GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

depa

Introduction to Supervised Learning:

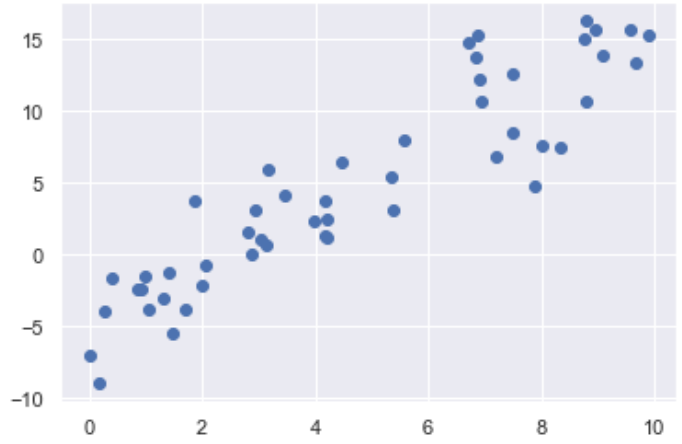
Linear Regression

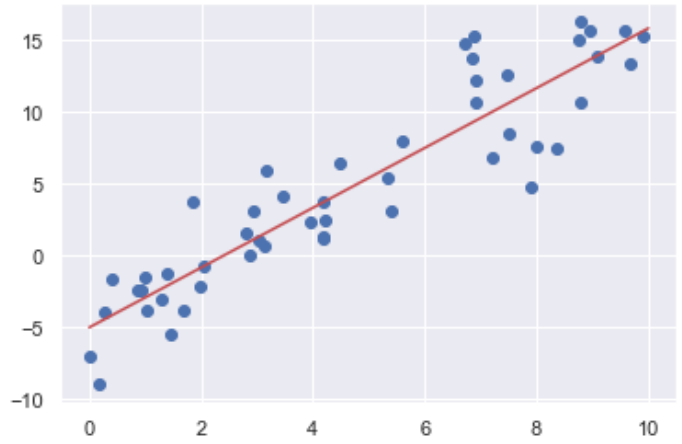
Thanakorn Thaminkaew

Data Scientist

Original materials by Dr. Duangjai Jitkongchuen, Papoj Thamjaroenporn, and Patipan Prasertsom

Regression





- Just like many other supervised learning tasks, regression task attempt to unravel the relationship between input features and target output.
- The model can then predict a numeral value of a target output when given new input data.
- For linear regression, a linear equation is used as model to be represent said relationships by attempting to find the most appropriate weights for the equation

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

Linear function review

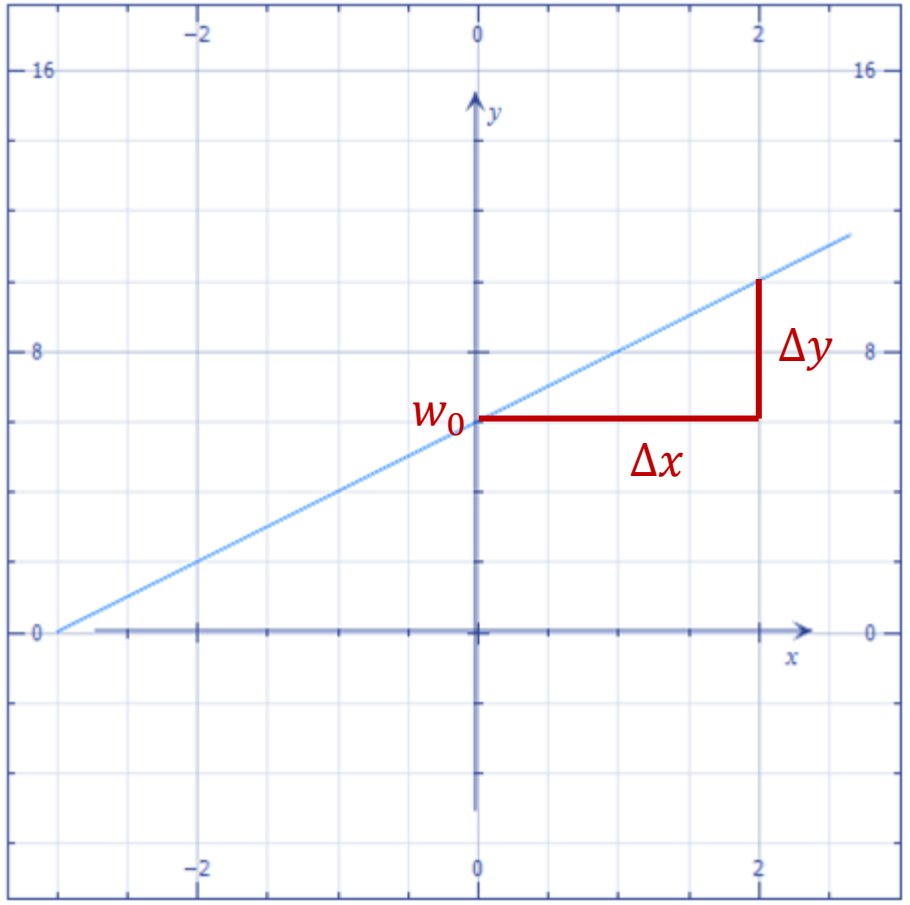
- Recall that in the case of 1 variable, a linear equation has the form of

$$y = w_0 + w_1 x_1$$

- Here, w_1 represent the gradient (slope) of the line, which is the rate of change in y with respect to the change in x

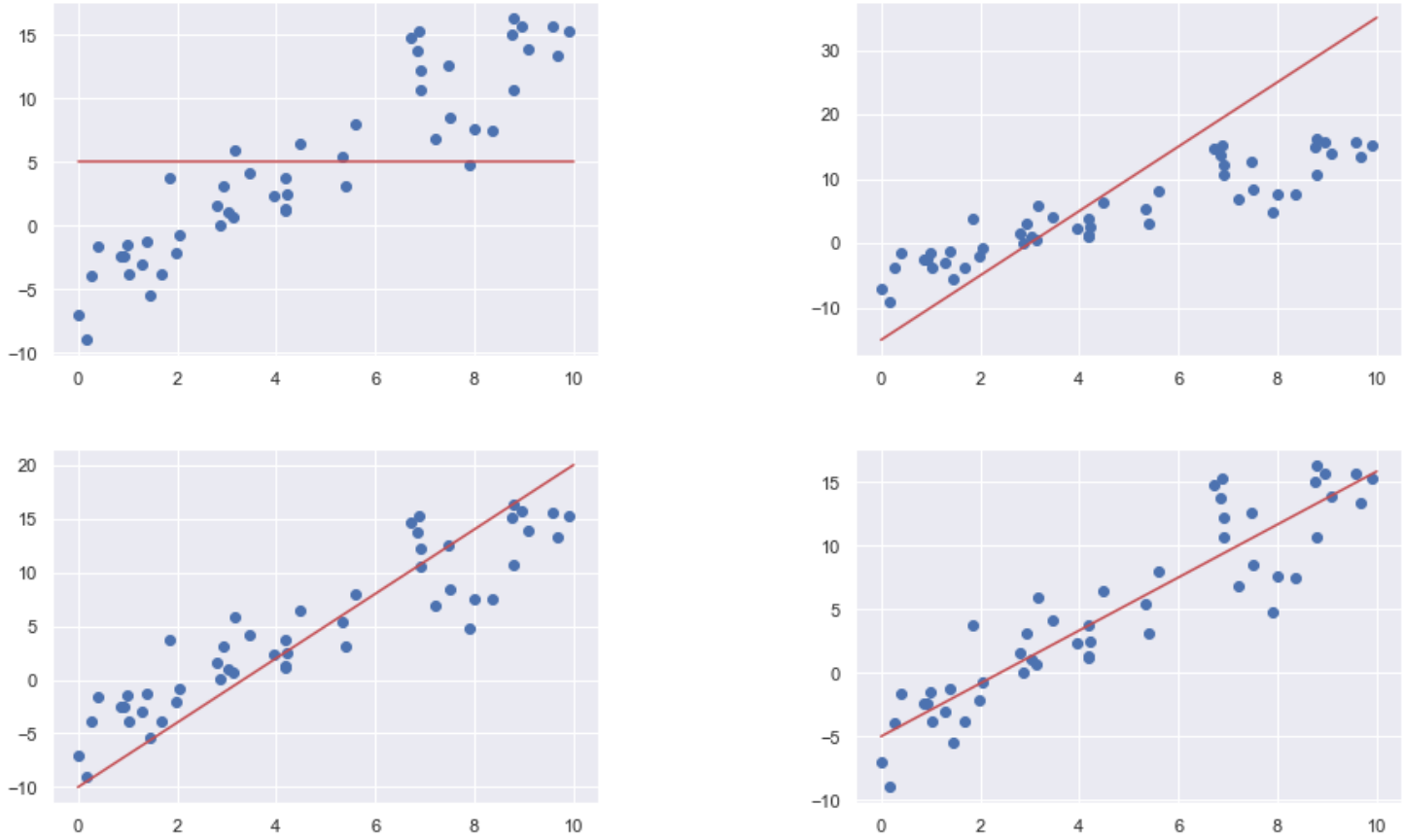
$$w_1 = \frac{\Delta y}{\Delta x}$$

- w_0 represent the y-intercept (bias)



Figures from Adrien1018 (https://commons.wikimedia.org/wiki/File:Linear_equation.png)

Multiple Curve Fitting Options



- Which line fits the best among these 4 lines? How do we compare them?

Error

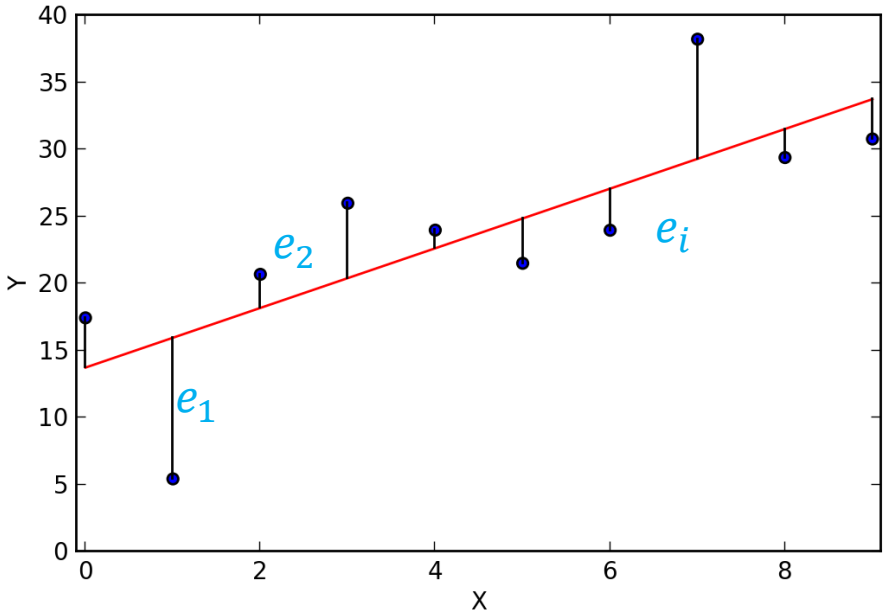
- For each data point, the error of a prediction is

$$e^{(i)} = \hat{y}_i^{(i)} - y^{(i)}$$

- In order to evaluate the overall error of a prediction model, Mean Square Error (MSE) is utilized.

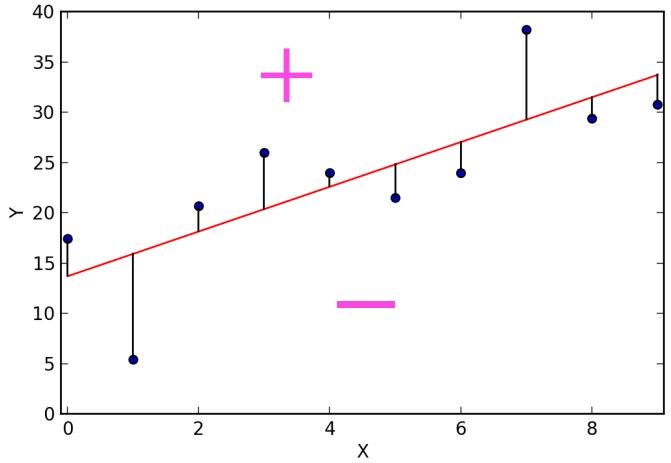
$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}_i^{(i)} - y^{(i)}\right)^2$$

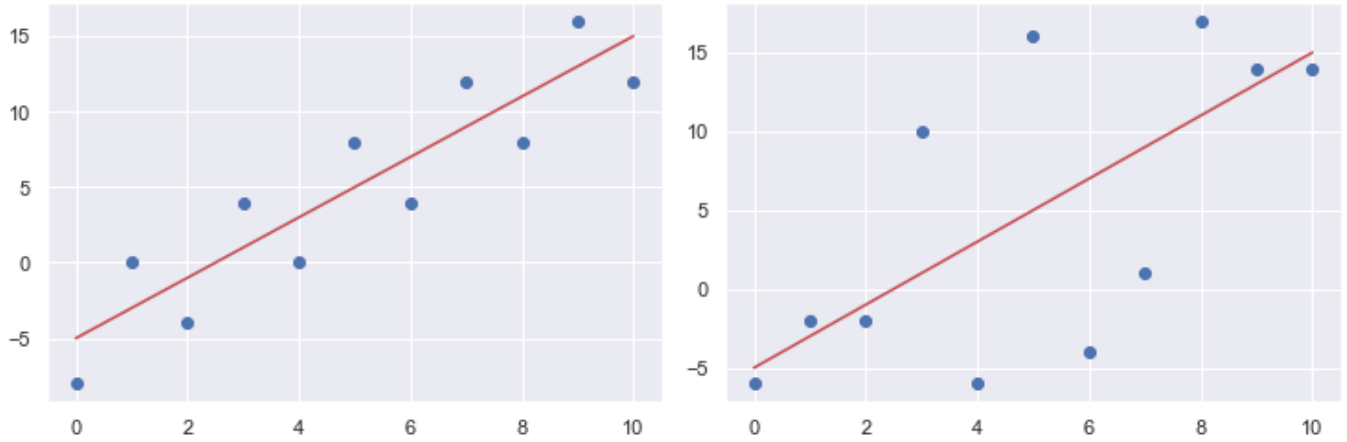
- If we want the overall error of the model, why don't we just sum/average the errors of all points?



Figures from Thomas.haslwanter (https://commons.wikimedia.org/wiki/File:Residuals_for_Linear_Regression_Fit.png)

Why MSE?





- The sum of errors will cancel each other!
- However, if we only want the errors to be positive,

why don't we simply use absolute value of errors?

- Which one of the line fits better?
- In addition to making errors positive, MSE does not treat an increase in errors linearly in scale
  - Specifically, by squaring the error, it heavily punish the points that are further from the predicted values

Why MSE?

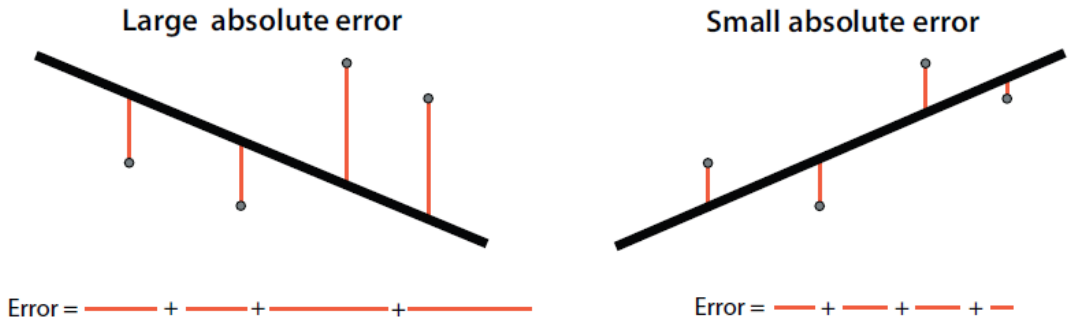


Figure 3.18 The absolute error is the sum of the vertical distances from the points to the line. Note that the absolute error is large for the bad model on the left and small for the good model on the right.

MAE

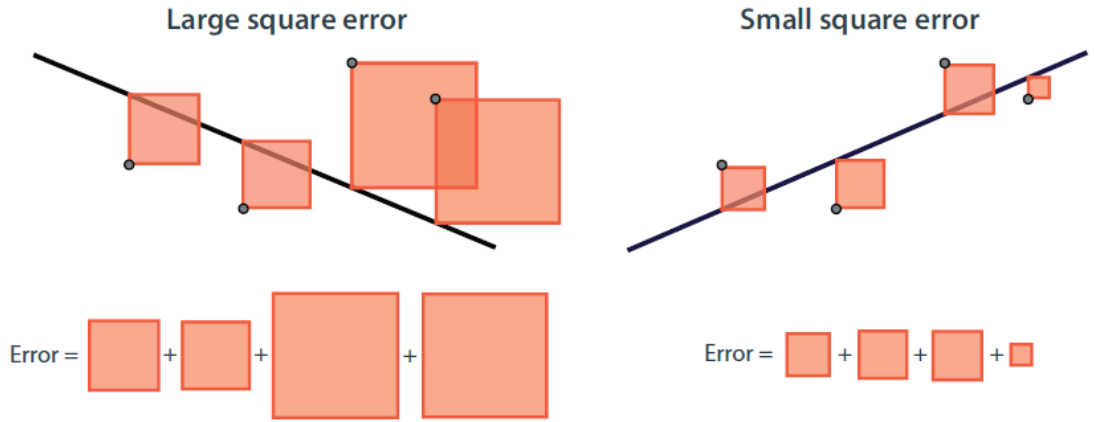
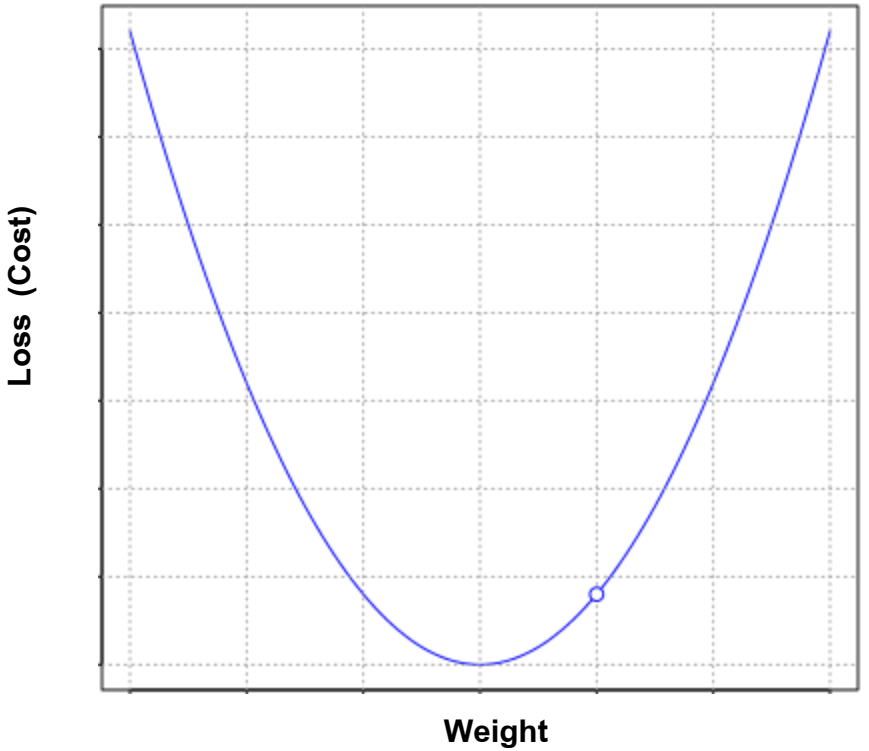


Figure 3.19 The square error is the sum of the squares of the vertical distances from the points to the line. Note that the square error is large for the bad model on the left and small for the good model on the right.

MSE

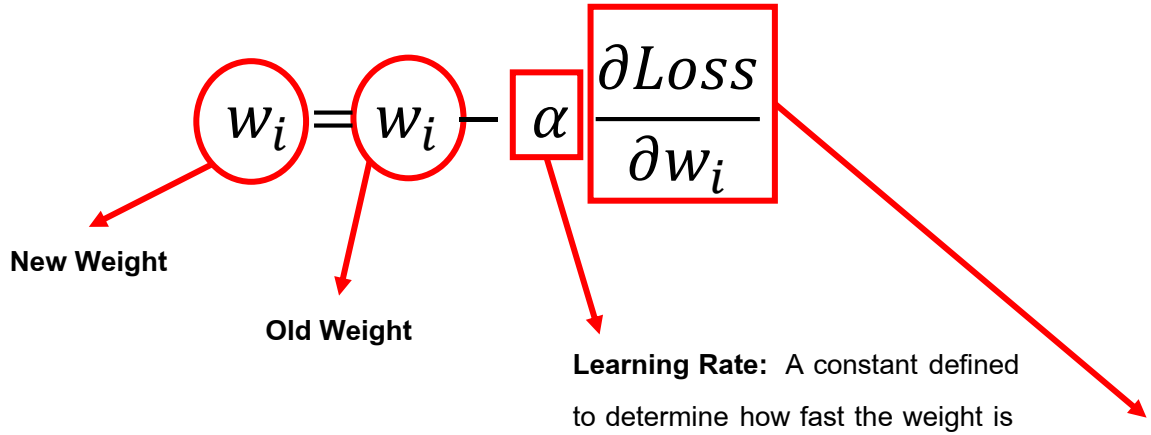
Adjusting Weights

- A goal of a linear regression (and most supervised machine learning models) is to find the mathematical model that best represent the relationship between input features and target output.
- In order to do so, we define a "**loss function**" (or cost function) that we're trying to minimize.
- For linear regression, a commonly used loss function is **MSE** (with modified constant) i.e. $\text{Loss} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y_i}^{(i)} - y^{(i)}\right)^2$
  - This means the model will adjust its weights to minimize overall error
- How would the algorithm know which way to adjust the weights so the model can reach the minimum error?



Solving Regression Problem Numerically: Gradient descent

- The *gradient* (slope) of the cost function with respect to each weight at each given point informs the algorithm of the direction it should adjust the weight.
- For each iteration, the parameter is adjusted toward minimizing the loss function



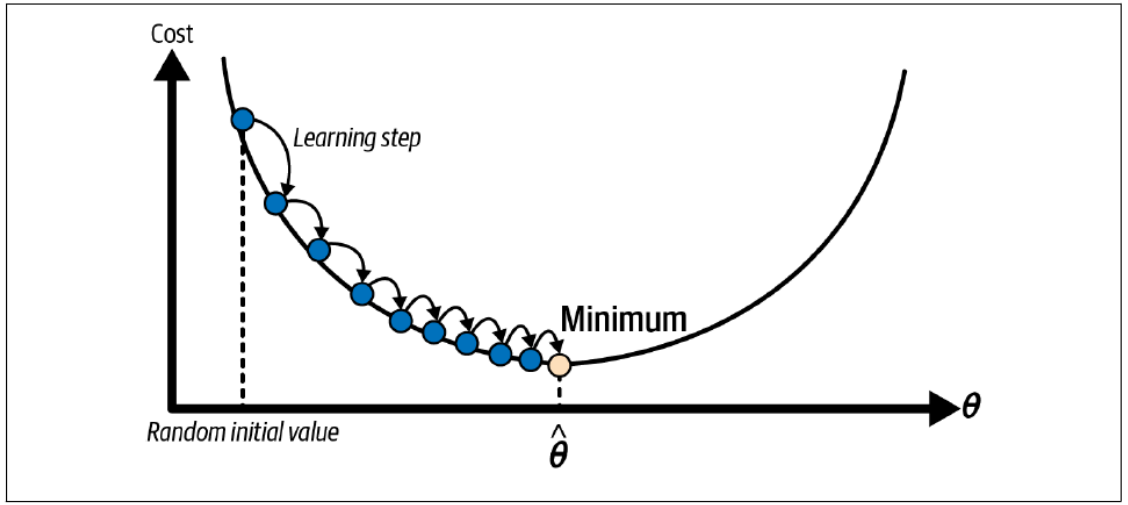


Figure 4-3. In this depiction of gradient descent, the model parameters are initialized randomly and get tweaked repeatedly to minimize the cost function; the learning step size is proportional to the slope of the cost function, so the steps gradually get smaller as the cost approaches the minimum

Solving Regression Problem Numerically: Gradient descent

- When do we stop adjusting the weight?
  - The loss function improvement is below threshold (converged)
  - Reached an iteration limit.
- In machine learning library, parameters such as learning rate and iteration count are configurable
- Smaller learning rate can result in model training taking a long time
- Large learning rate can result in oscillating performance

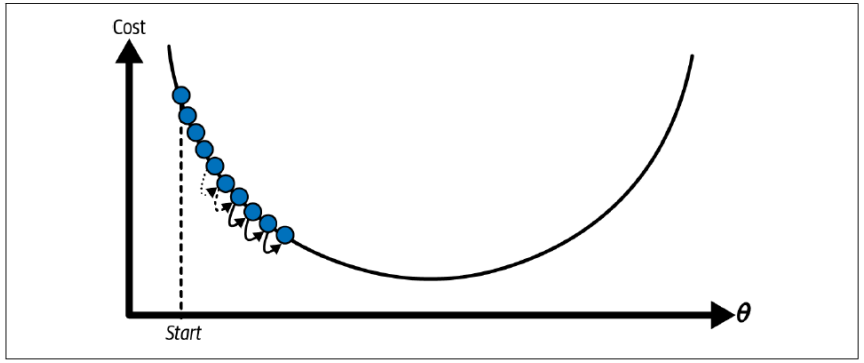


Figure 4-4. Learning rate too small

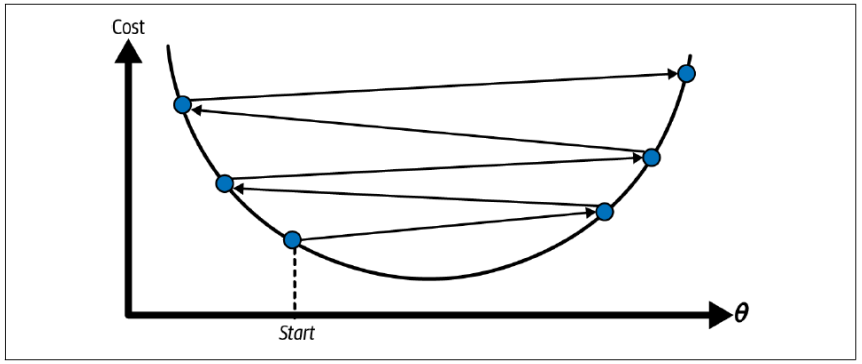


Figure 4-5. Learning rate too high

Multiple Linear Regression (MLR)

- While the examples of graphs presented so far are showing only one feature, the gradient descent algorithm applies updates to all the weights simultaneously inside the equation in each iteration.

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

$$w_i = w_i - \alpha \frac{\partial Loss}{\partial w_i}$$

- In many cases, there are multiple relevant features that can (and should) be used to train a regression model.
- However, having multiple input features introduces more factors that may affect the performance such as
  - Variety of features' scales
  - The dependencies among features

Feature Scaling

- If the features of are not on similar scale, the process of gradient descent might take a long time before reaching convergence as the value of the cost function might end up oscillating.
- One of the common practices is to get features to be approximately within the range of -1 and 1

  $\Rightarrow$ In order to do so, we can first apply **mean normalization** (replacing x_i with $x_i - \mu_i$) to center data at 0

  $\Rightarrow$ Then we can perform normalization techniques like **mean normalization** or **standardization** to scale down the size of feature.
- In other words,

Mean normalization $$x'_i = \frac{x_i - \overline{x_i}}{\max(x_i) - \min(x_i)}$$

Standardization $$x'_i = \frac{x_i - \overline{x_i}}{S_i}$$

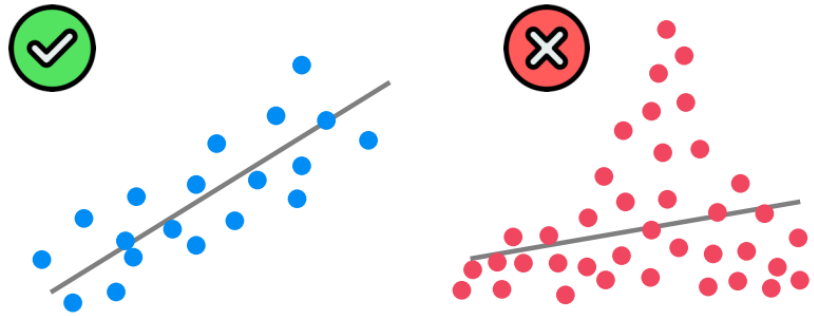
- There are functions available in the popular machine learning libraries to accomplish this.

Linear Regression Assumptions

- **L**inearity – The relationship between input features and output should be linear
  - If the relationship is non-linear, higher degree model or non-linear transformation should be applied.
- **I**ndependence – There should be no correlation among error terms of data points (no autocorrelation)
  - If there is autocorrelation, using time series models might be more appropriate.
- **N**ormality – The error terms should be normally distributed
  - If the distribution is not normal, then there are abnormal data points. Handling outliers and perform non-linear transformation might help.
- **E**qual Variance
  - Generally non-constant variance occurs when there are outliers or extreme leverage value. Non-linear transformation might help.

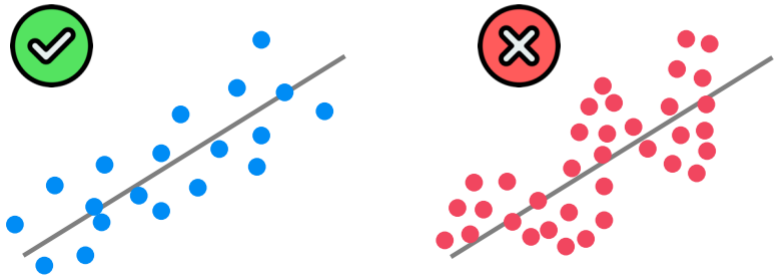
Linear Regression Assumptions

Linearity

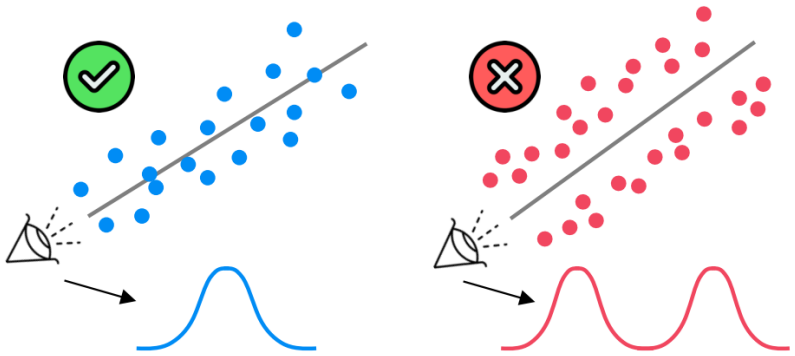


Independence

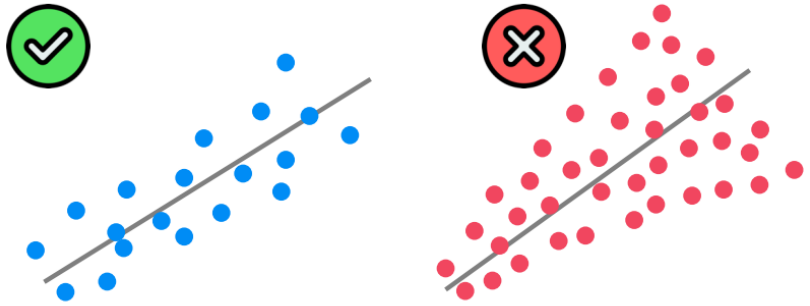
(of observations. Includes “no autocorrelation”)



Normality



Equal Variance



https://sds-platform-private.s3-us-east-2.amazonaws.com/uploads/SuperDataScience-Poster-Assumptions-of-Linear-Regression.pdf

Note on Correlations

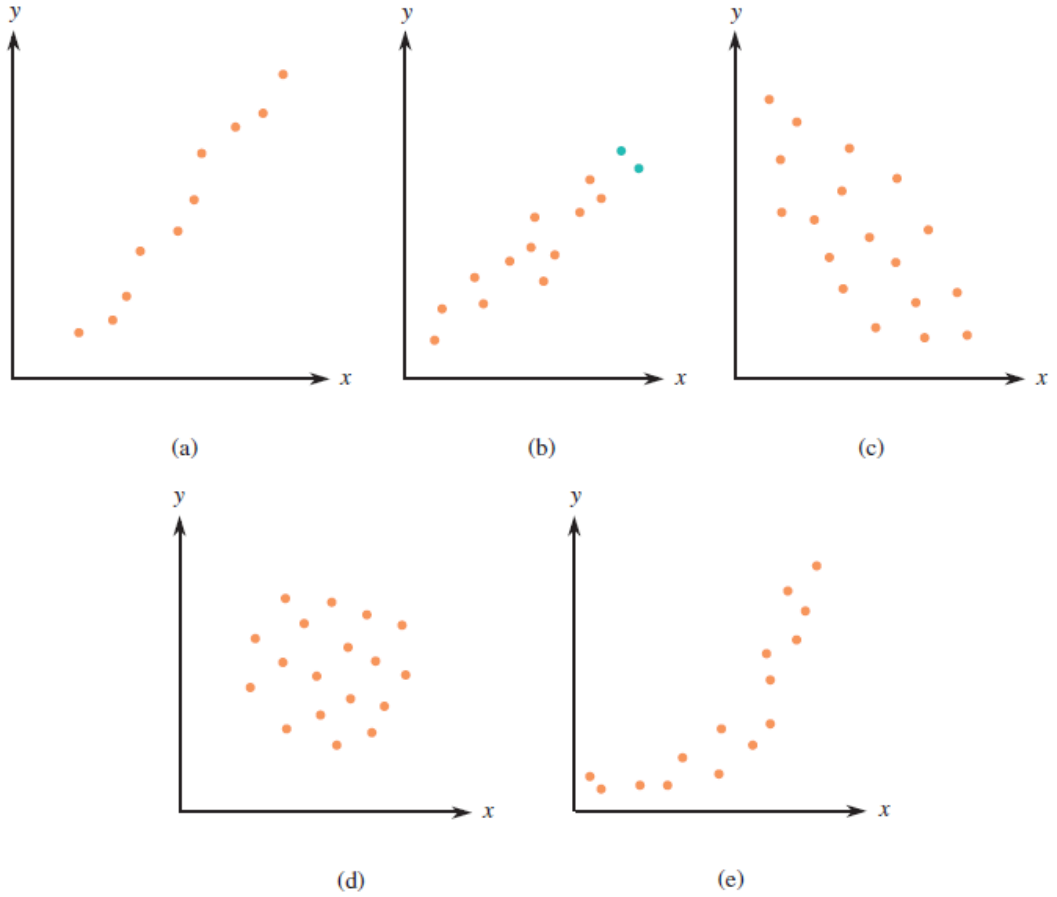


FIGURE 5.1
Scatterplots illustrating various types of relationships: (a) positive linear relationship; (b) another positive linear relationship; (c) negative linear relationship; (d) no relationship; (e) curved relationship.

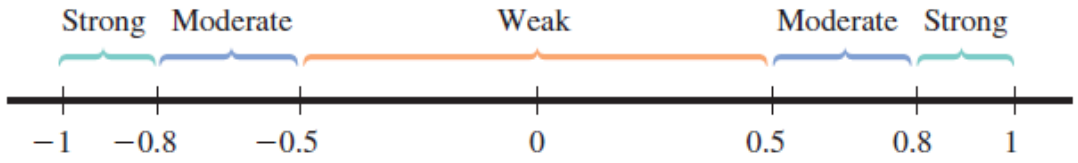


FIGURE 5.4
Describing the strength of a linear relationship.

- Correlation coefficient has a value between -1 and 1
  - The closer the magnitude is to 1, the stronger the correlation
- Pandas has corr() function that we can use to find correlation among features/with target
  - Usage: df.corr()

Figures from Introduction to Statistics & Data Analysis by Peck et. al

Coefficient of determination (r^2)

- Coefficient of determination is a measure of the proportion of variability in the target variable that can be explained by the linear relationship between the input features and the target.

$$r^2 = 1 - \frac{SSResid}{SSTotal}$$

Unexplainable Variance

Explainable Variance

- Where $SSResid = \sum(y - \hat{y})^2$ and $SSTotal = \sum(y - \bar{y})^2$
- Can be used to measure how well the regression equation is doing as it tell us how much better our predictive model is performing to just using mean as predictor.
- r^2 value has a maximum value of 1. The higher the value, the more useful the model

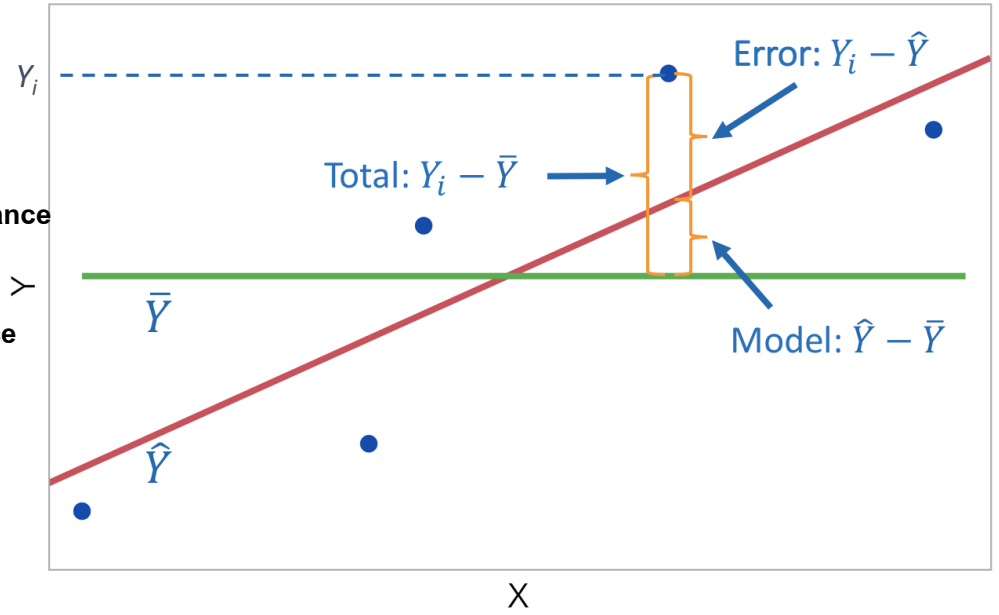


Figure from https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-regression/interpreting-regression-results.html

Content from Introduction to Statistics & Data Analysis by Peck et. al and KMUTT Big Data Experience Center's lecture on Linear Regression

Machine Learning Library: Scikit-learn (sklearn)

- Scikit-learn is one of the most popular open-source machine learning library for python.
- Large user base globally and constant updates for a (approximately) 3-month cycle.
- Provides a large variety of machine learning models and tools including but not limited to.
  - Supervised learning models such as regression, tree-based models, ensemble models, neural network, etc.
  - Unsupervised learning models such as clustering models, GMM, density estimation, PCA, etc.
  - Model selection, hyperparameters tuning, and evaluation tools such as grid search, cross-validation, various evaluation metrics, etc.

Python Tutorial

- Colab!
- 2.1 Linear Regression (Salary_Data)
- 2.2 Linear Regression (foodtruck)

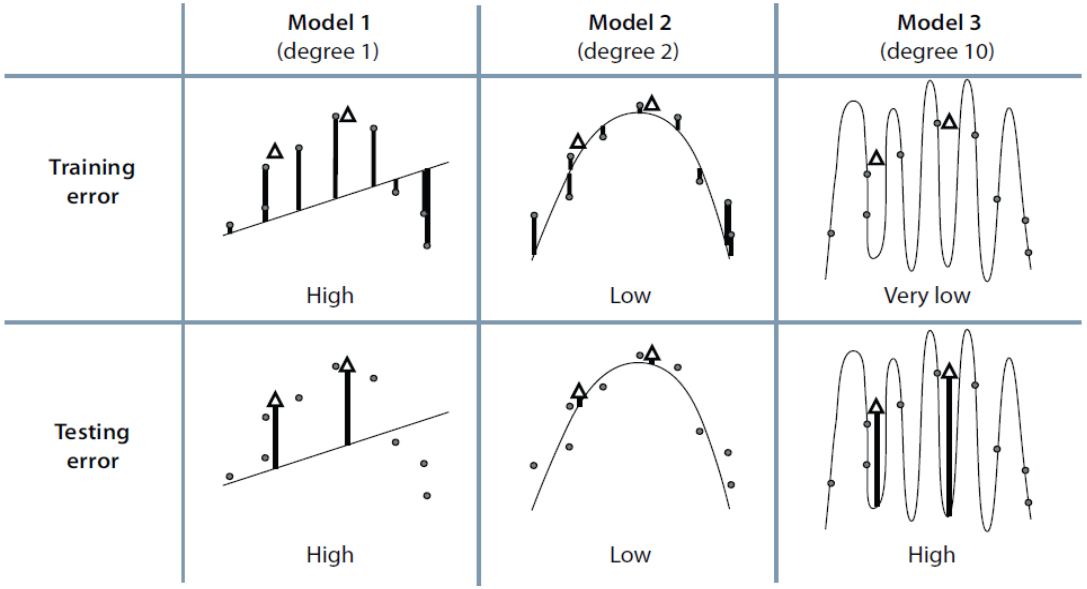
Recap: Overfitting vs. Underfitting

Overfitting

Overfitting occurs when the trained model adjust its parameters to fit the training data too well and became unable to generalize well. This behavior is observed when the model performs well with during the development with training data but performs poorly on the unseen data.

Underfitting

Underfitting happens when the trained model is unable to adjust itself to sufficiently fit the training data, resulting in poor performances overall. This behavior is observed when the model is unable to perform well even during the development with training data.



Dealing with Overfitting: Regularization

- There are many causes of overfitting ranging from the presents of outliers, the lack of data, to the complexity of the model itself being too high, making it fits the training data *too well*.
- **Regularization** is one of the methods used to address overfitting. It penalizes the model with large coefficients and causes them to shrink during optimization process. This is done by introducing penalty terms into the loss function.

$$Loss = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}_i^{(i)} - y^{(i)}\right)^2 + \lambda \sum_{i=1}^{n} w_i^2$$

Example of penalty terms

- Small coefficients will help reduce the impact of less relevant input features
- The regularization constant λ determines the strength of the penalty. It is a tunable hyperparameter of the model.
  - The higher it is, the more penalty imposed. This can help the model becomes more generalizable.
  - Too large regularization constant might lead to underfitting.

Dealing with Overfitting: Ridge regression

- Ridge regression is an extension for linear regression. It's basically a regularized linear regression model. The λ parameter is a scalar that should be learned as well, using a method called cross validation
- We need to notice about ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model.

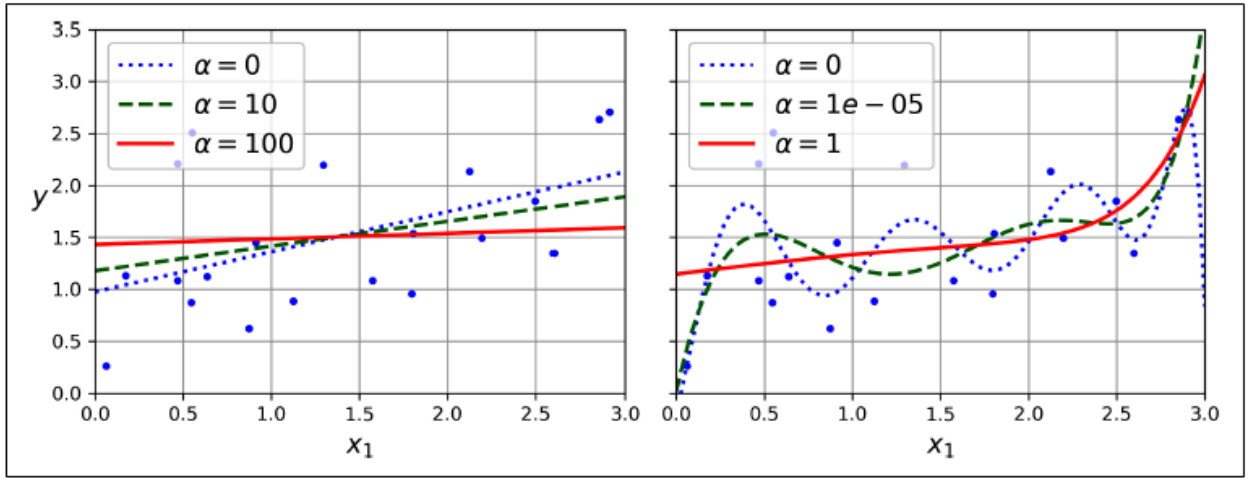
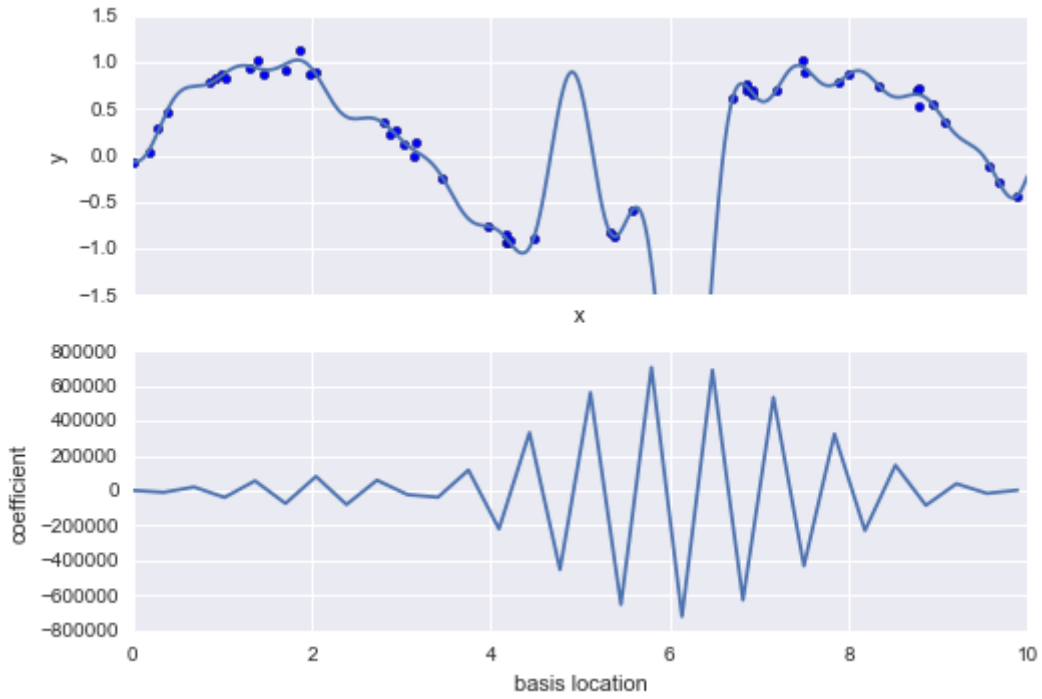
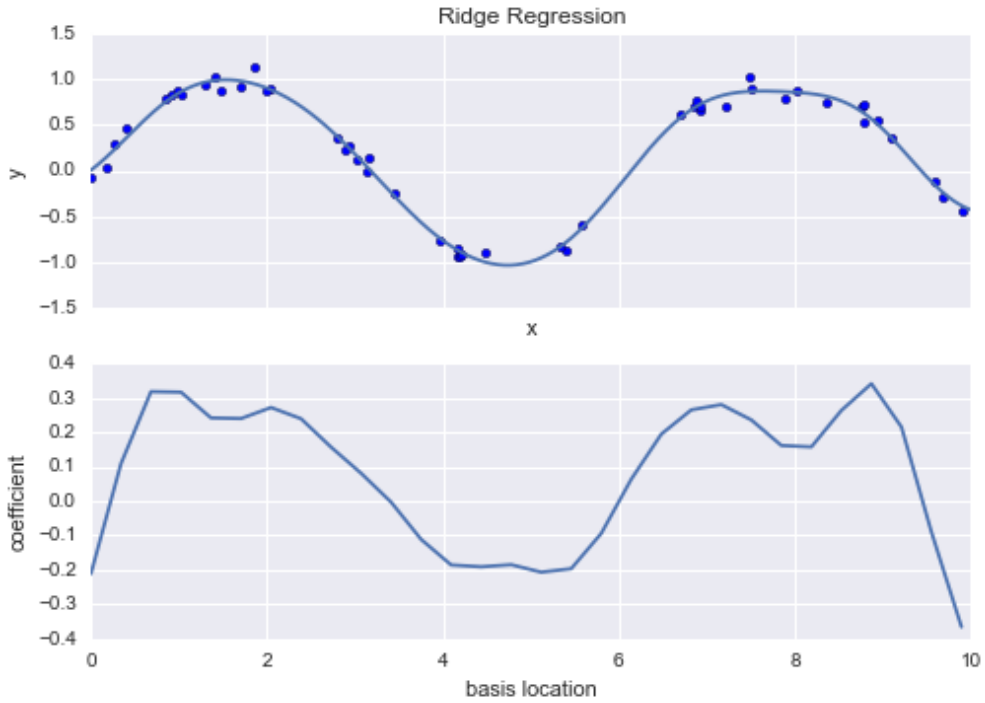


Figure 4-17. Linear (left) and a polynomial (right) models, both with various levels of ridge regularization

Regularization Impact: Ridge



No Regularization



Ridge

- In ridge regression, the complexity of the model is reduced by decreasing the magnitude of coefficients, but it never sets the value of coefficients to absolute zero

Dealing with Overfitting: Lasso regression

- The only difference from Ridge regression is that the regularization term is in absolute value. But this difference has a huge impact on the trade-off we've discussed before. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but setting them to zero if they are not relevant. Therefore, you might end up with fewer features included in the model than you started with, which is a huge advantage.

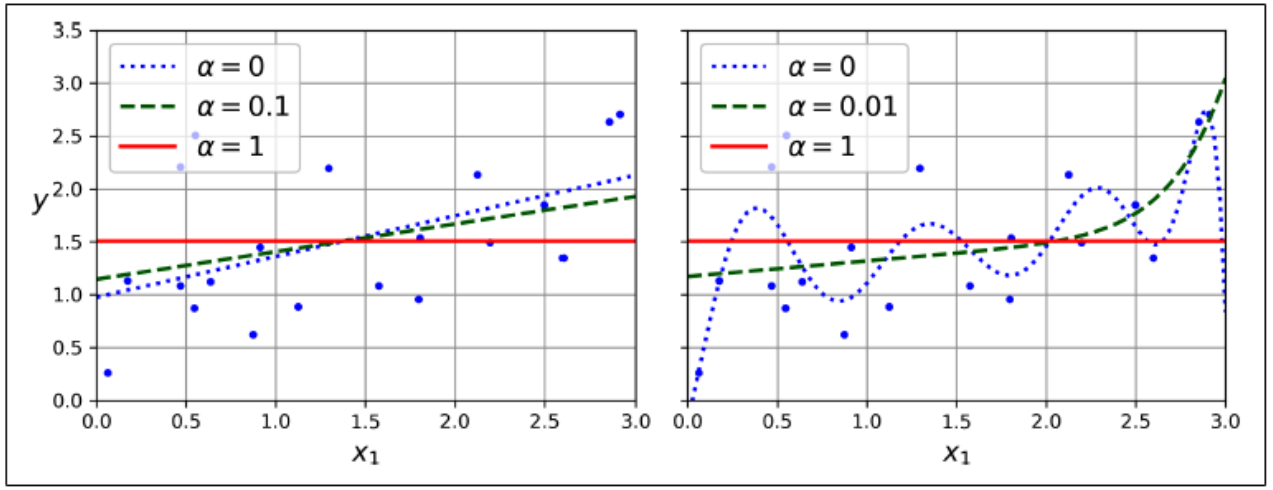
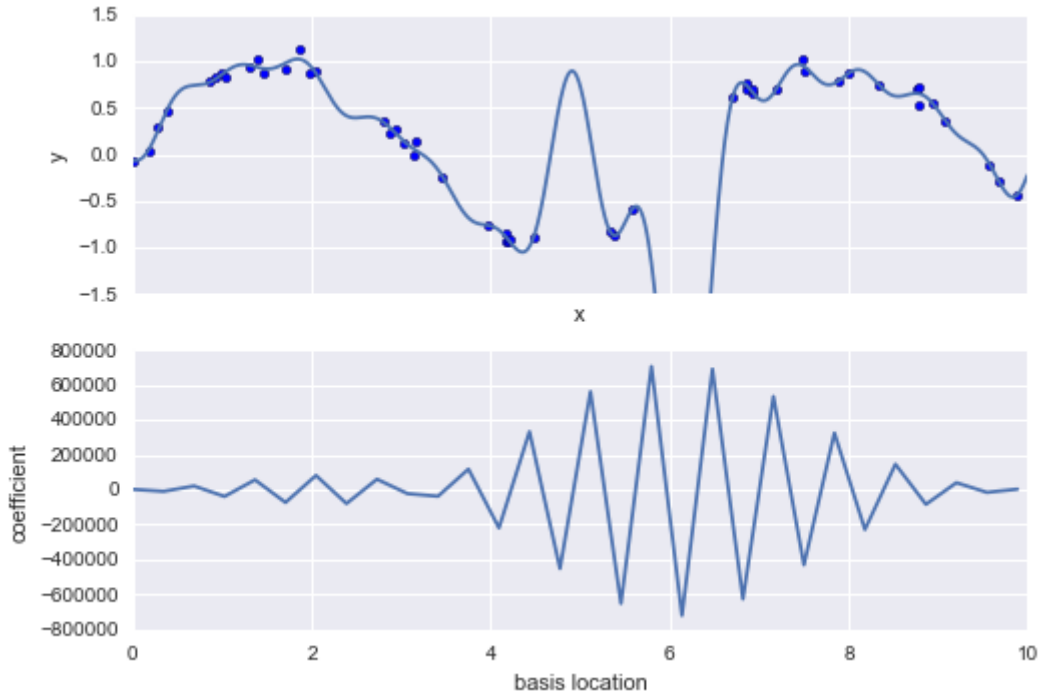
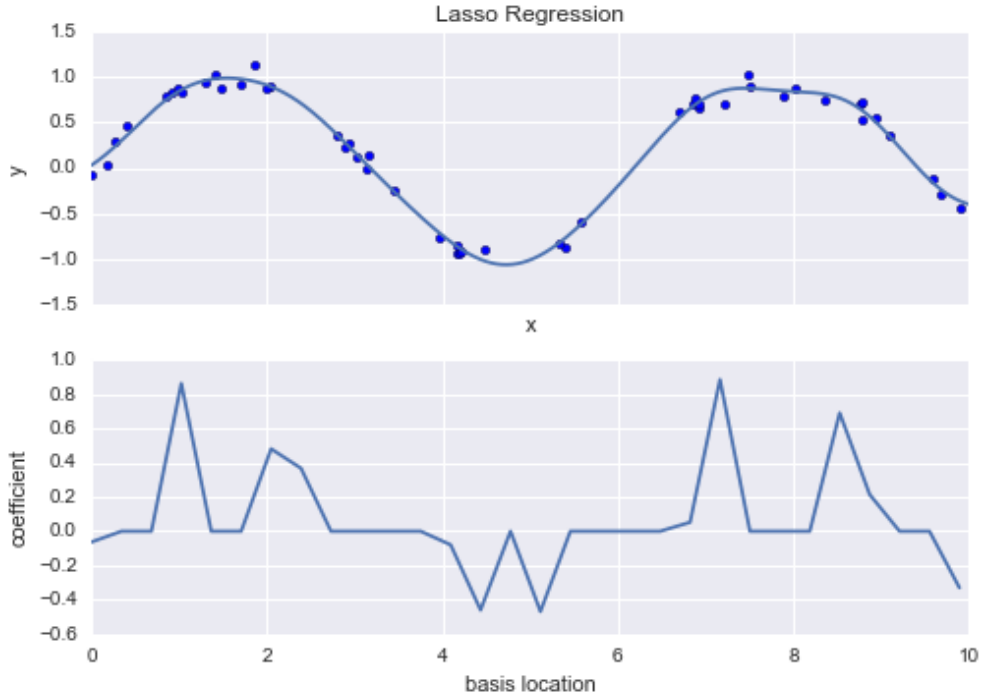


Figure 4-18. Linear (left) and polynomial (right) models, both using various levels of lasso regularization

Regularization Impact: Lasso



No Regularization



Lasso

- With the lasso regression penalty, most of the coefficients are exactly zero, with the functional behavior being modeled by a small subset of the available basis functions.

Regularization: Ridge & Lasso (& Elastic Net) Regression

- **Ridge regression** adds an L2 regularization penalty to the cost function
- **Lasso regressions** adds an L1 regularization penalty to the cost function
- **Elastic Net** uses both L1 and L2 penalty The regularization term is a weighted sum of both ridge and lasso's regularization terms, and you can control the mix ratio r.

$$Loss = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}_i^{(i)} - y^{(i)}\right)^2 + \boxed{\lambda\sum_{i=1}^{n} w_i^2} \rightarrow \text{L2 Penalty}$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)} - y^{(i)}\right)^2 + \lambda\sum_{i=1}^{n} w_i^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + \sum_{j=1}^{n} w_j x_j^{(i)} - y^{(i)}\right)^2 + \lambda\sum_{i=1}^{n} w_i^2$$

$$Loss = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}_i^{(i)} - y^{(i)}\right)^2 + \boxed{\lambda\sum_{i=1}^{n} |w_i|} \rightarrow \text{L1 Penalty}$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + \sum_{j=1}^{n} w_j x_j^{(i)} - y^{(i)}\right)^2 + \lambda\sum_{i=1}^{n} |w_i|$$

Regularization: Ridge & Lasso Regression Example

$$\hat{y} = 22x_1 + 103x_2 - 14x_3 + 109x_4 - 93x_5 + 203x_6 + 87x_7 - 55x_8 + 8$$

If we add regularization and train the model again, we end up with a simpler model. The following two properties can be shown mathematically:

- If we use L1 regularization (lasso regression), you end up with a model with fewer coefficients. Thus, we may end up with an equation like $\hat{y} = 2x_1 + 1.9x_4 + 8$
- If we use L2 regularization (ridge regression), we end up with a model with smaller coefficients. Thus, we may end up with an equation like $\hat{y} = 2x_1 + 4x_2 - 0.5x_3 + 1.2x_4 - 9x_5 + 2.5x_6 + 0.7x_7 - 0.5x_8 + 8$

A quick rule of thumb to use when deciding if we want to use L1 or L2 regularization follows: if we have too many features and we'd like to get rid of most of them, L1 regularization is perfect for that. If we have only few features and believe they are all relevant, then L2 regularization is what we need, because it won't get rid of our useful features.

Dealing with Overfitting: Early Stopping

- Another way to regularize iterative learning algorithms such as gradient descent is to stop training as soon as the validation error reaches a minimum. We decide which model to use is to pick the one that has the smallest validation error.
- Plotting the training and testing errors can give us some valuable information and help us examine trends.

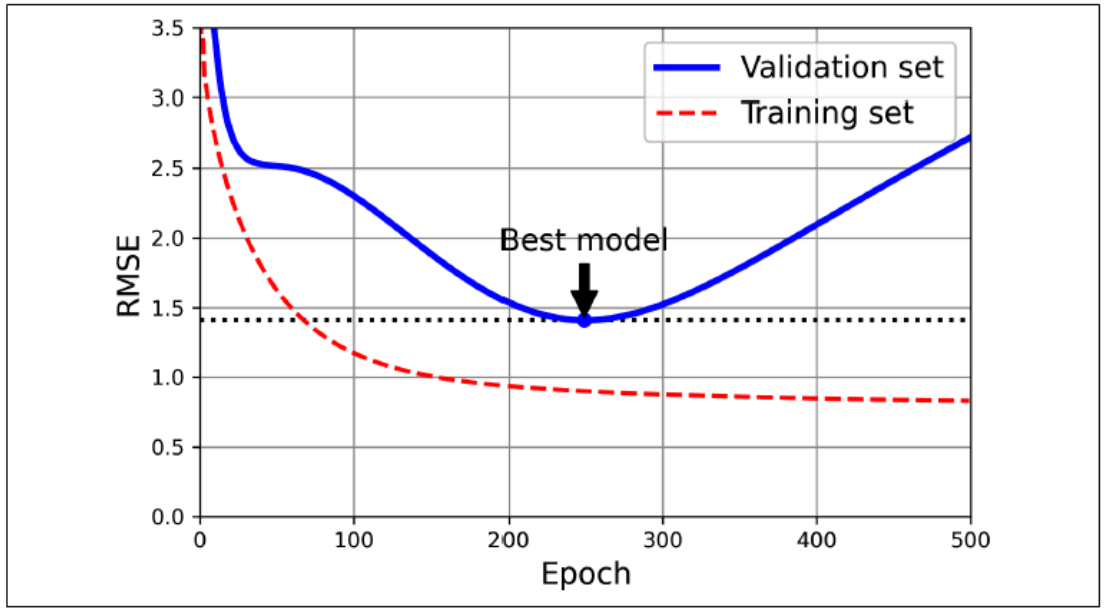


Figure 4-20. Early stopping regularization

Polynomial Regression (Higher Degree Fitting)

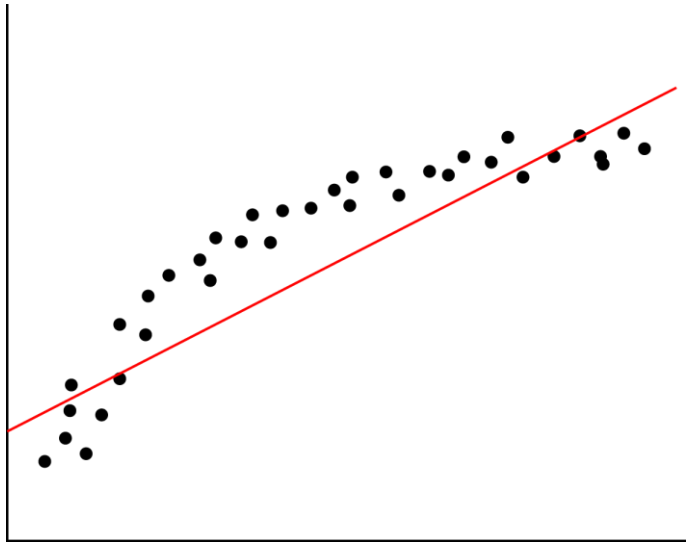
- In the case that relationship between input features of the data and the target output isn't linear, it may be more appropriate to use a polynomial equation

$$y = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n$$

- Ex: when fitting a polynomial of degree 3, the equation will be

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

- Do we need to develop a new model for this?



Figures from Mike de Waard (https://xyclade.github.io/MachineLearning/)

Polynomial Features

- Note that when there are multiple features, polynomial regression can find relationships between features, which is something a plain linear regression model cannot do. This is made possible by the fact that Polynomial Features also adds all combinations of features up to the given degree.
- For example, if there were two features *a* and *b*, Polynomial Features with degree=3 would not only add the features *a*2, *a*3, *b*2, and *b*3, but also the combinations *ab*, *a*2*b*, and *ab*2.

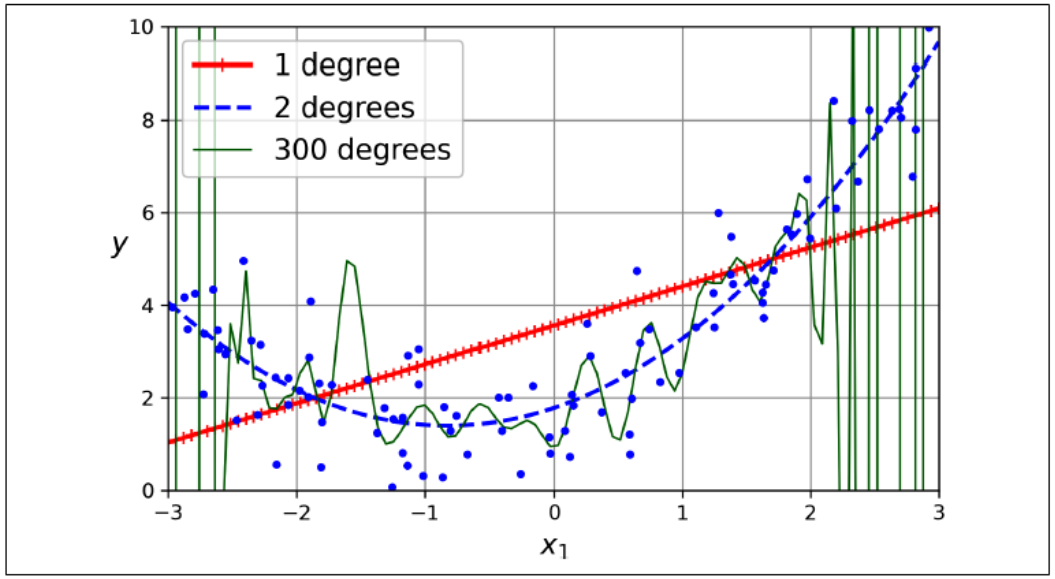


Figure 4-14. High-degree polynomial regression

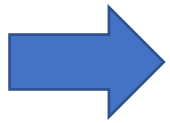
Polynomial Features

- Considering the general linear equation.

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

- Notice that if we let $x_1 = x, x_2 = x^2, x_3 = x^3$, then we are fitting the polynomial model.
- By transforming the input features into desired degree, it is still possible to utilize the linear regression model

X
1
2
3
…

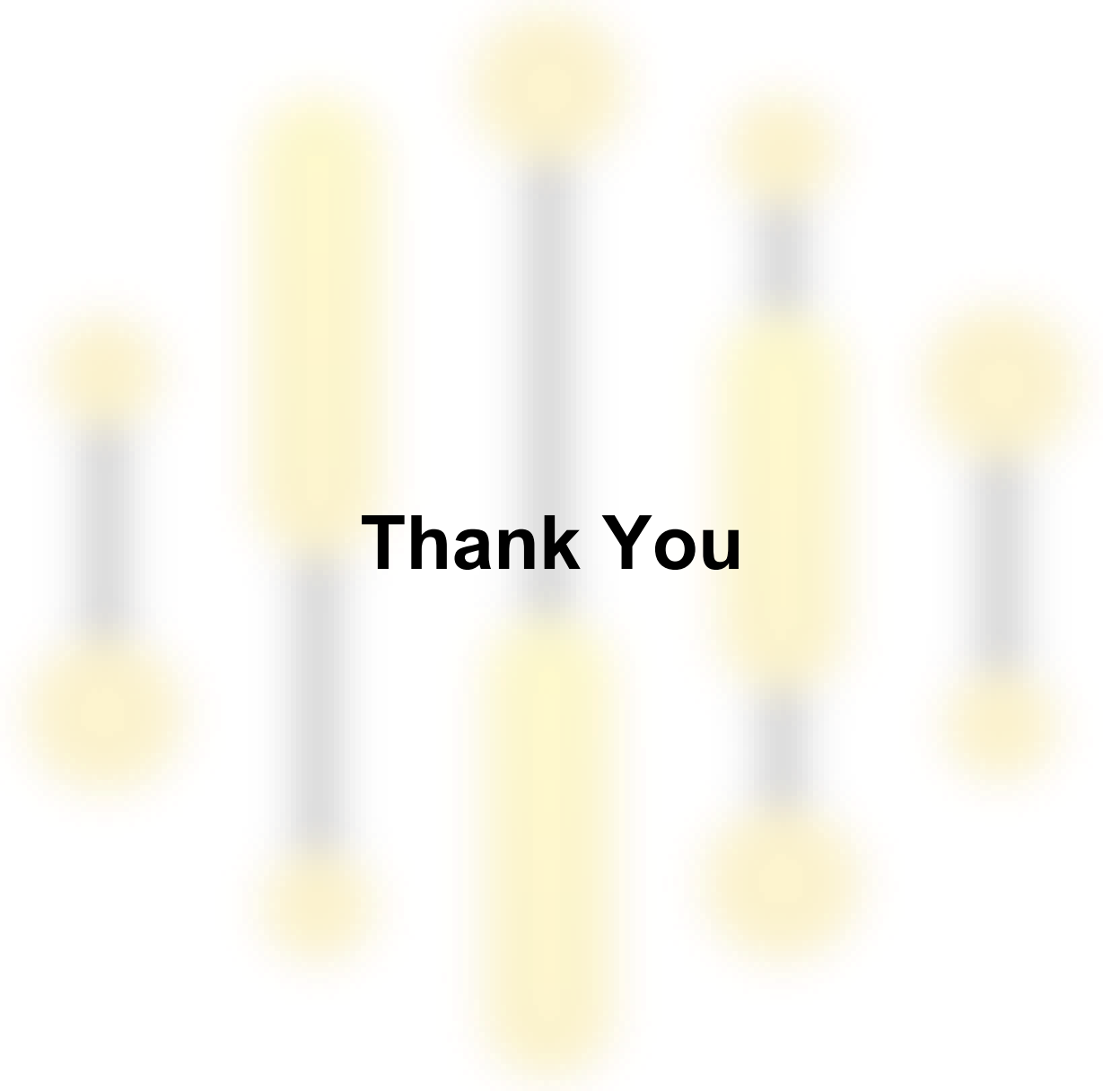


X	X^2	X^3
1	1	1
2	4	8
3	9	27
…	…	…

- Again, there is a library command to do perform this task.
- Note that Ridge and Lasso regularizations can be applied on polynomial regression as well

Python Tutorial

- Colab!
- 2.3 Regularization (Housing-simple)
- 2.4 Polynomial Regression (poly_example)



Thank You

GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวช.)

Facebook

Twitter

Blockdit

govbigdata

Follow us on

GBDi

gbdi.depa.or.th

YouTube

Government Big Data Institute (GBDi)

Line Official

@gbdi

BIG DATA
THAILAND

Follow us on

BIG DATA
THAILAND

Website

Facebook

Blockdit

Twitter