# What is Supervised Learning?

* Use Input(X) to label Output(Y)
* Learn from **labeled** data (with right answer)
* Prediction : Find approximate **Linear relation**
* Classification : Predict Category(bin/class)

# What is Unsupervised Learning?

* Find cluster(group) from **unlabeled** data
* Dimensionality reduction : Find structure easily

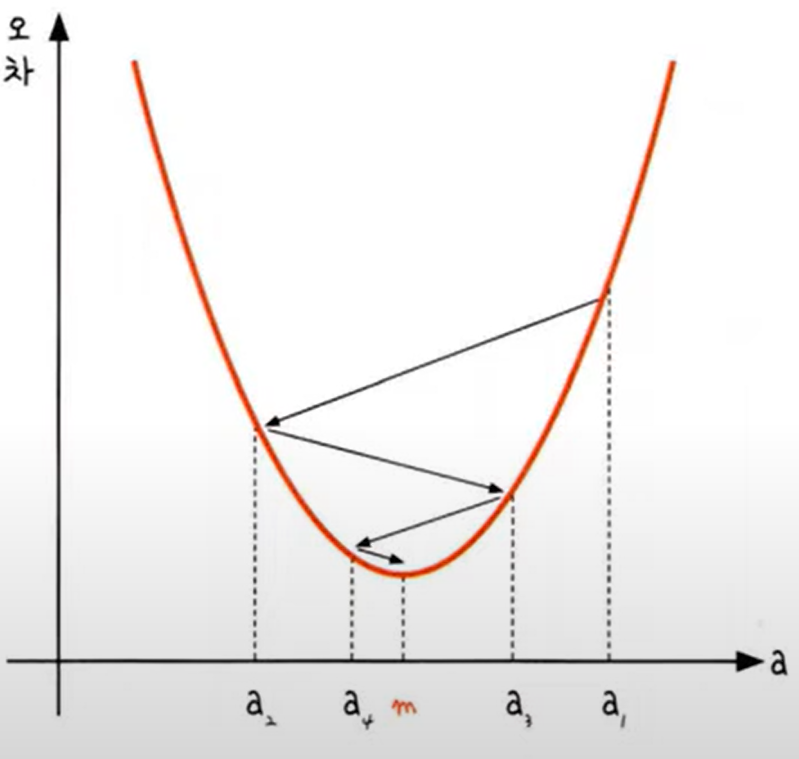
@ Linear Regression

# Single Linear Regression

* Learn algorithm using training set > function(model)
* Prediction value(y-hat) is estimated y
* Because of the line’s feature, only one variable is available.

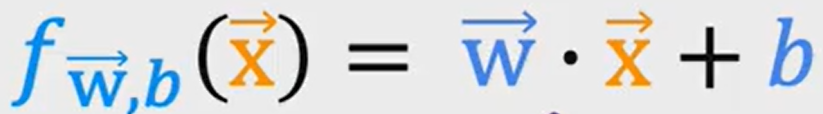
# Cost function (J)

* Calculate gap between mode’s predicted value and real value
* Lower cost function is a better model.



# Gradient descent algorithm

* Repeat until convergence min(w, b) (Repetition)
* 목표 지점(m)에 대한 방향으로 기울기를 하강시키면서 m에 접근함
* gradient가 감소할수록, 하강하는 거리도 줄어든다.
* Learning rate (advanced. DL) : 어느 정도의 weight로 w를 향해 접근하는가. 크기에 따라서 Overfitting/Underfitting 문제 발생 가능.
* **StandardScaler** : Scale all features for computation



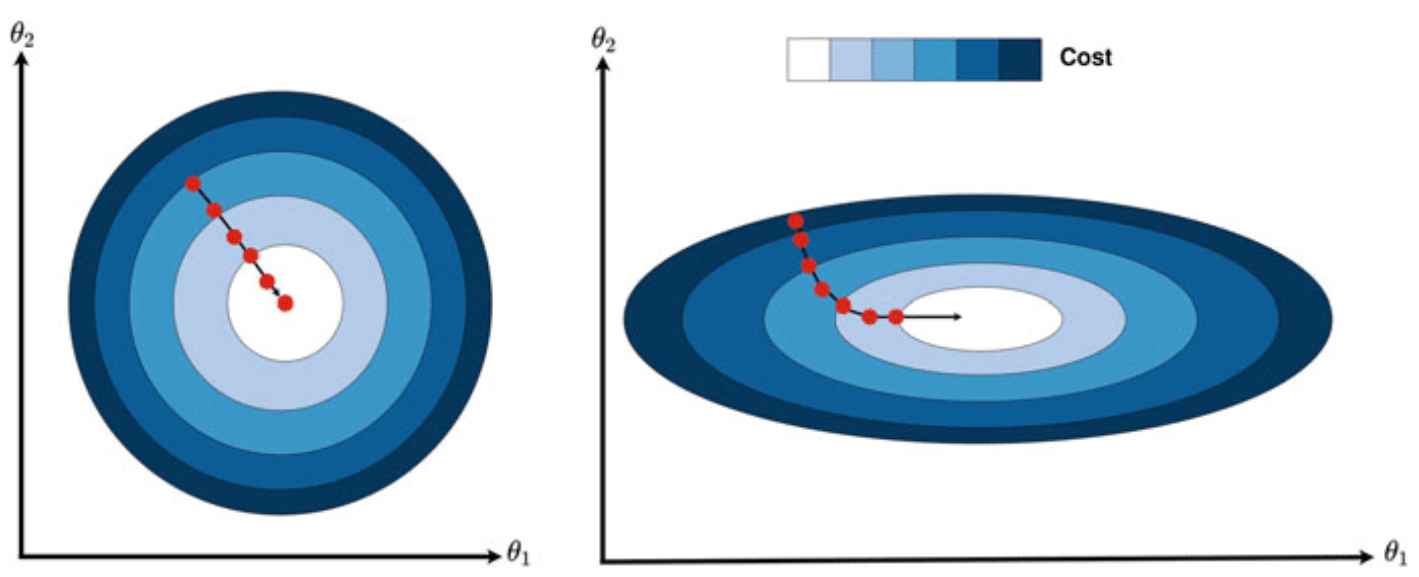
# Multi-Linear Regression

* x is row vector (cluster of features)
* Multiplying w, x bar makes **dot product**
* Python uses ‘np.dot(w,x)’ for this calculation.

@ Gradient Descent Algorithm

# Feature Scaling

* If the parameter's range gap is too high, the weight gap is also high.
* So, not all features can't be applied appropriately.
* Feature scaling solves this problem.
* goal : minimize J(Cost function)
* Automatic convergence test : after iterations, find if J is lower than ε(goal point)

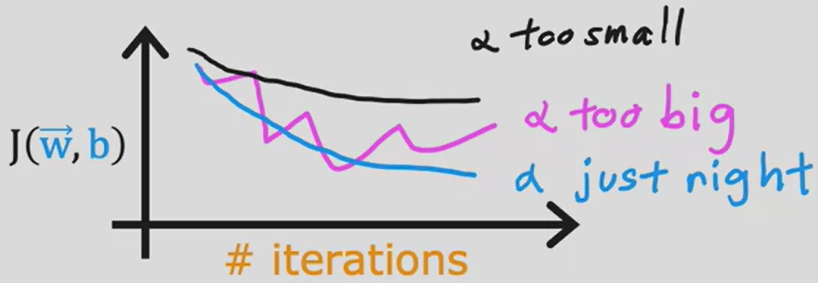


# Typical Normalization

* Mean : Place all data between CoVar [-1,1]
* Z-score : Use SVD

# How to choose the α(Learning Rate)?

* Over-fitting : If it is too high, cost doesn’t decrease consistently.
* Under-fitting : If it is too small, too high iterations occur.



# Polynomial Regression

* Linear models aren't always right. Polynomial regression helps it.
* Square, Root value ..etc are used
* How to decide? : Lesson II (Algorithm)

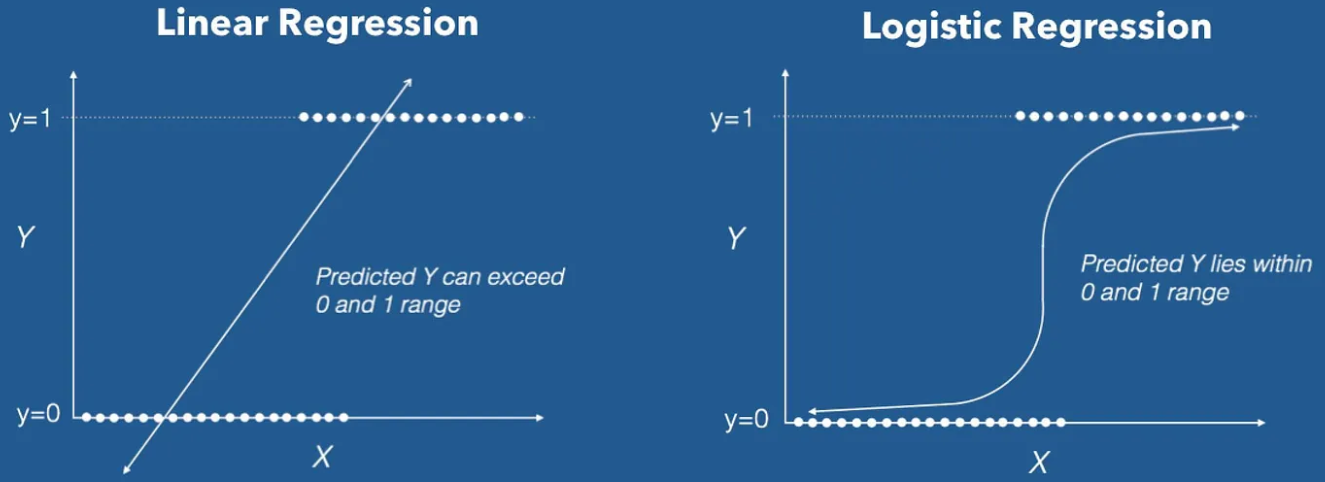
@ Classification with Logistic Regression

# Classification

* Use binary classification
* Decision boundary : classified boundary line

# Logistic Regression

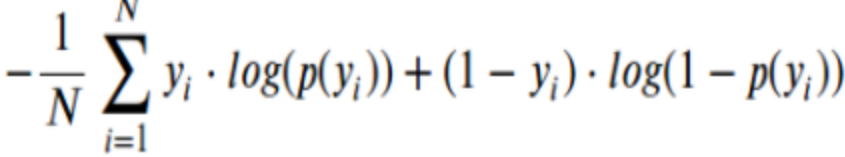
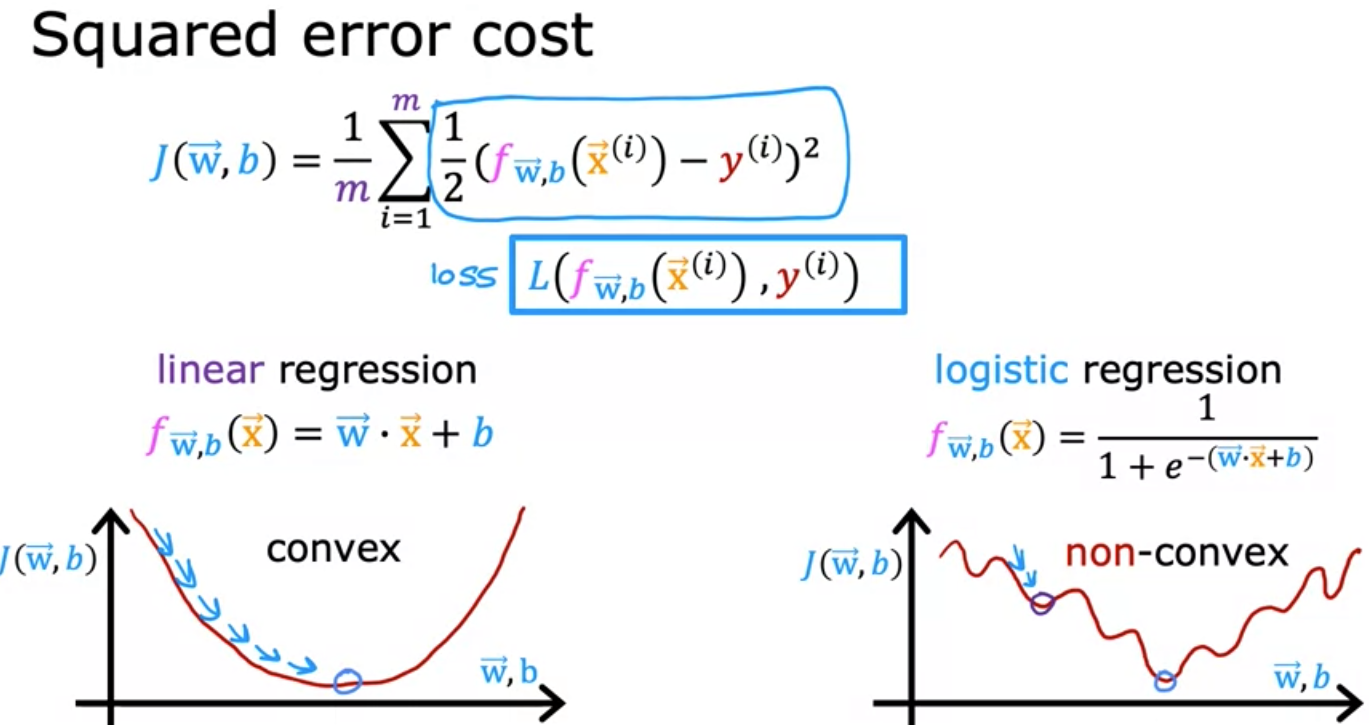
* Regression을 이용해 boundary를 추정함.
* Input feature의 sum(weight)을 측정한 후, error(오차)를 측정한다.
* Complement Linear model’s classification problem.



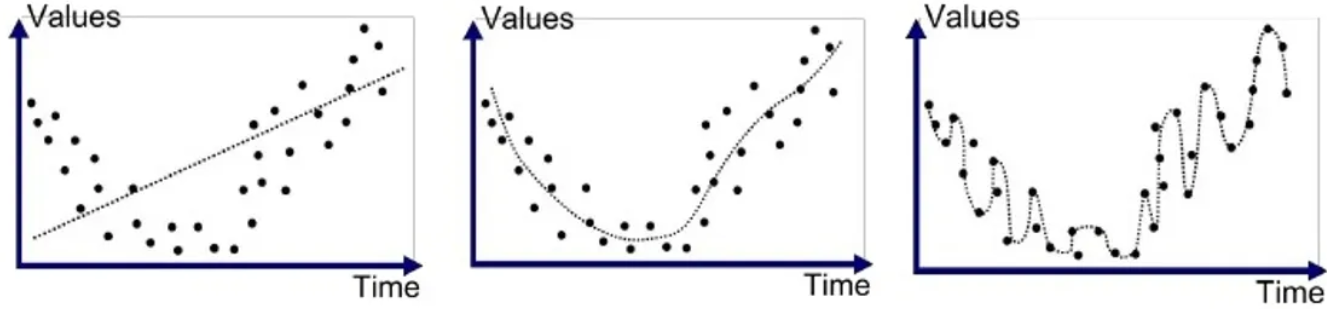
# Decision boundary

* If function f is sigmoid, when x>0, p(Y=1)>p(Y=0). Thus, Classification is available.
* If input(x) is ‘wx+b’ or more polynomial object, the boundary can be redefined by x.
* More polynomial input(x) means more variable features.

# Cost function for logistic regression

* R) In gradient descent, f(cost)’s shape was convex, so goal was finding a(‘f’(a)=0’).
* But in Sigmoid(more polynomial) function, too many gradient=0 point exist
* So, redefining the cost function is needed.

@ The problem of overfitting and solution

* Overfitting : Fit to the model extremely well
* Solution for Overfitting : Collect more data, Include/Exclude features, Regularization

# Regularization

* Decrease overly large parameter’s size -> balance(weight)
* L1(Lasso), L2(Ridge), Elastic Net