2695 Machine Learning
Masters Program in Economics, Finance and Management

NOVA SCHOOL OF BUSINESS & ECONOMICS

2a Video Lecture

LINEAR REGRESSION

Different types of ML algorithms

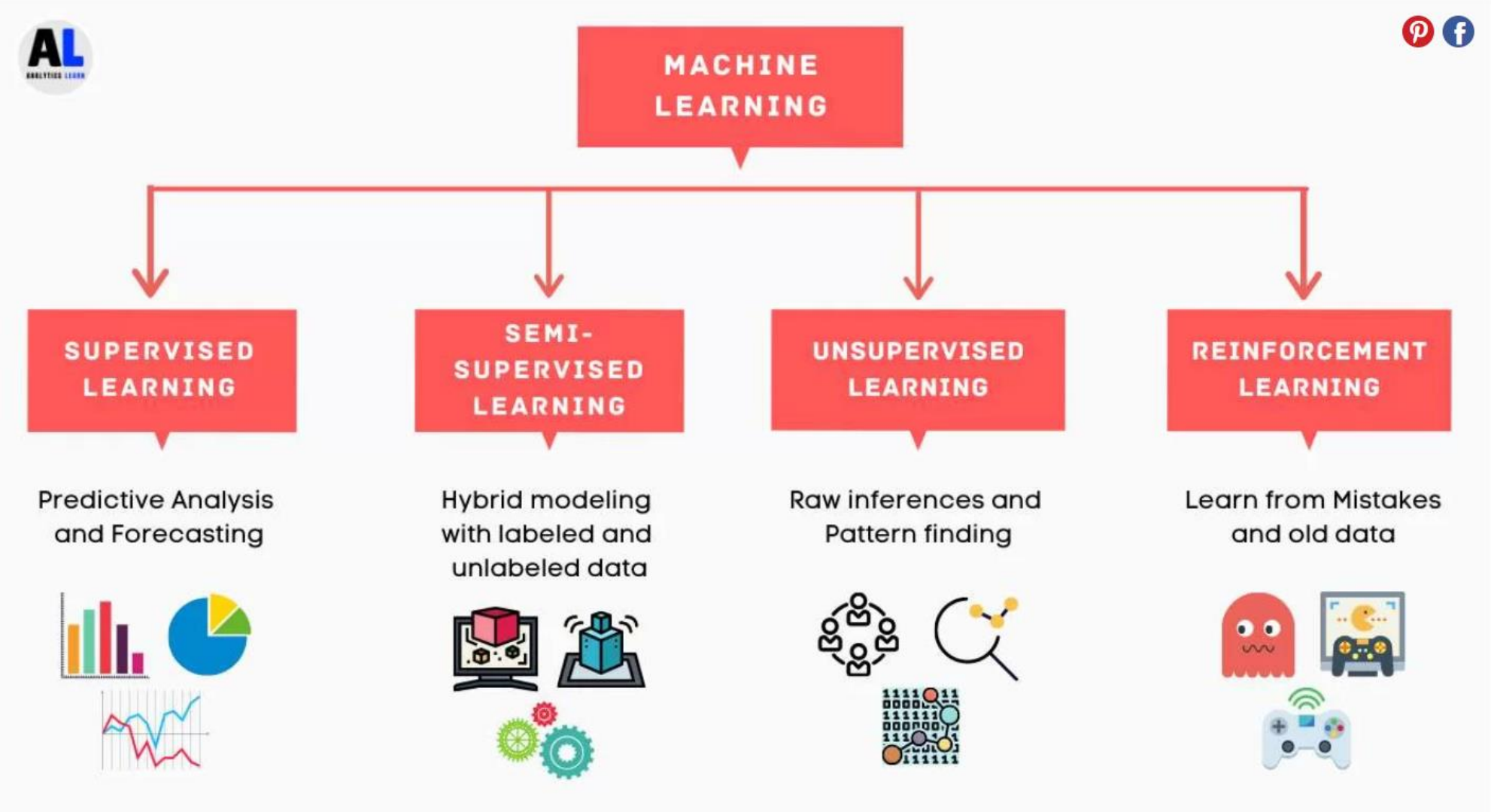
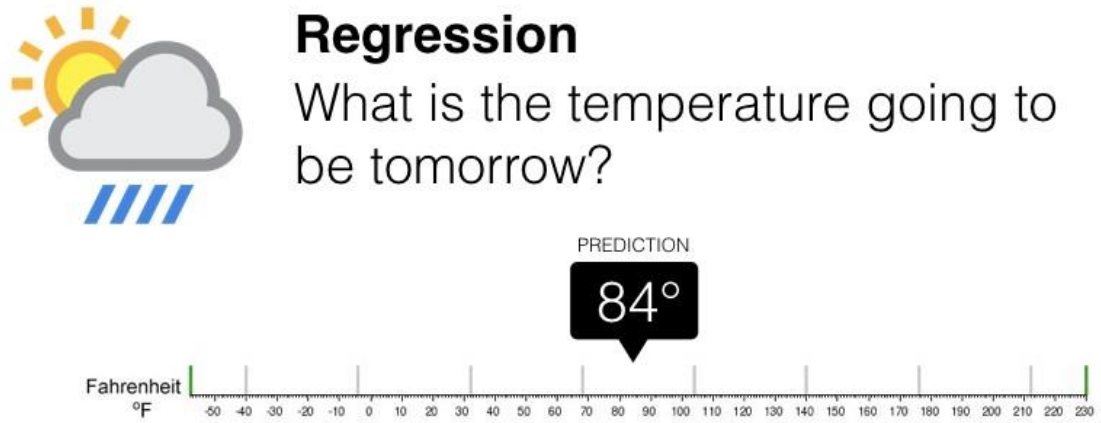


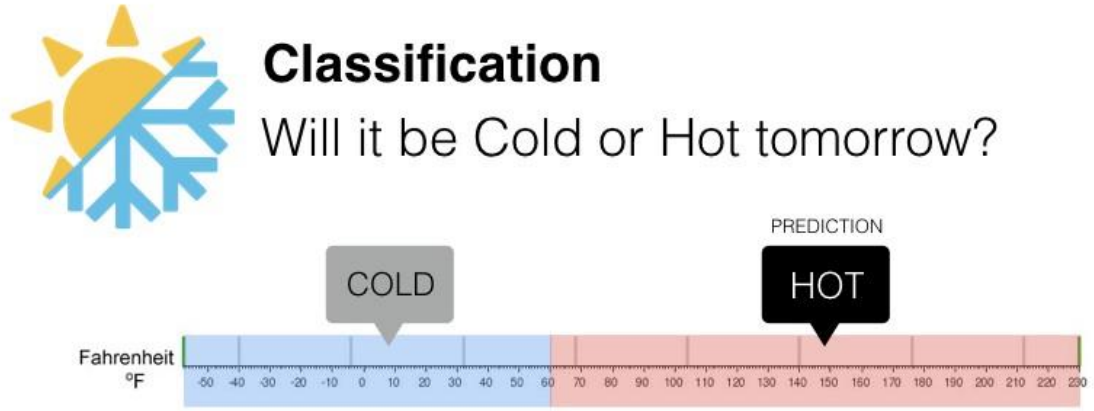
Image credit: Analytics Learn

Supervised learning: Classification vs Regression

Predict: **numeric** continuous value

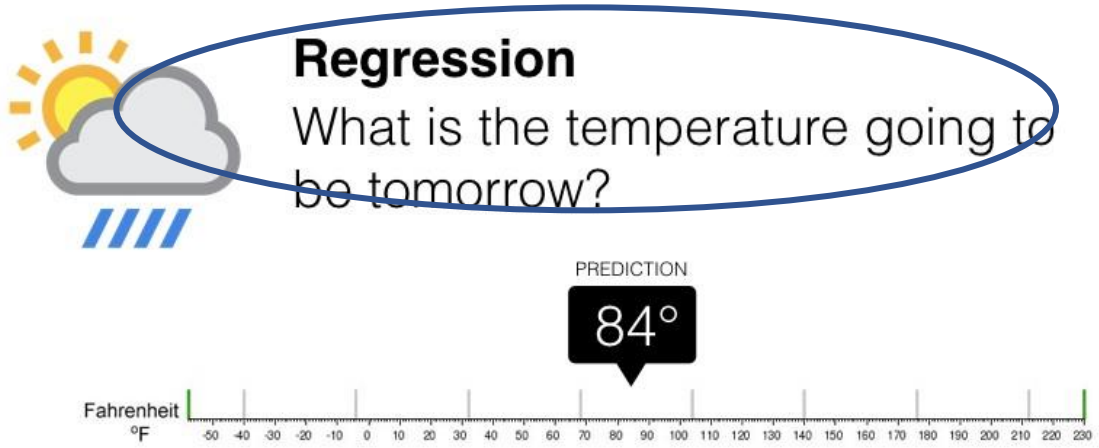


Predict: **categorical** value

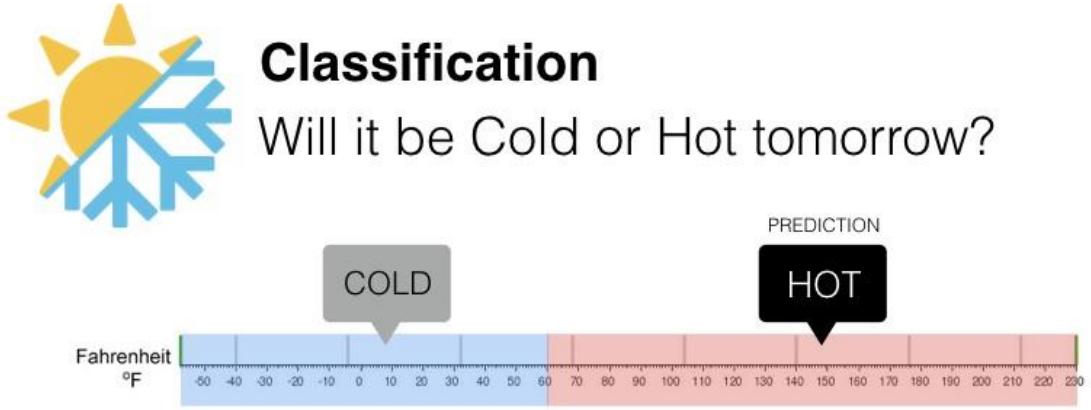


Supervised learning: Classification vs Regression

Predict: **numeric** continuous value



Predict: **categorical** value



Goal of Linear regression

Goal: Given input x we would like to compute output y

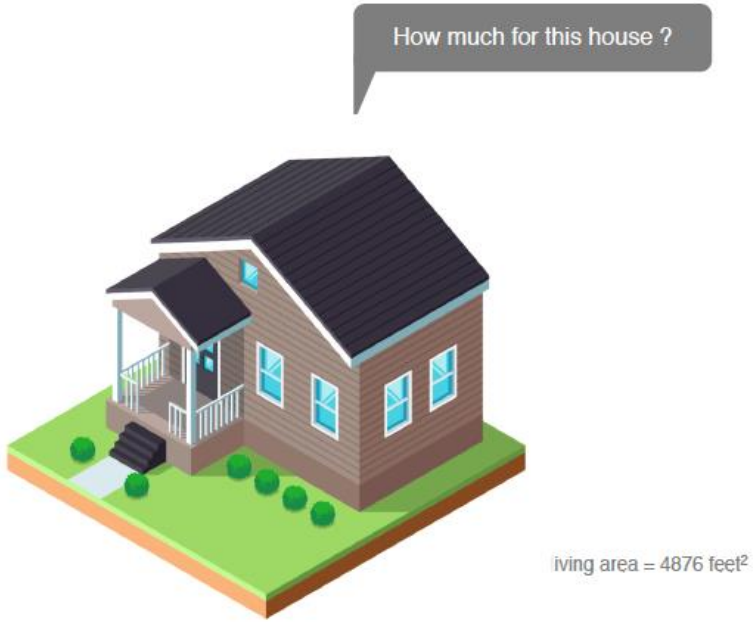
Living Area (Feet²)	Price ($)
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1780	229,500
1890	323,000
3560	662,500
1160	468,000
1430	310,000
1370	400,000
1810	530,000
...	...
x	y

Data set

Example: Predict house price

x: living area

y: price



Linear model

Goal: **Given an input x we would like to compute an output y**

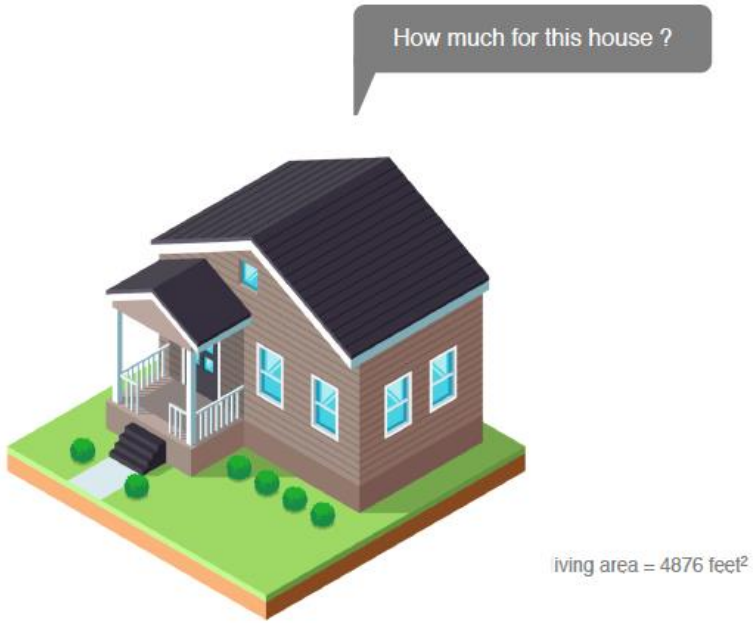
Living Area (Feet²)	Price ($)
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1780	229,500
1890	323,000
3560	662,500
1160	468,000
1430	310,000
1370	400,000
1810	530,000
...	...
x	y

Data set

Example: Predict house's price

x: living area

y: price



In linear regression we assume that y and x are related with the following equation:

$$y = \theta\, x + \varepsilon$$

what we are trying to predict

what we observe

- θ: parameter
- ε: measurement or other noise

$$\hat{y} = \theta\, x$$

our estimate

The best possible line

Our goal is to estimate θ from a training data of $<x^i,y^i>$ pairs in order to find the "*best possible line*".

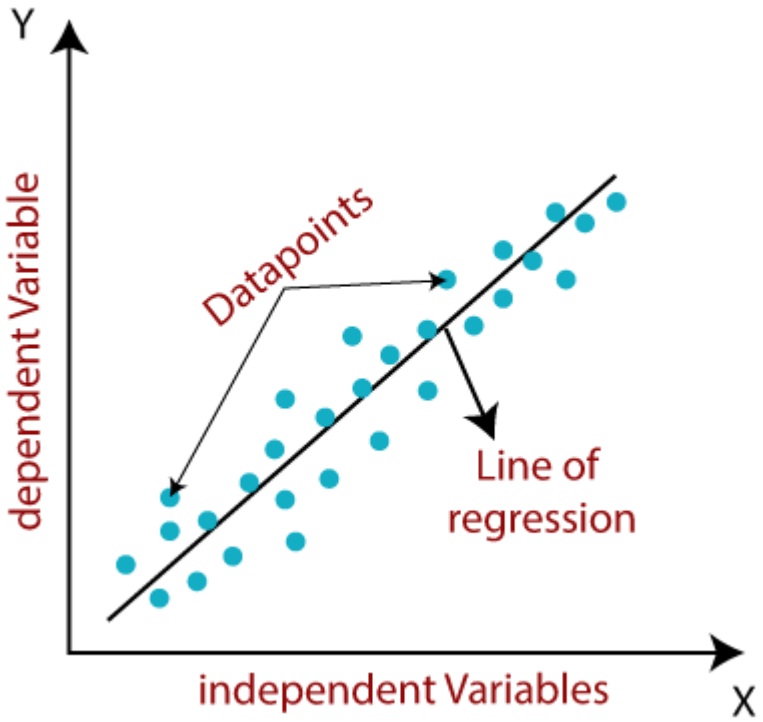
Living Area (Feet²)	Price ($)
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1780	229,500
1890	323,000
3560	662,500
1160	468,000
1430	310,000
1370	400,000
1810	530,000
...	...
x	y

Data set

1^{st} data point $<x^1,y^1>$

3^{rd} data point $<x^3,y^3>$

i^{th} data point $<x^i,y^i>$



What is the line with the *best fit*

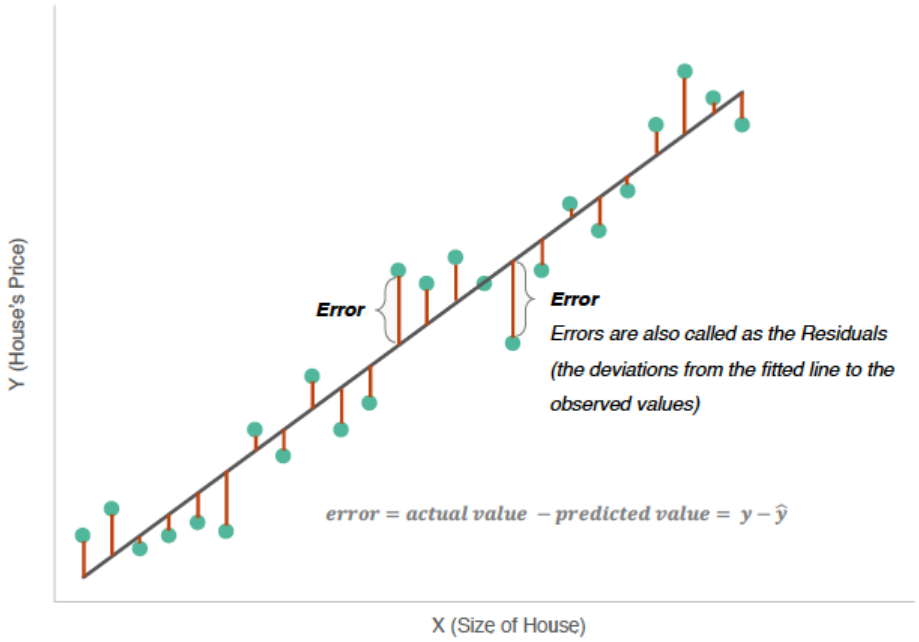
- "Best" line depends on the **objective (loss, cost) function**: Objective function should reflect our goal.
- Cost function determines how much penalty should be assigned to an instance based on the error in the model's predicted value
- There exist many different cost functions, since there are many ways to compute the error between an estimated value and an actual value.

Mean squared error

- "Best" line depends on the **objective (loss, cost) function**: Objective function should reflect our goal.
- Cost function determines how much penalty should be assigned to an instance based on the error in the model's predicted value
- There exist many different cost functions, since there are many ways to compute the error between an estimated value and an actual value.

- $\frac{1}{m}\sum_i \left(y^i - \theta x^i\right)^2$

m: number of training data points



$\hat{y} = \theta\ x$

Objective functions

- "Best" line depends on the **objective (loss, cost) function**: Objective function should reflect our goal.
- Cost function determines how much penalty should be assigned to an instance based on the error in the model's predicted value
- There exist many different cost functions, since there are many ways to compute the error between an estimated value and an actual value.

- $\frac{1}{m}\sum_i \left(y^i - \theta x^i\right)^2$

m: number of training data points

- Why this objective function:
  - minimizes squared distance between measurements and predicted line: strongly penalizes very large errors
  - easy to compute

Finding the optimal value of the parameter

- Our goal is to estimate θ from a training data of $<x^i,y^i>$ pairs to find the "*best possible line*" $y = \theta x + \varepsilon$
- We can find the optimal value for the parameters by minimizing our cost function:

$$\frac{1}{m}\sum_i \left(y^i - \theta x^i\right)^2$$

- *Optimal value* means:
  - For any other values of θ, the cost function would be higher for our **training data.**
- *Optimal value does not* mean:
  - that our model is generally a good model for the data
  - that our model will perform well on new instances.

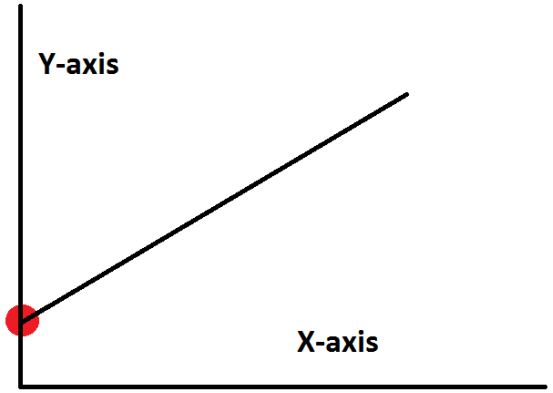
Introducing the bias (intercept term)

So far, we assumed that the line passes through the origin.

- What if the line does not?
- No problem, simply change the model to:

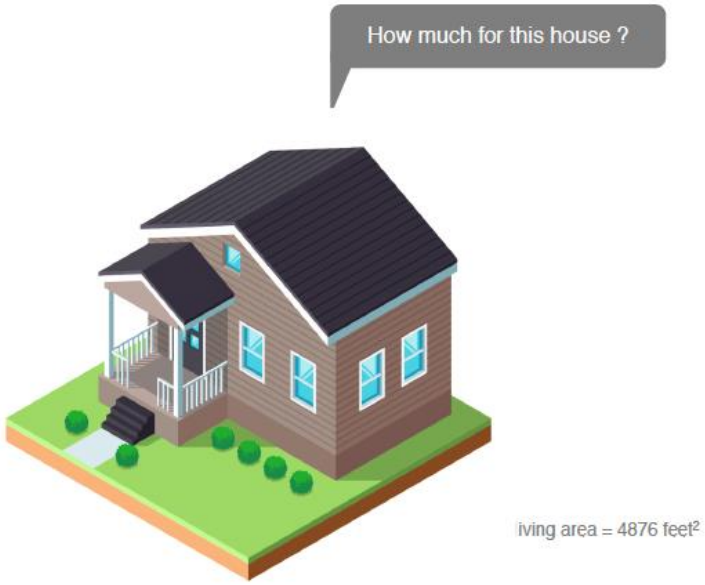
$$y = \theta_0 + \theta_1 x + \varepsilon$$

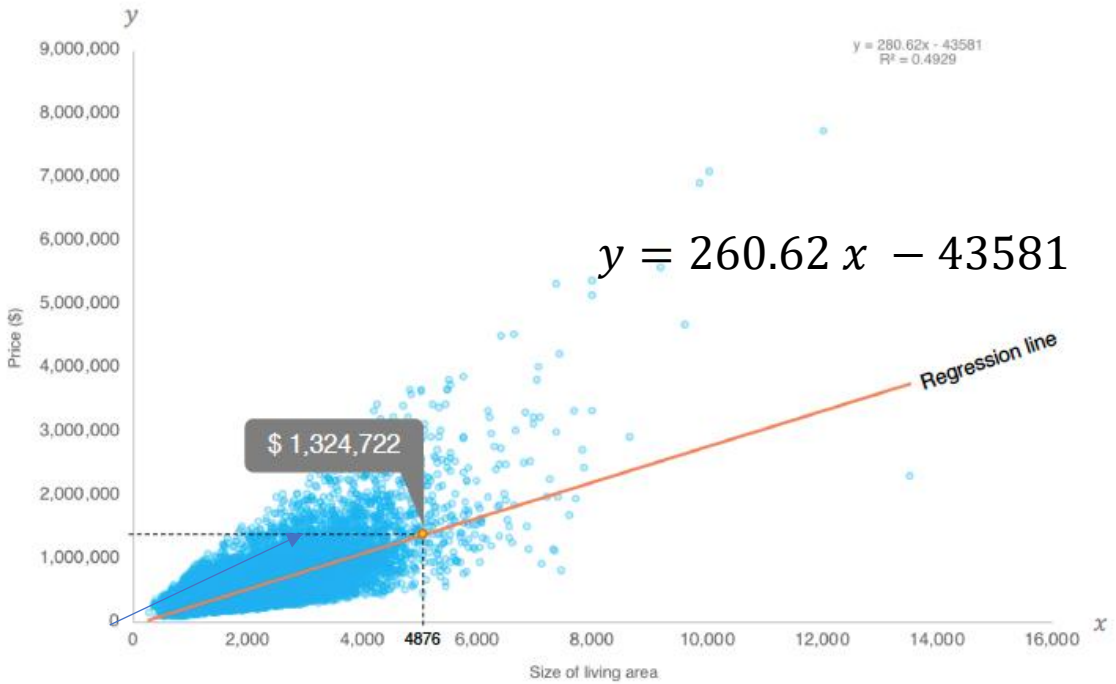
- y intercept: value at which the fitted line crosses the y-axis



Training vs Inference

- **Training:** We learn the values of parameters θ_1 and θ_0 using pairs <x^i,y^i>.
- **Inference (Prediction):** Using the model, we can predict the value of y, for a *new* x.





$$y = 260.62\,x\ - 43581$$

- What is the price of a house with this area?

From simple to multiple linear regression

- Generally, one dependent variable (y) depends on multiple factors (x_1, x_2,…).
- For example, the price of a house depends on many factors like the size of the house, number of rooms, attached facilities, distance of nearest station from it, distance of nearest shopping area from it…

The house's price depends on **multiple features** (variables)

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Multiple (multivariate) linear regression

This is quite like the simple linear regression model, but with multiple independent variables contributing to the dependent variable.
Each training sample has n attributes (x_1,x_2,…,x_n) and a corresponding single value of y.

example

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

In this example,

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

Price — size — Number of bedrooms — Number of floors — Age of Home

General linear regression

- What if your data is actually more complex than a simple straight line?
- Instead of the input variables (x) use some function of these values.
- Calculate functions of x, the regression model remains linear with respect to the coefficients.

Linear regression:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$\hat{y}=\theta\, x$

General linear regression:

$$\hat{y} = \theta_0 + \theta_1 \phi_1(x) + \theta_2 \phi_2(x) + \ldots + \theta_n \phi_n(x)$$

$\hat{y} =\theta\, x^2$

Polynomial regression

- Add powers of each feature as new features, then train a linear model on this extended set of features.

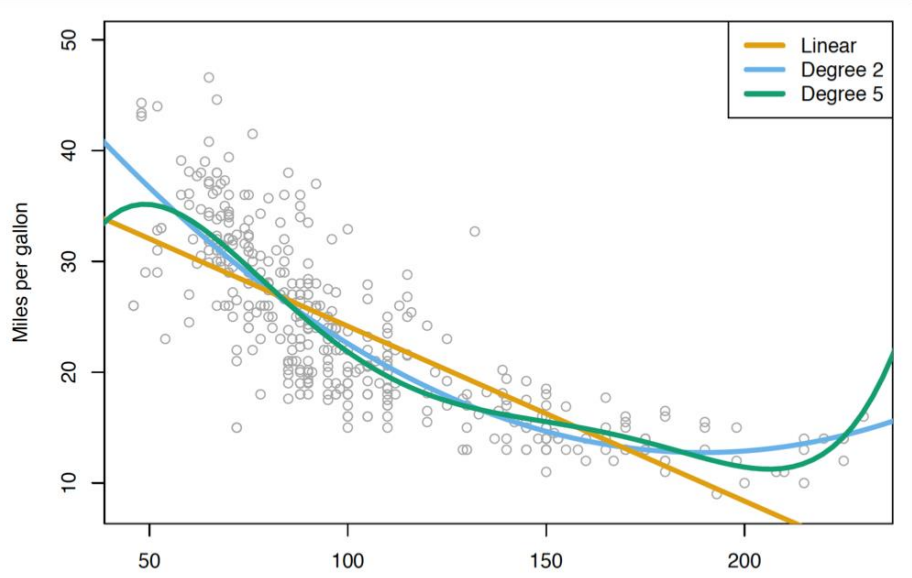
$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_n x^n$$

where *n* is called the **degree** of the polynomial

For lower degrees, the relationship has a specific name like quadratic.

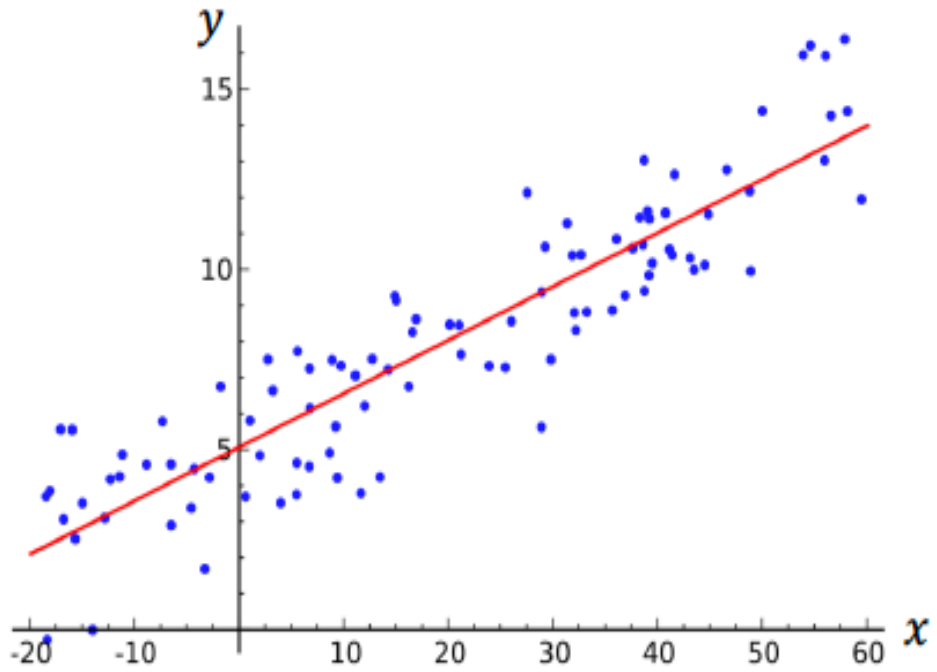
- n = 1, a first degree polynomial is simply a **straight line**. $\dashrightarrow \hat{y} = \theta_0 + \theta_1 x$
- n = 2 , a second degree polynomial is called a **quadratic.** $\dashrightarrow \hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2$
- n = 3, it is **cubic.** $\dashrightarrow \hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
- n = 4, it's called **quartic.** $\dashrightarrow \hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

The polynomial models can be used to approximate a complex nonlinear relationship



Linear Regression properties

- Supervised algorithm
- Training data point: one or more input values and a single output value
- Training: learning the line with the best fit
- Prediction: using the learnt line to predict the output value
- Simple to interpret, easy to train
- Used as a baseline for evaluating other, more complex models



NOVA SCHOOL OF BUSINESS & ECONOMICS

2b Video Lecture

ESTIMATING PARAMETERS OF LINEAR REGRESSION

Equation for calculating the parameters

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- $\hat{y}$ is the predicted value.
- n is the number of features.
- x_i is the i^{th} feature value.
- θ_j is the j^{th} model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \cdots, \theta_n$).

$$x = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ 1 & x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

i^{th} data point with n features

Normal Equation (Closed Form)

Solve θ analytically: a mathematical equation that gives the result directly

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

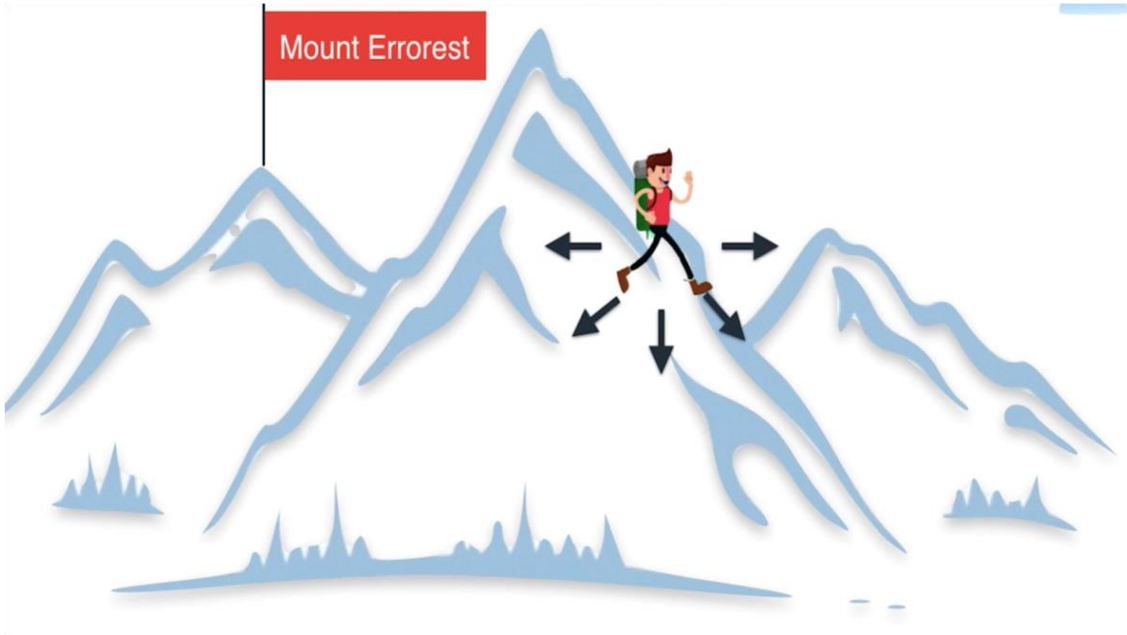
- $\hat{\theta}$ is the value of θ that minimizes the cost function.
- $\mathbf{y}$ is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

It's suitable for small feature set (e.g. < 1000 features)

Gradient Descent

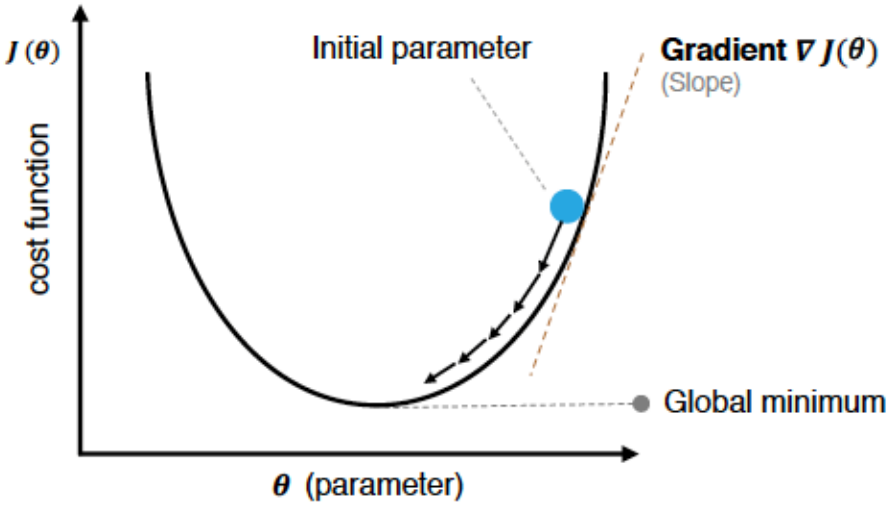
One of the most popular optimization methods in Machine Learning (*)

How to get to the lowest place?



(*) used in training of neural networks

How does Gradient descent work



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

Gradient descent algorithm

The parameter are iteratively updated in the following equation:

Learning rate (Step size)

$$\theta_{new} = \theta_{old} - \alpha \cdot \nabla_\theta J(\theta)$$

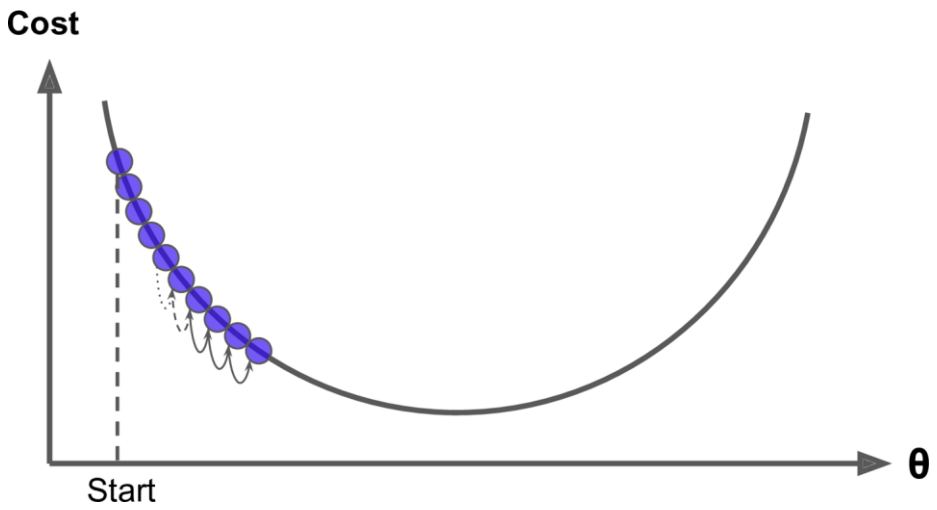
Gradient $\nabla_\theta J(\theta) = \frac{d}{d\theta} J(\theta)$

1. Pick a value for the learning rate α
2. Start by filling θ_{old} with random values (random initialization)
3. Calculate the gradient at the point θ_{old}. Update new parameter θ_{new} by following the opposite direction
4. Repeat until cost converges to the minimum

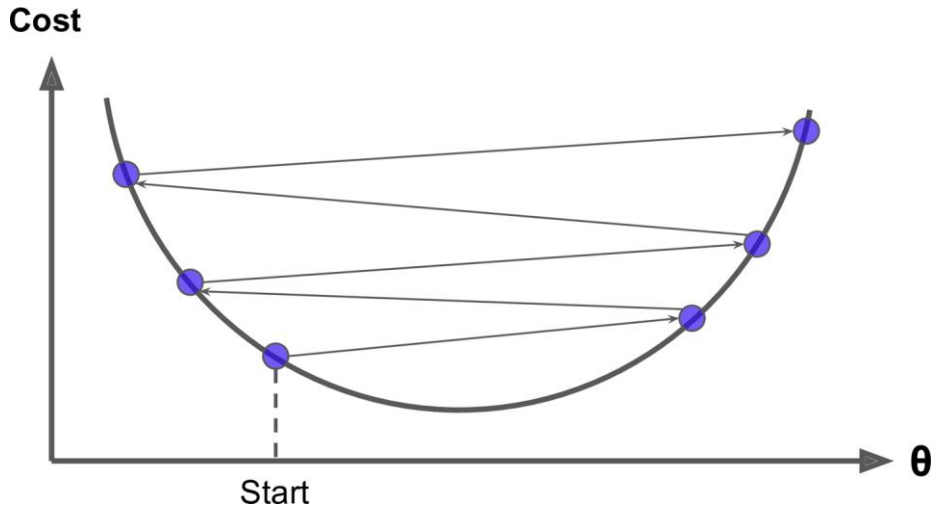
Learning rate

Learning rate is a hyperparameter

If the learning rate is too small



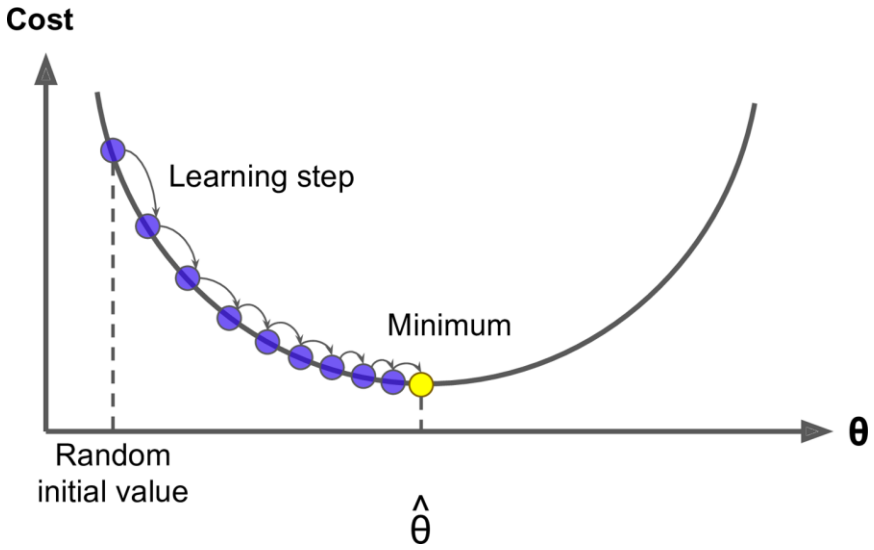
If the learning rate is too large



Gradient Descent Pitfalls

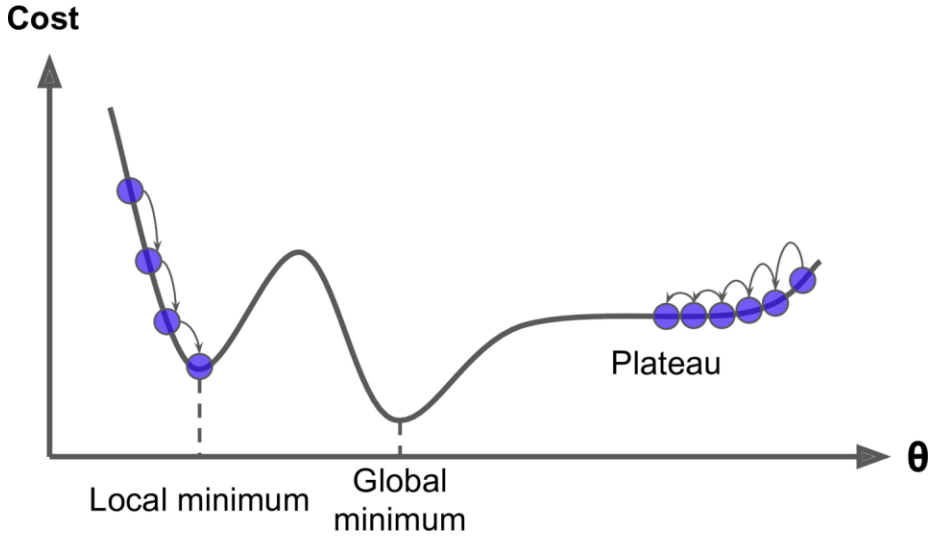
Gradient Descent for convex function & non-convex function

If the cost function is convex



The gradient descent is guaranteed to converge arbitrarily close to the global minimum when cost function is convex

If the cost function is not convex



If the random initialization starts the algorithm on the left, then it will converge to a local minimum, which is not as good as the global minimum.

2c Video Lecture

EVALUATION METRICS REGRESSION

Criteria for a *good* model

Accurate

Are we making correct predictions?

Interpretable

How easy is it to explain how the predictions are made?

Fast

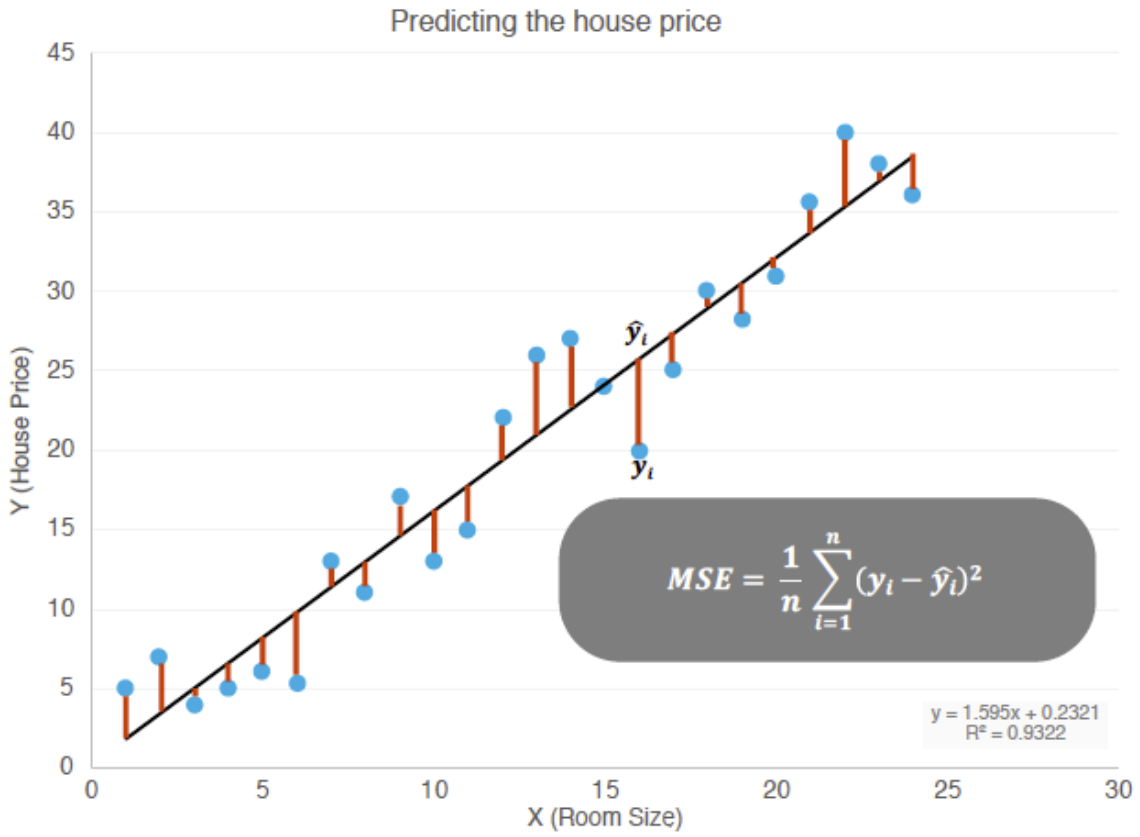
How long does it take to build a model and how long does the model take to make predictions?

Scalable

How much longer do we have to wait if we build/predict using a lot of data?

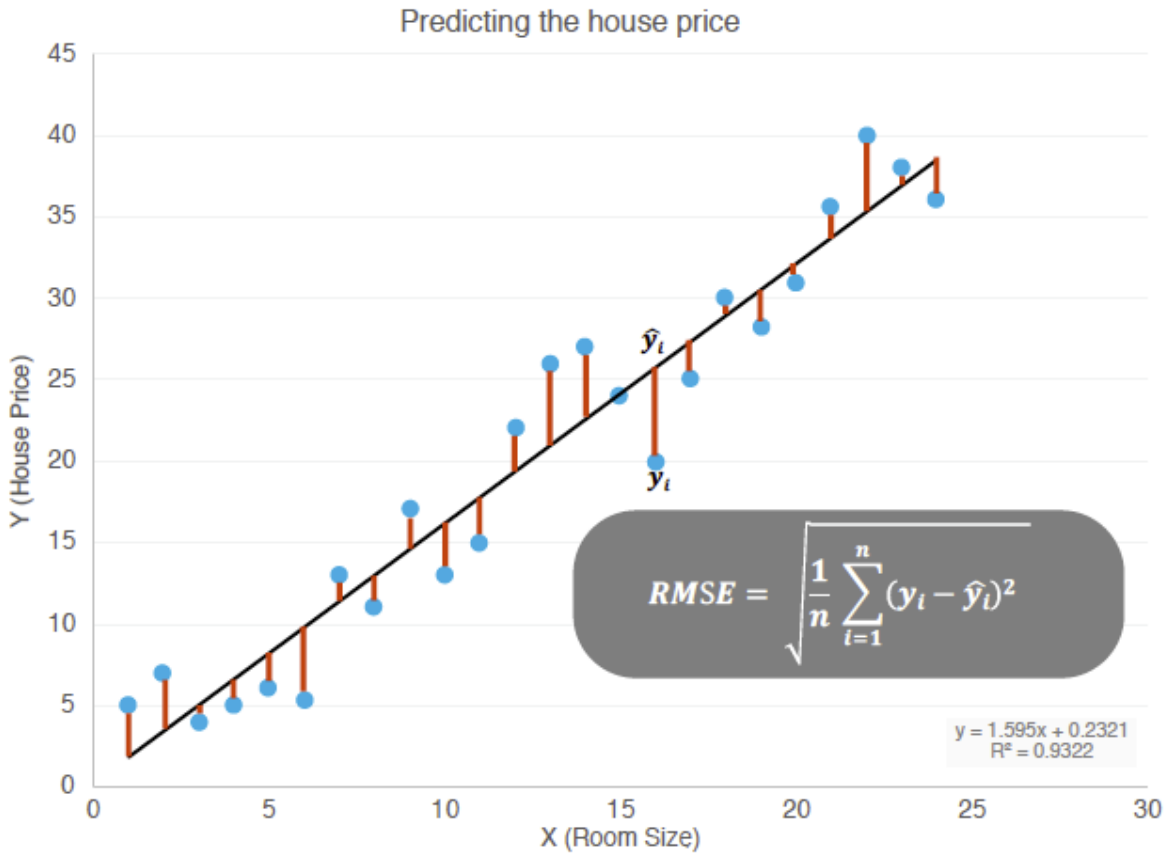
Mean Squared Error

MSE is the mean of the squared value of the errors



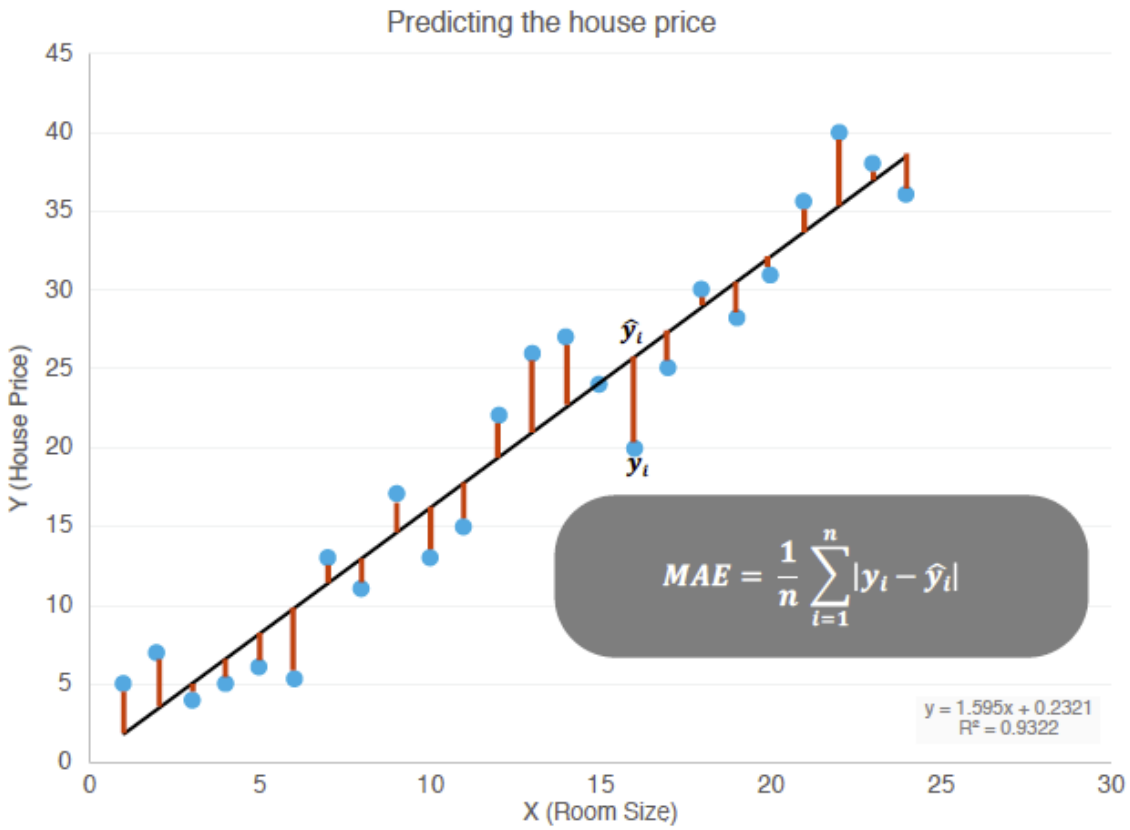
- MSE: mean of the squares of the difference between the actual value and the predicted values for all data points.
- A smaller score is better.

Root Mean Squared Error



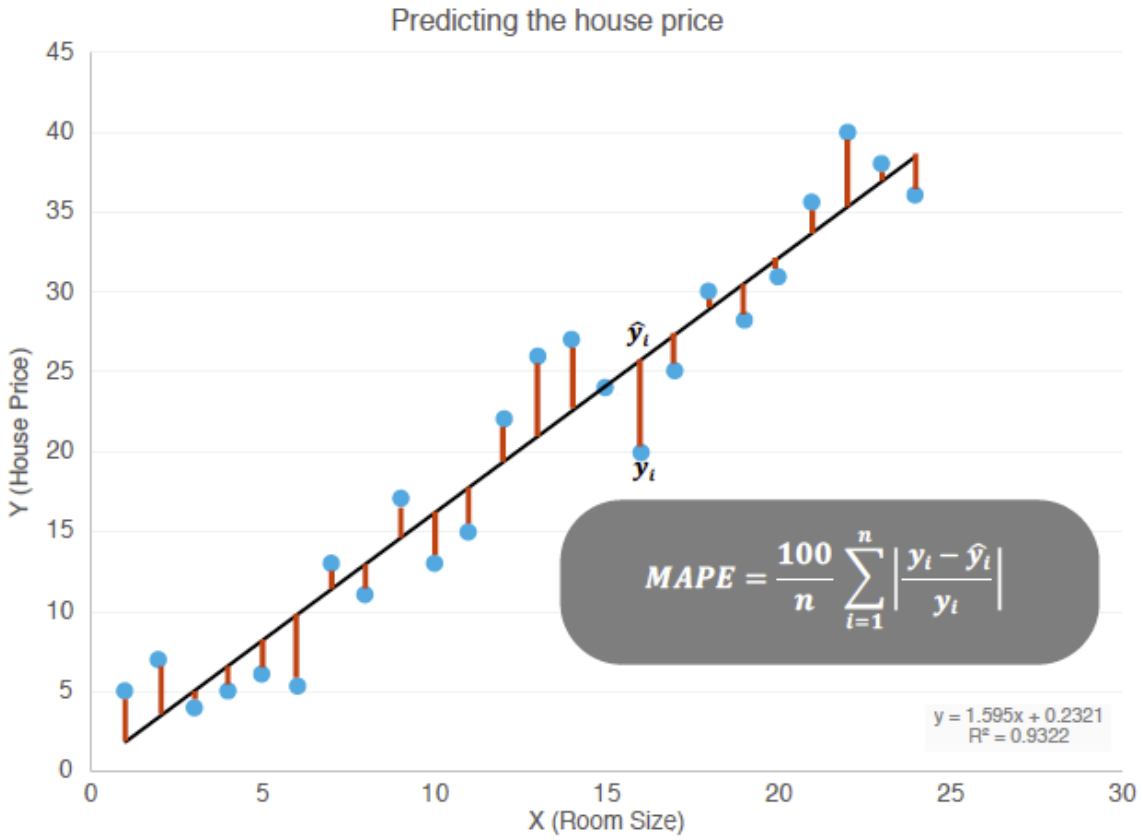
- The most popular evaluation metrics used in regression problems. It has same unit with “Y”.
- A smaller score is better.

Mean Absolute Error



- MAE: Mean of the absolute difference between the actual value and the predicted values for all data points.
- A smaller score is better.

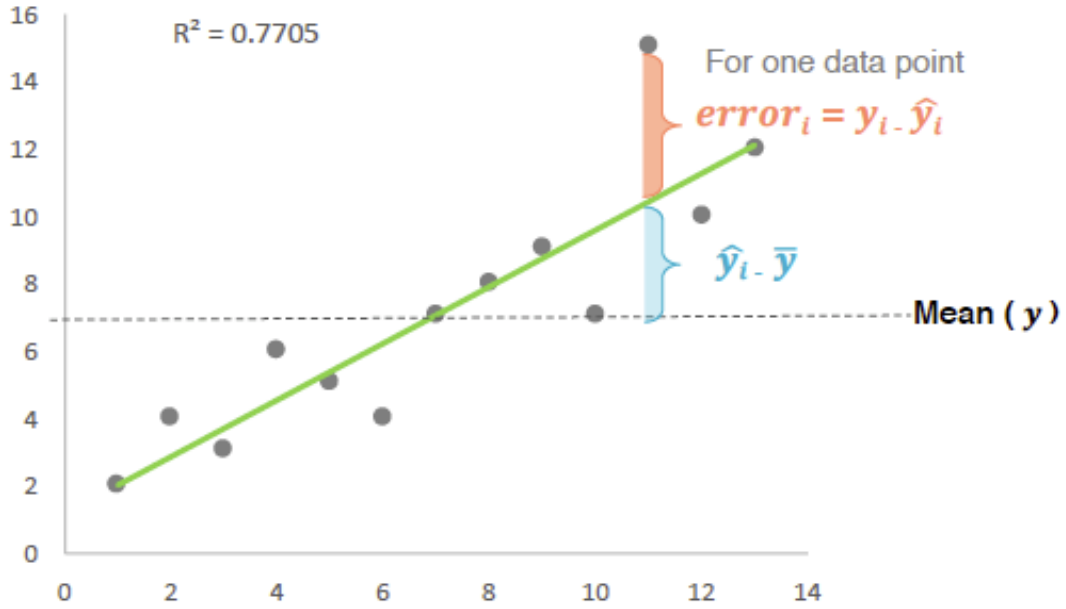
Mean Absolute Percent Error



- MAPE: measures the size of the error in percentage terms.
- One of the most popular measures for forecasting time series error.
- A smaller score is better.

R^2 - coefficient of determination

- R^2 Score is based on comparing our model to the simplest possible model (mean of y)



Calculate the sum of squares for All data points

The error sum of squares

$$SS_{error} = \sum_i (y_i - \hat{y}_i)^2$$

The regression sum of squares

$$SS_{reg} = \sum_i (\hat{y}_i - \bar{y})^2$$

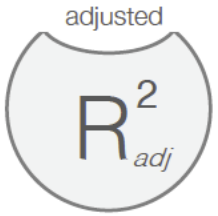
- $\hat{y}$ (y hat) is the predicted value
- y_i is the actual value/target value)
- y is the mean value

The total sum of squares

$$SS_{total} = SS_{reg} + SS_{error}$$

$$R^2 = \frac{SS_{reg}}{SS_{total}} = 1 - \frac{SS_{error}}{SS_{total}}$$

Adjusted R^2



Adjusted R^2 penalizes the model for the inclusion of variables:

- more useless variables to a model, adjusted R^2 decreases
- more useful variables, adjusted R^2 increases

Adjusted R^2 is always less than or equal to R^2

- 0: model has no predictive power
- 1: model that perfectly predicts
- In the real world, adjusted R^2 lies between these values

$$R_{adj}{}^2 = 1 - \left[\frac{(1-R^2)(n-1)}{n-k-1}\right]$$

n is the sample size (i.e., number of points in your data sample)
k is s the total number of explanatory variables in the model (not including the constant term)

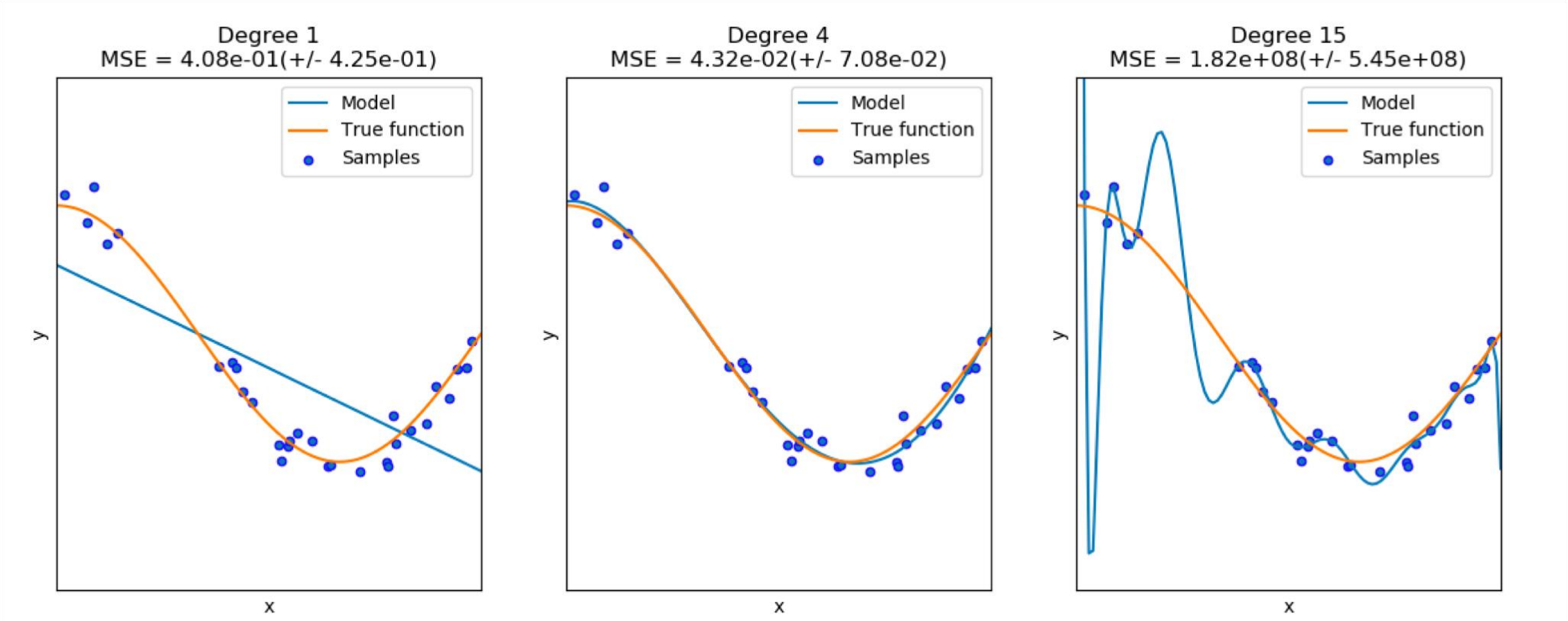
2d

OVERFITTING AND REGULARIZATION

What do we want to achieve with our ML model?

Plots of polynomials having various orders n

The plot shows the real function that we want to approximate, which is a part of the cosine function



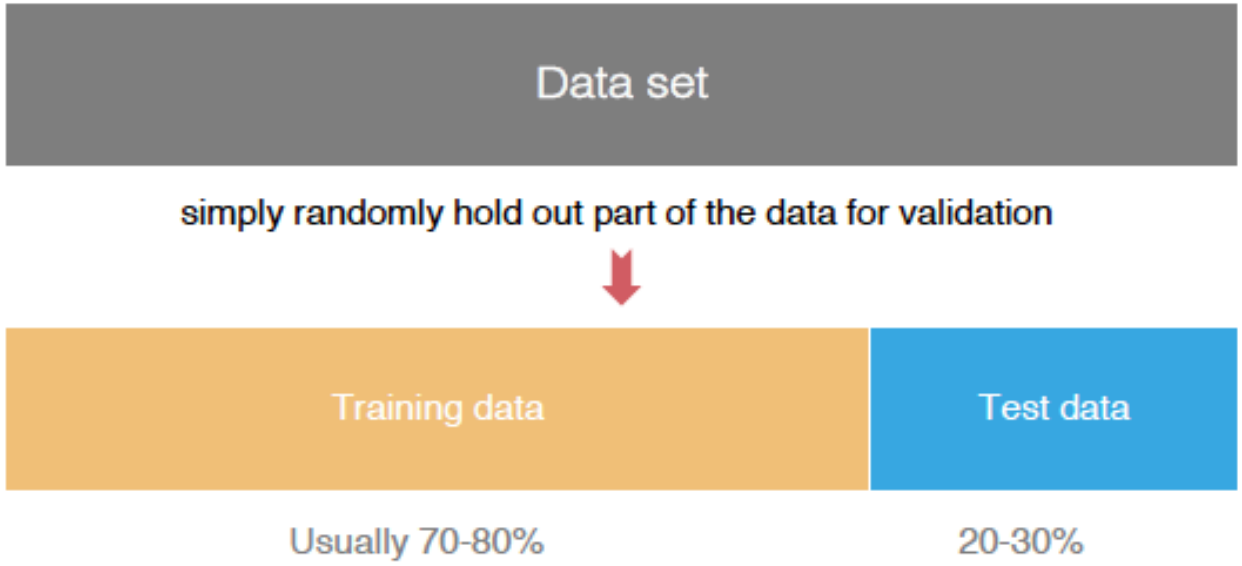
https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html#

Very important concepts: Generalization, underfitting and overfitting

- **Generalization:** model performs well on new, unseen data
- **Underfitting**: model is too simple to capture the patterns in the data
- **Overfitting**: model learns noise from the training data, leading to poor performance on new data

Ensuring good generalization by evaluating on unseen data

Train model on the training set and then test it on the test set



Use of regularization to prevent overfitting

Regularization: introducing a preference for simpler models

New cost function = original cost function + **penalty**

L_1 regularization

Penalizes **absolute values of the coefficients** (not the bias term)**.**

L_2 regularization

Penalizes **sum of the squares of the coefficients .**

L1 (Lasso) regularization

- L_1 reduces the impact of less important features, shrinks their coefficients all the way to 0.
- Performs **feature selection,** produces a **sparse model** (i.e., with few nonzero feature coefficients)

New cost function = MSE(Θ)+ $\alpha \sum_1^n |\Theta_i|$

Penalizes **absolute values of the coefficients**

The optimal parameters are those that minimize the new cost function.

hyperparameter: regularization strength

L2 (Ridge) regularization

- Does not eliminate features completely (coefficients can be very very small, but they are not zero).
- Reduces the magnitude of all coefficients

New cost function = MSE(Θ)+ $\alpha \sum_1^n \theta_i^2$

Penalizes **sum of the squares of the coefficients.**

The optimal parameters are those that minimize the new cost function.

hyperparameter: regularization strength