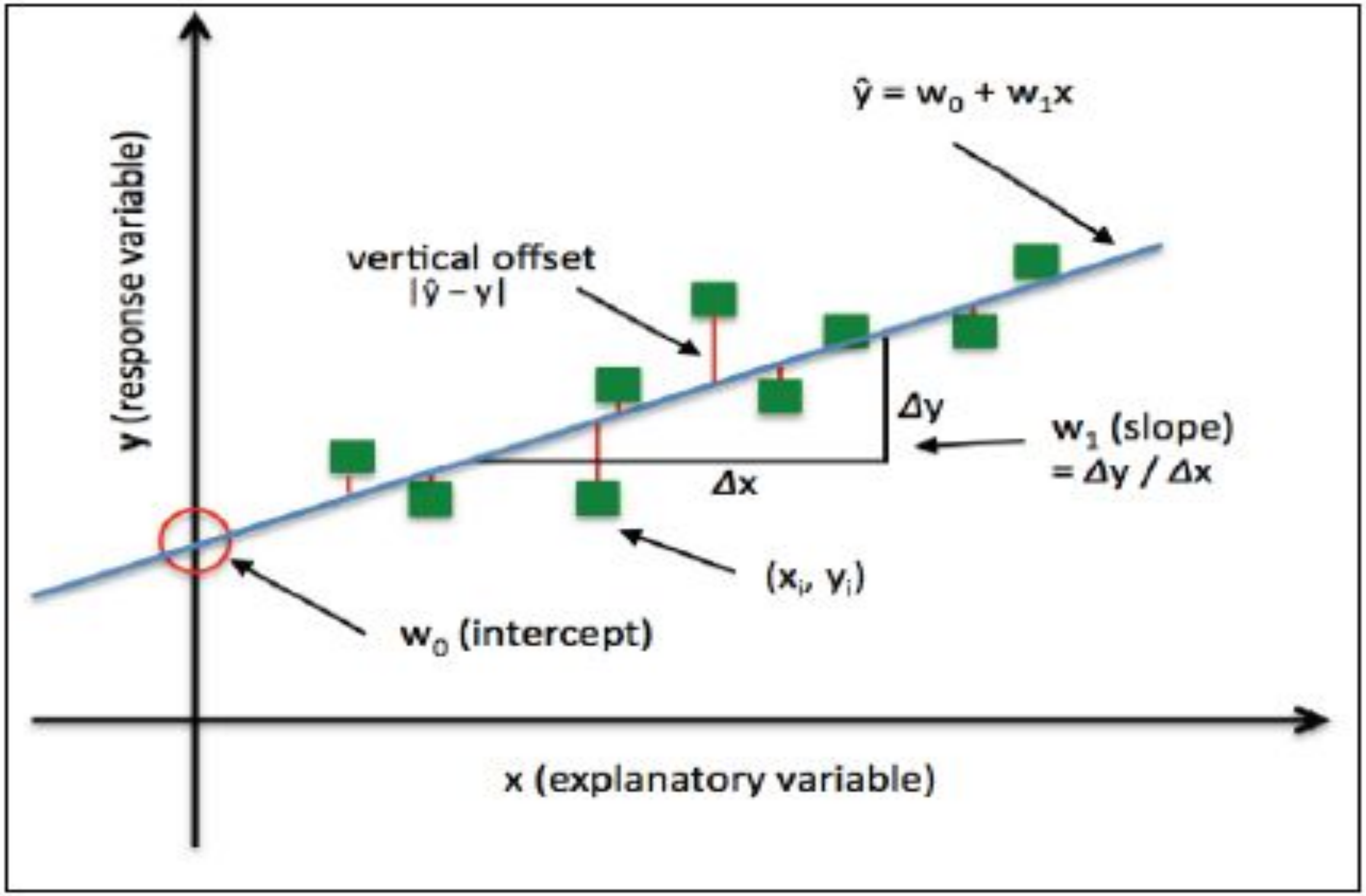


Linear regression

- The goal of simple linear regression (univariate) model is to finds the relation between two variable.
  - A single feature (variable x) and a continuous valued response (target variable y).
  - X is called independent variable (predictor)
  - Y is called the dependent (target or response) variable.

$$y = w_0 + w_1 x$$

Linear regression



Linear models in general

- For leaner model:

$$y = w_0 + w_1 x$$

- These are the parameters we want to learn
- Need to define a criteria to optimize these parameters of the model
  - cost function (objective)
  - Minimize the cost function

Cost function

- The cost function helps find optimal model parameters
  - Best fit line for the data points.
- Searching for these parameters is a minimization problem
  - Model with minimum error between the predicted value and the actual value.
- One such cost function is:
  - Mean Squared Error(MSE):

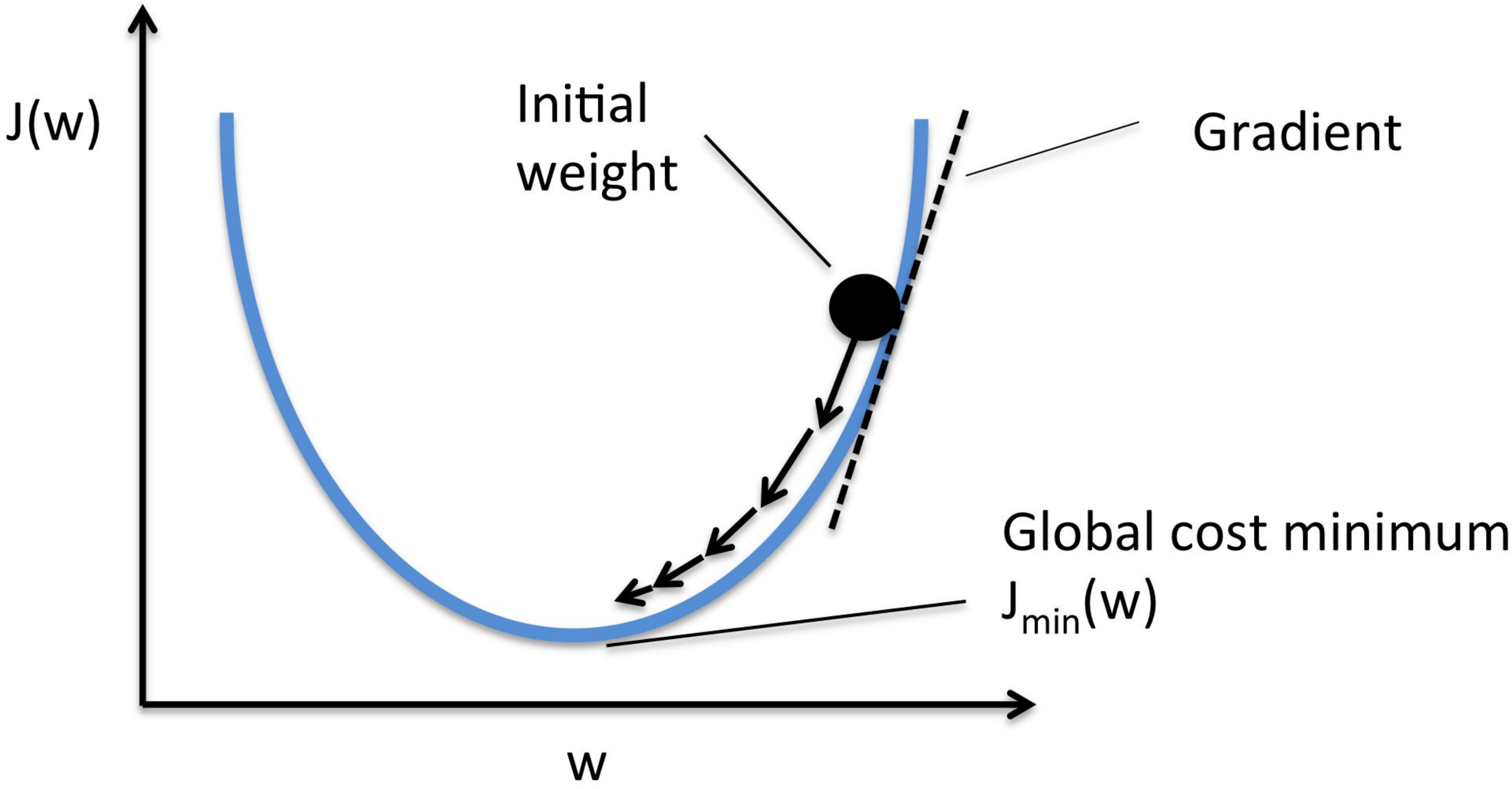
$$J(w) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- $\hat{y}_i$: is predicted label
- y_i: Original label

Gradient Descent

- Gradient descent is an optimization algorithm
- It helps for searching for the optimal model parameters
- Update parameters according to the gradient values.
- A gradient measures how much the output of a function changes if you change the parameter values.

Gradient Descent



Surrogate loss functions

0/1 loss: $$I(y,\hat{y}) = \begin{cases} 1 & y \neq \hat{y} \\ 0 & y = \hat{y} \end{cases}$$

Hinge

: $$l(y,y') = \max(0, 1 - yy')$$

Exponential

: $$l(y,y') = \exp(-yy')$$

Squared loss: $$l(y,y') = (y - y')^2$$

Today’s Lecture

Linear regression with multiple features

- The idea of linear regression can be extended for multiple variables.
  - A set of multi features (X) will be input to the model
  - Y is continuous valued response (target variable y).

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 \dots + w_n x_n$$

$$\hat{y} = h_w(x) = W . x$$

Where x $= x_1, x_2, \dots x_n$ and W = $w_0, w_1, .. w_n$

Linear regression with multiple features

- $$\hat{y}_1 = w_0 + w_1 x_1$$
  $$\hat{y}_2 = w_0 + w_1 x_2$$
  $$\hat{y}_3 = w_0 + w_1 x_3$$

  $$\hat{y}_n = w_0 + w_1 x_n$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ . \\ . \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \\ . \\ . \\ w_0 + w_1 x_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ & . \\ & . \\ 1 & x_n \end{bmatrix} \text{x} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Cost function

- The cost function helps find optimal model parameters
  - Best fit line for the data points.
- Searching for these parameters is a minimization problem
  - Model with minimum error between the predicted value and the actual value.
- One such cost function is:
  - Mean Squared Error(MSE):

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- $\hat{y}_i$: is predicted label
- y_i: Original label

Optimization: Gradient Descent

- Initialize **w** (e.g., randomly)
- Update the values of ***w*** based on the gradient:

$$w_i = w_i - \lambda \frac{\partial J}{\partial w_i}$$

- *Where λ is learning rate*
- *To find w_0 take derivate of the function with respect to it:*

$$w_0 = w_0 - \lambda \frac{\partial J}{\partial w_0}$$

Gradient Descent

- To find w_1 take derivate of the function with respect to it:

$$w_1 = w_1 - \lambda \frac{\partial J}{\partial w_1}$$

- After solving for the two parameters we get:

$$\frac{\partial J}{\partial w_1} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \hat{y_i})x_i$$

$$\frac{\partial J}{\partial w_0} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \hat{y_i})$$

Optimization: Gradient Descent

- To find w_1 take derivate of the function with respect to it:

$$w_1 = w_1 - \lambda \frac{\partial J}{\partial w_1}$$

$$w_n = w_n - \lambda \frac{\partial J}{\partial w_n}$$

Optimization: Gradient Descent

- Variants of Gradient Descent are available for optimization in ML
  - Batch Gradient Descent
  - Stochastic Gradient Descent(SGD)
  - Mini-batch Gradient Descent
- **Batch Gradient Descent**
  - Updates the weight vector over the full training data
  - It is very slow on very large training data.

$$w_1 = w_1 - \lambda \frac{\partial J}{\partial w_1}$$

$$w_1 = w_1 - \lambda \frac{2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_i$$

Optimization: Gradient Descent

- **Stochastic Gradient Descent(SGD)**
  – It updates the parameters for each training data, according to its own gradients:

$$w_1 = w_1 - \lambda \frac{\partial J}{\partial w_1}$$

$$w_1 = w_1 - \lambda\,(y_i - \hat{y}_i)x_i$$

Optimization: Gradient Descent

- **Mini-batch Gradient Descent**
  - It computes the gradients on small random sets of instances called mini-batches.
  - It has shown better performance than SGD
  - More robust stable than SGD

The Normal Equation

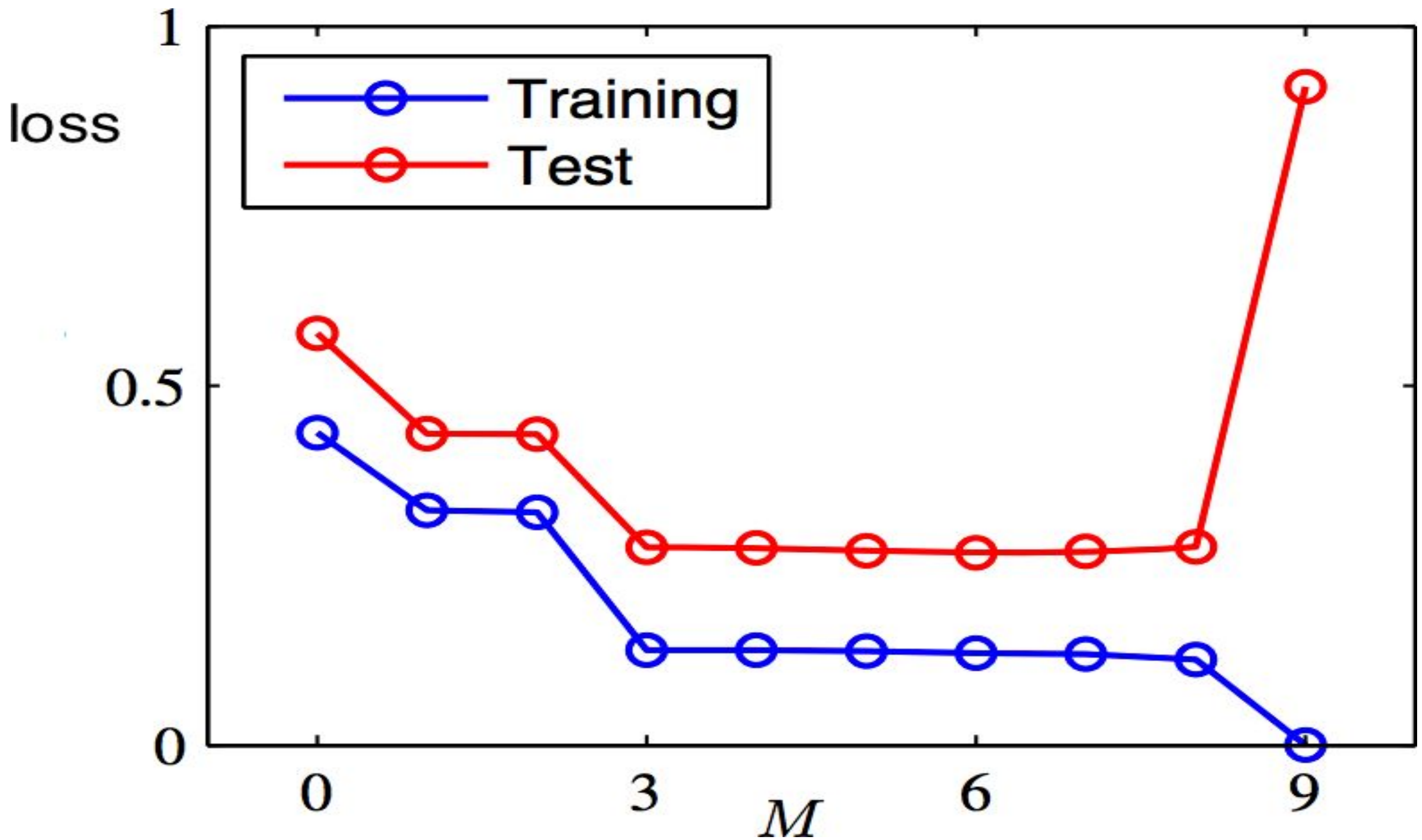
- Find the optimal values of the parameters directly
- The value of **W** that minimizes the cost function
  - closed-form solution
- This is called the *Normal Equation.*

$$W_i = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

Generalization

- The model's ability to adapt properly to new and previously unseen data.
- We expect a model to perform well on both training and test data sets.
- What if model shows high accuracy on Training data and low accuracy on test data?
  - Not generalized well

Generalization



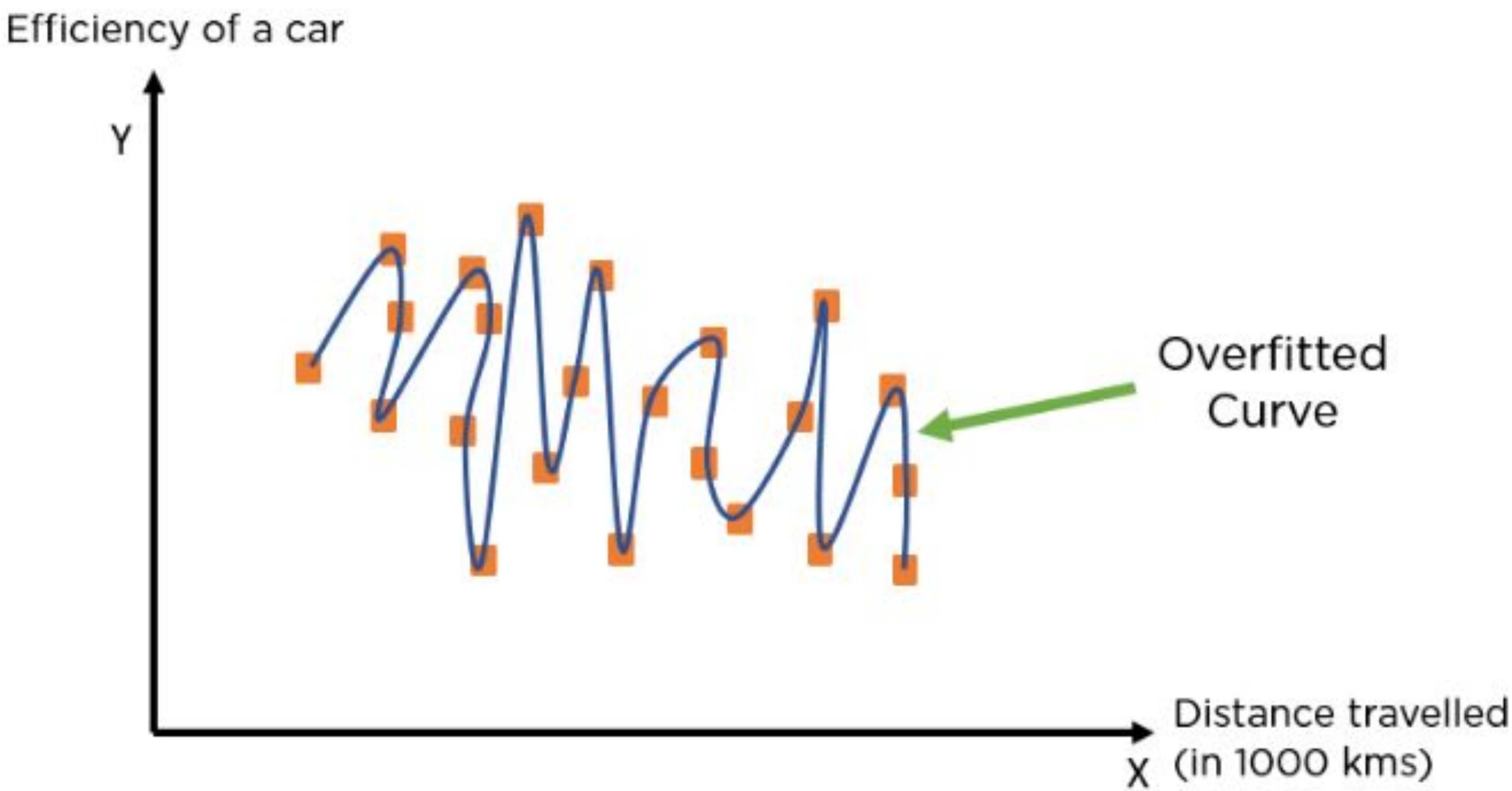
Overfitting

- Overfitting refers to a model that learns the training data too well but shows low accuracy on test data
- Overfitting happens when a model learns the detail and no
- ise in the training data to the extent that it negatively impacts the performance of the model on new data.
- Result in poor performance of classifier

Overfitting

- Noise in the data
- The model has a high variance
- Small size of the training dataset
- The model is too complex

Overfitting



Underfitting

- The model in unable to learn the training data well.
  - Low accuracy on training data.
  - May not generalize well on the new data
- Underfitting occurs due to high bias and low variance of model.
- The size of the training dataset used is not enough
- The model is too simple
- Not enough iterations

Underfitting vs Overfitting

Feature Scaling

- Feature scaling in machine learning is an important pre-processing steps
  - Affect performance of the model
- The difference in range of values of features may cause one feature to dominate other.
- The most commonly used techniques:
  - Normalization
  - Standardization.

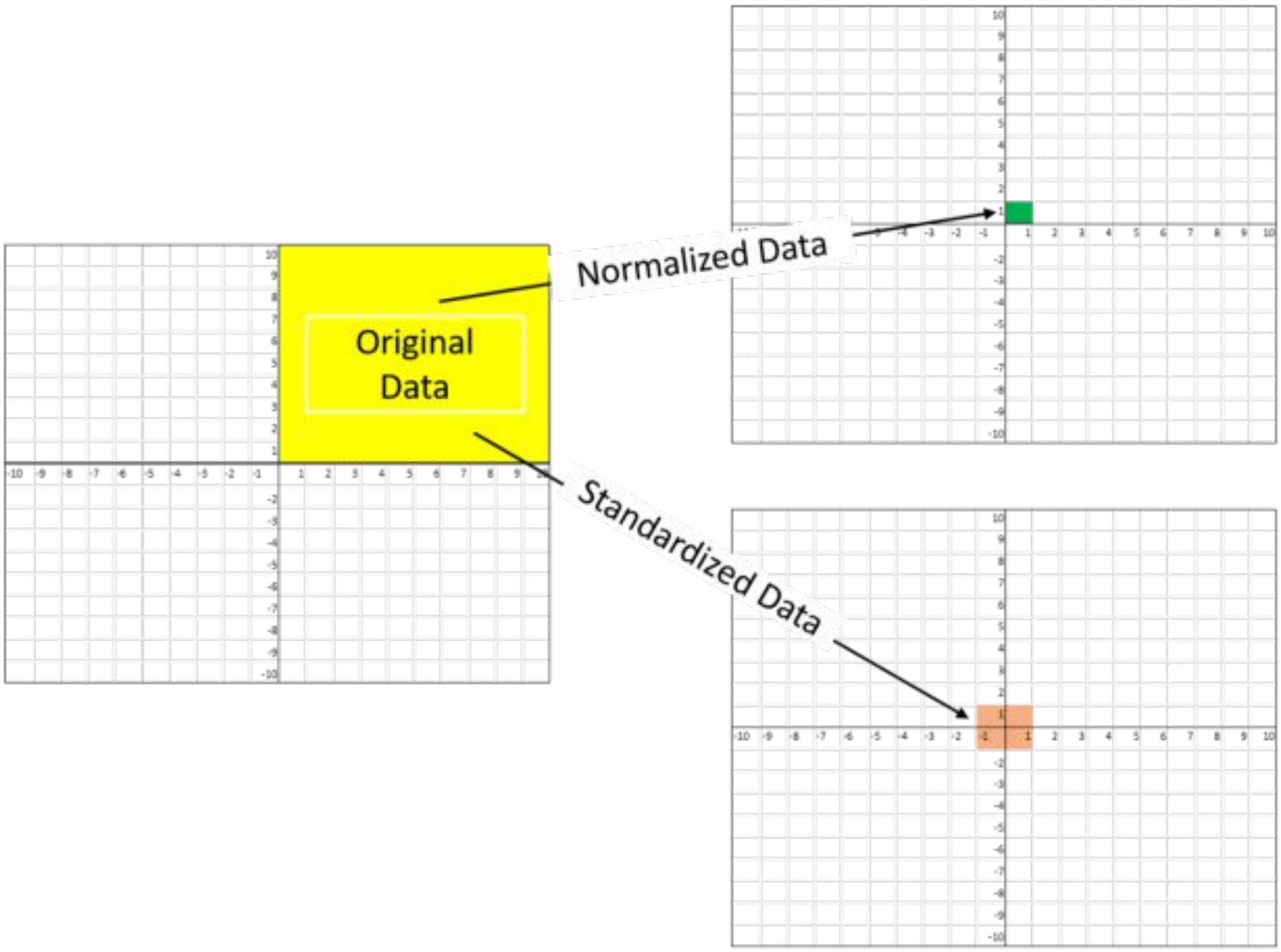
Feature Scaling

- **Normalization:** The values of each feature are bound between two numbers, e.g. [0,1] or [-1,1].
  - Min-Max Normalization

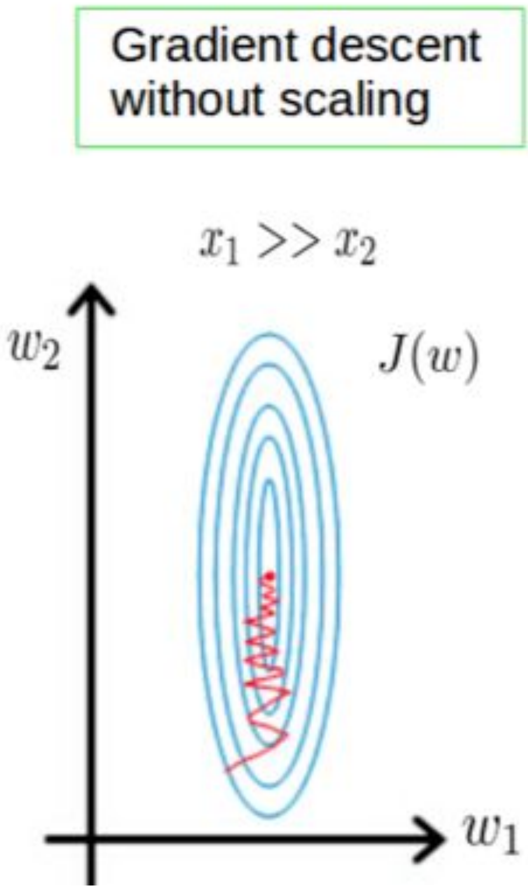
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

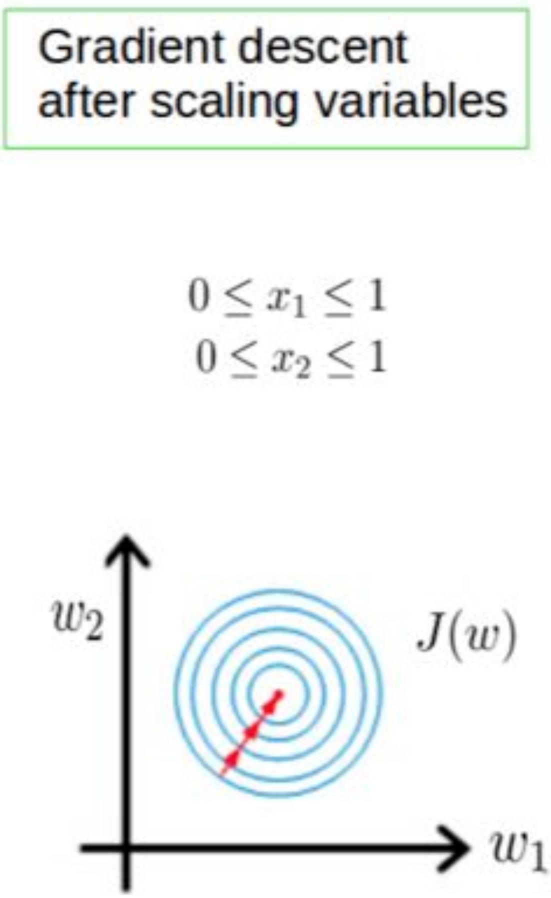
- **Standardization** transforms the data to have zero mean and a variance of 1
  - Make our data **unitless**.

$$x_{new} = \frac{x - \mu}{\sigma}$$


Original
Data
Normalized Data
Standardized Data

Feature Scaling





Reference

- Chapter 5, Deep Learning MIT Press 2016, Ian Goodfellow
- Chapter 3 Pattern Recognition and Machine Learning, Christopher M. Bishop
- Some graphics from the internet:
  - https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Thank You ☺