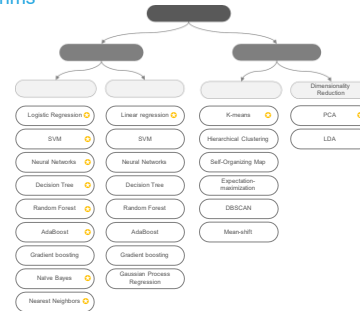


getting started with Machine Learning

:: Now let's study some popular algorithms



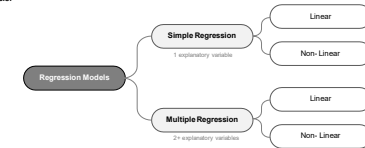
:: Linear Regression

Regression Analysis

ML models for regression problems predict a numeric value. Examples of Regression Problems

- What will the temperature be in Seattle tomorrow?
- For this product, how many units will sell?
- What price will this house sell for?

Type of regression models:



Part 2

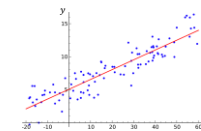
The well-known algorithms

Linear Regression

:: Linear
Regression

What's Linear regression ?

Linear regression is a linear approach for modeling the relationship between a dependent variable y and explanatory variables (or independent variables) denoted x .



- It's a supervised algorithm that learns from a set of training samples.
- Each training sample has one or more input values and a single output value.
- The algorithm learns the line, plane or hyper-plane that best fits the training samples.
- Use the learned line, plane or hyper-plane to predict the output value for any input sample.
- Because linear regression models are simple to interpret and easy to train, they are often the first models to be fitted to a new dataset?
- Linear regression can be used as a baseline for evaluating other, more complex, regression models?

© source: https://en.wikipedia.org/wiki/Linear_regression
1.2.2024/10/31 10:00:00 AM

:: Linear Regression

What's Linear regression used for ?

Linear regression has many practical uses. Most applications fall into one of the following two broad categories: ¹

1. Making Prediction

Based on the relationship we learn from the sample data, the linear regression model can be used to forecast an unknown value.

In other words "Linear Regression" is a method to predict dependent variable y based on values of independent variables x_1, x_2, \dots, x_n .

Example: What's the sales value over the next year ?

2. Establishing if there is a relationship between variables

Given a variable y and a number of variables x_1, x_2, \dots, x_n that may be related to y , linear regression analysis can be applied to quantify the strength of the relationship between y and the x_i , to assess which x_i may have no relationship with y at all, and to identify which subsets of the x_i contain redundant information about y . ²

More specifically, establish if there is a statistically significant relationship between two variables. For example, income might have significant relationship with education level.

¹Source: <https://www.khanacademy.org/machine-learning/a/linear-regression-what-it-is-and-how-it-works/a/linear-regression-what-it-is-and-how-it-works>

:: Linear Regression

Type of Linear Regression

Simple linear regression

only one explanatory variable

Multiple linear regression

more than one explanatory variable

:: Linear Regression

Example: Predict house's price using linear regression

Suppose we have a dataset giving the living areas and prices of 21,813 houses from House Sales in King County, USA. Given data like this, we can learn to predict the prices of other houses in King County.

Living Area (Foot ²)	Price (\$)
1180	221,900
2070	538,000
770	180,000
1980	604,000
1880	510,000
5400	1,225,000
1715	257,500
1600	291,800
1750	229,500
1880	323,000
3960	692,000
1140	460,000
1400	310,000
1370	480,000
1810	530,000
—	—
x	y

How much for this house ?

living area = 4873 foot²

:: Linear Regression

Type of Linear Regression

Simple linear regression

only one explanatory variable

Multiple linear regression

more than one explanatory variable

:: Linear Regression

Let's take a look at Simple Linear regression

Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables: ¹

- One variable, denoted y , is regarded as the **predictor, explanatory, or independent variable**.
- The other variable, denoted x , is regarded as the **response, outcome, or dependent variable**.

Simple Linear Regression is termed as simple because there is only independent variable. The relationship between the input x and outcome y can be represented in a form of following equation

Intercept

slope of the line

$\hat{y} = \theta_0 + \theta_1 x$

dependent variable

independent variable

Linear equation example

$\hat{y} = \theta_0 + \theta_1 x$
 $\hat{y} = 3.7218 + 0.4794x$
 $\theta_1 = \Delta y / \Delta x = 0.4794$

The slope of 0.4794 means that each increase of one unit in X , we predict the average of Y to increase by an estimated 0.4794 units.

¹Source: <https://www.khanacademy.org/machine-learning/a/linear-regression-what-it-is-and-how-it-works/a/linear-regression-what-it-is-and-how-it-works>

:: Linear Regression

Example: Predict house's price using linear regression

Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables ¹. In this example, the price has positive correlation with the size ($r=0.49$)

Living Area (Foot ²)	Price (\$)
1180	221,900
2070	538,000
770	180,000
1980	604,000
1880	510,000
5400	1,225,000
1715	257,500
1600	291,800
1750	229,500
1880	323,000
3960	692,000
1140	460,000
1400	310,000
1370	480,000
1810	530,000
—	—
x	y

correlation coefficient $r = 0.49$


The correlation coefficient r is a valuable numerical measure of association between two variables. The correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

2

Linear Regression

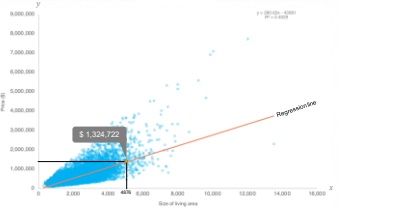
Example: Predict house's price using linear regression

Then we can identify best-fitting relationship (line) between the variables. The regression line is the "best-fit" line and has the formula $y = \theta_0 + \theta_1 x$. Thus, given a value of X , we can predict a value of Y .



Size of living area = 4575 ft²

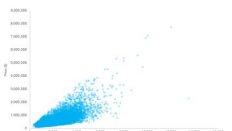
Based on this regression line, we can predict the price for a house with a certain size of living area



Linear Regression

The goal of linear regression is to find the best fit line

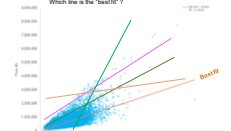
The goal is to find estimated value for the parameters θ_0 and θ_1 which would provide the "best" fit for the data points within the training set.



$h(x) = \theta_0 + \theta_1 x$

Parameters θ_0, θ_1

Different parameters θ_0 and θ_1 will result in different lines



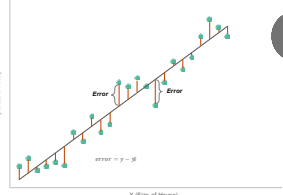
Which line is the "best fit"?

Best fit line: $\theta_0 = 43381, \theta_1 = 283.82$

Linear Regression

How do we come up with "best" parameter value that corresponds to a good fit to the data?

The idea is to choose the appropriate parameter θ_0 and θ_1 so that the predicted value \hat{y} is close to y for training example (x, y) . In other words, the sum of squared error should be least.



error = $y - \hat{y}$

$h(x) = \theta_0 + \theta_1 x$

How to find the appropriate parameter θ_0 and θ_1 in order to minimize the error - i.e. cost function $J(\theta_0, \theta_1)$?

To minimize $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$

Cost function (also known as Loss Function)

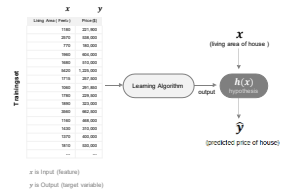
- m is the number of training instances
- \hat{y}_i is the predicted value
- y_i is the actual value

Video: Cost Function Intuition #1 by Andrew Ng (11 mins)
<https://www.youtube.com/watch?v=7DdG0aDd0W4>

Linear Regression

Actually linear regression model guesses a hypothesis function $h(x)$ from training set

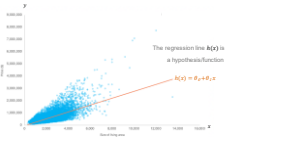
Given a training set, the model learn a function $h(x)$ which can be used for predicting the new/unknown data.



Learning Algorithm

output $h(x)$

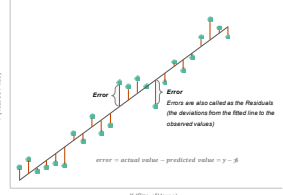
input x is Input (features)
 y is Output (target variable)



Linear Regression

How do we know which line is the "best fit" ?

The good line is one that minimizes the sum of the "squared differences" between the points and the regression line. Usually some data points will deviates from the line somehow (if a point lies on the fitted line exactly, then its vertical deviation is 0).



error = actual value - predicted value = $y - \hat{y}$

Least Squares Method

The optimization goal is to minimize the sum of "squared error" (least squares). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values.

Goal: to minimize the sum of squared error

Sum of Squared Errors = $\sum_{i=1}^m (y_i - \hat{y}_i)^2$

where $\hat{y} = \theta_0 + \theta_1 x$
 m is the number of training instances

The smaller the sum of squared differences, the better the fit of the line to the data.

Linear Regression


The computation methods for estimating the parameters

There are a couple of approaches to find the value of parameter θ that minimizes the cost function $J(\theta)$.

The approaches to solving $\theta = \underset{\theta}{\operatorname{argmin}} J(\theta)$

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

- Normal Equation (Closed Form)**
It's a method to solve for θ analytically.
Using a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model/parameters that minimize the cost function over the training set).
It's suitable for small feature set (e.g. < 1000 features).
- Gradient Descent**
Using an iterative optimization approach, called Gradient Descent (GD), that gradually iterate the model/parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method.
Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory.



How to solve for the parameters?

:: Linear Regression

The computation methods for estimating the parameters

There are a couple of approaches to find the value of parameter θ that minimizes the cost function $J(\theta)$.

The approaches to solving $\theta := \underset{\theta}{\operatorname{argmin}} J(\theta)$

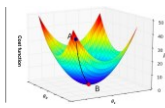
Normal Equation (Closed Form)
It's a method to solve for θ analytically.

Using a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set).

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

Gradient Descent
Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method.

Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory.

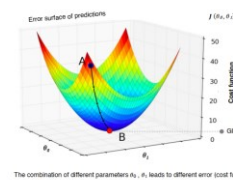
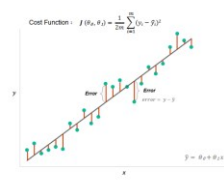


:: Linear Regression

Gradient descent is an iterative algorithm to find a minimum of a function

The goal of gradient descent is to start on a random point on this error surface, and find out the best parameters (θ_0, θ_1) which yields the minimum value of the cost function. In other words, the best parameters (θ_0, θ_1) corresponds to the the lowest point on the error surface.

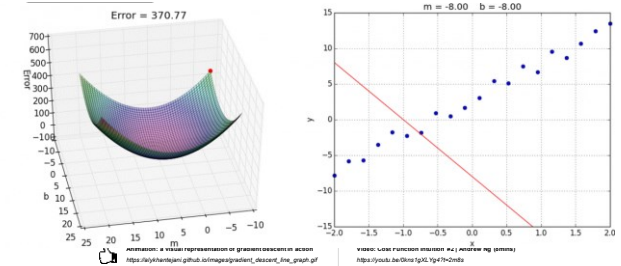
The cost function $J(\theta_0, \theta_1)$ can be visualized as a 3D plot



The combination of different parameters θ_0, θ_1 leads to different error (cost function). The error surface of the predictions corresponds to the different value of parameters.

:: Linear Regression

Example : find the best-fit line through Gradient Descent algorithm



:: Linear Regression

Gradient Descent

Gradient Descent is one of the most popular optimization algorithms used in Machine Learning and Deep Learning. Many machine learning algorithms such as linear regression, logistic regression, neural networks use Gradient Descent.



If you're blindfolded hiker, how to get to the lowest place?

We can illustrate Gradient Descent optimization as a process that a blindfolded hiker wants to go down a mountain as soon as possible.

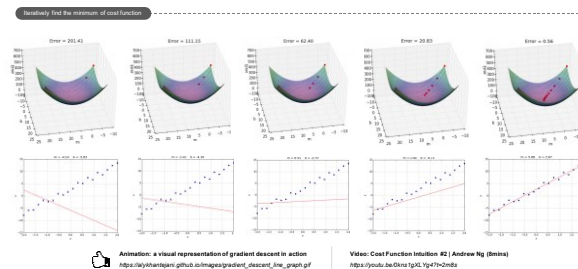
Suppose you are at the top of a mountain, and you have to go to the valley which is at the lowest point of the mountain. But now you are blindfolded and you can only feel the slope of the ground below your feet. So, what approach will you take to reach the lake?

A good strategy to get to the bottom of the valley quickly is to go downhill in the direction of the steepest slope. This is exactly what Gradient Descent does: it measures the local gradient of the error function with regards to the parameter vector θ , and it goes in the direction of descending gradient. Once the gradient is zero, you have reached a minimum!

Video: simple introduction of Gradient Descent (2 mins)
<https://www.youtube.com/watch?v=anmumG2Ae80&list=PL01iYtPyZWde>

:: Linear Regression

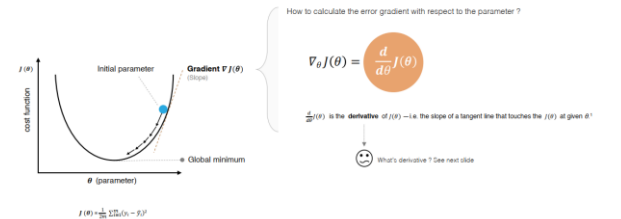
Example : find the best-fit line through Gradient Descent algorithm



:: Linear Regression

What's gradient?

Gradient is another word for "slope". A gradient is the slope of a function at a specific point. The gradient of loss function/cost function is equal to the derivative (slope) of the curve.



:: Linear Regression

What's derivative & partial derivative?

It's important to understand the basic concept of derivative. It's strongly recommended to watch the following videos if you have already forgotten those concept.



$$\frac{\partial J(\theta)}{\partial \theta}$$



1 Derivative
www.youtube.com/watch?v=5u1U1u1u1u1



2 More Derivative Examples
www.youtube.com/watch?v=5u1U1u1u1u1



3 Derivatives of Activation Functions
www.youtube.com/watch?v=5u1U1u1u1u1



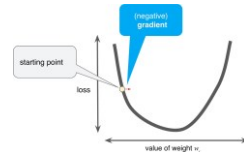
4 Partial Derivatives, Introduction
www.youtube.com/watch?v=5u1U1u1u1u1

:: Linear Regression

How does Gradient descent work? - continued

The gradient of loss is equal to the derivative (slope) of the curve. Note that a gradient is a vector, so it has both of the following characteristics:

- a direction
- a magnitude



The gradient always points in the direction of steepest increase in the loss function.

The gradient descent algorithm takes a step in the direction of the **negative gradient** in order to reduce loss as quickly as possible.

:: Linear Regression

How does Gradient descent work? - continued

More generally, the cost function depends on more than one parameters. In Simple Linear Regression, there are 2 parameters θ_0, θ_1 .

Gradient Descent algorithm

The parameter are iteratively updated in the following equation:

repeat until convergence {

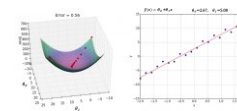
 # Simultaneously update θ_0, θ_1

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

}

- Start with some random value θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta)$ until we hopefully end up at a minimum



Youtube Video: Gradient Descent by Andrew Ng (11 mins)
<https://youtu.be/F6GSRDd8-Cg>



:: Linear Regression

How does Gradient descent work?

When there are multiple parameters, the gradient is a vector of partial derivatives with respect to the parameters. For the sake of simplicity, consider the cost function $J(\theta)$ that depends on only one parameter θ .

Gradient Descent algorithm

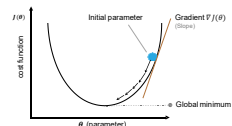
The parameter are iteratively updated in the following equation:

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

Learning rate (step size)

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

1. Pick a value for the learning rate α
2. Start with a random point θ
3. Calculate the gradient $\nabla J(\theta)$ at the point θ . Follow the opposite direction of gradient to get new parameter θ_{new}
4. Repeat until the cost function converges to the minimum



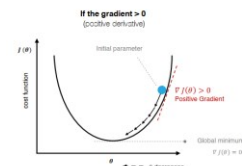
In this example, initially the slope is large and positive. So, in the update equation, θ is reduced. As θ keeps getting reduced, notice that the gradient also reduces, and hence the updates become smaller and smaller and eventually, it converges to the minimum.

Source: <https://the-researcher.com/2020/gradient-descent-7349702/>

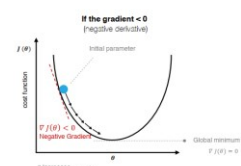
:: Linear Regression

How does Gradient descent work? - continued

For the sake of simplicity, consider the cost function $J(\theta)$ that depends on only one parameter θ .



Because the gradient is positive value, the parameter update will be negative and will reduce the current values of θ to converge to the minimum

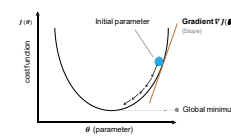


Because the gradient is negative value, the parameter update will be positive and will increase the current values of θ to converge to the minimum

:: Linear Regression

What's Learning rate?

An important parameter in Gradient Descent is the size of the steps, determined by the learning rate hyper-parameter.



$$\theta_1 := \theta_0 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

Learning rate α
(step size)

Gradient

the gradient gives the direction of the steepest ascent

The gradient vector has both a direction and a magnitude. Gradient descent algorithms multiply the gradient by a scalar known as the **learning rate** (also sometimes called **step size**) to determine the next point.

For example, if the gradient magnitude is 2.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.025 away from the previous point.

:: Linear Regression

Learning rate is a hyperparameter

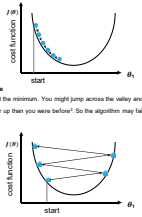
Hyperparameters are the knobs that programmers tweak in machine learning algorithms. Most machine learning programmers spend a fair amount of time tuning the learning rate.

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

the gradient gives the direction of the steepest ascent

Learning rate α
(Step size)

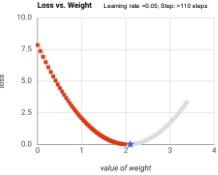
- If the learning rate is too small
gradient descent would take long time to converge and can be very slow
- If the learning rate is too large
gradient descent can overshoot the minimum. You might jump across the valley and end up on the other side, possibly even higher up than you were before! So the algorithm may fail to converge, or even diverge.



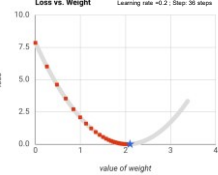
:: Linear Regression

Try different learning rates by yourself in Google's interactive demo

Experiment with different learning rates and see how they affect the number of steps required to reach the minimum of the loss curve.



Loss vs. Weight Learning rate = 0.05, Step = 110 steps



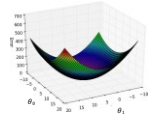
Loss vs. Weight Learning rate = 0.2, Step = 36 steps

Live playground from google
<https://developers.google.com/machine-learning/crash-course/linear/graph>

:: Linear Regression

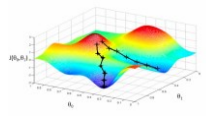
Gradient Descent for convex function & non-convex function

If your error surface is convex and with an appropriate learning rate (i.e. not too high), then convergence to the global minima is guaranteed. However, in many real-world problems, the error surface is generally highly non-convex, meaning there are a plethora of local minima. In this situation, depending on the initial parameters we choose, the solution could converge to a local minima that is not necessarily the global minima.¹



Convex function : Just one minimum

- Fortunately, the MSE cost function for a Linear Regression model happens to be a convex function.¹
- This implies that there are no local minima, just one global minimum.¹



Non-Convex function: More than one minimum

¹ $J(\theta)$ is usually non-convex in many real-world problem, gradient descent

- Can be susceptible to local minimum?
- Strong dependency on initial values. Different starting points end up with completely different local minimum.
- In practice converging to local minimum is not usually a huge problem!

:: Linear Regression

How to choose learning rate ?

Choosing a suitable value for learning rate is a sort of art. So try different values for learning rate and plot the cost function. You can see how it is performing.


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

the gradient gives the direction of the steepest ascent

Learning rate α
(Step size)

- If α is too small: slow convergence
- If α is too large: $J(\theta)$ may not decrease on every iteration, may not converge.

The most commonly used rates are : 0.001, 0.003, 0.01, 0.03, 0.1, 0.3

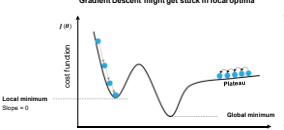


Youtube Video: Learning Rate by Andrew Ng (8 mins)
<https://youtube.com/watch?v=4j5P9UuhyYw>

:: Linear Regression

Gradient Descent's issue

Not all cost functions look like nice regular bowls. There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult.¹



Gradient Descent might get stuck in local optima


- 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.¹
- If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.¹

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

if as the gradient (slope) at the local minimum is 0

So $\theta_1 := \theta_1 - \alpha \cdot 0$ if so the gradient descent might get stuck in local minimum

:: Linear Regression



6

:: Linear Regression

The computation methods for estimating the parameters

There are a couple of approaches to find the value of parameter θ that minimizes the cost function $J(\theta)$.

The approaches to solving $\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$

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

• Normal Equation (Closed Form)

It's a method to solve for θ analytically.

Using a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set).

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

• Gradient Descent

Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method.¹

Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory.



1. Gradient Descent: Intro, Motivation, and Derivation | Andrew Ng

:: Linear Regression

Normal Equation (advanced topic)

To find the value of θ that minimizes the cost function, there is a closed-form solution—in other words, a mathematical equation that gives the result directly. This is called the Normal Equation.

• Equation of multivariate linear regression :

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

How we'll represent the problem in matrix notation :

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

So we can now rewrite the hypothesis function as

$$h_{\theta}(x) = \theta^T x$$

• Set derivatives equal to zero and solve for parameters $\theta_0, \theta_1, \dots, \theta_n$

We will minimize $J(\theta)$ by explicitly taking its derivatives with respect to the θ_i 's, we set its derivatives to zero, and obtain the normal equations:

$$\frac{\partial}{\partial \theta} J(\theta) = 2X^T X \theta - 2X^T y = 0 \implies X^T X \theta = X^T y$$

Thus, the value of θ that minimizes $J(\theta)$ is given in closed form by the equation

$$\theta = (X^T X)^{-1} \cdot X^T \cdot y \quad \text{if } (X^T X)^{-1} \text{ is inverse of matrix } X^T X$$



The Normal Equation gets very slow when the number of features grows large (e.g., 100,000!).

Video : Normal Equation



<https://www.youtube.com/watch?v=8vU10110u0k>
<https://www.kurdtk.com/normal-equation/>

:: Linear Regression

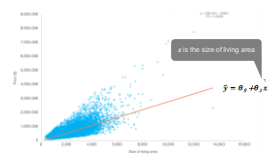
Multiple Linear Regression

In previous example for predicting house price, we only use the size of living area to predict the price. It's pretty simple but of very less use in real world scenarios.



Living Area (Feet ²)	Price(\$)
1180	221,000
2670	1,240,000
770	180,000
6600	550,000
4000	310,000
3400	1,100,000
4800	391,000
4710	307,000
1700	230,000
6600	550,000
3000	460,000
1100	490,000
1400	710,000
1070	400,000
9010	530,000
...	...

This simple linear regression model is only based on one feature "Living Area"



Source: Andrew Ng, Coursera

:: Linear Regression

The computation methods for estimating the parameters

There are a couple of approaches to find the value of parameter θ that minimizes the cost function $J(\theta)$.

The approaches to solving $\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$

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

• Normal Equation (Closed Form)

It's a method to solve for θ analytically.

Using a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set).

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

• Gradient Descent

Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method.¹

Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory.

In a simple linear regression the coefficients are calculated as:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}, \quad \hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

In a matrix notation, the coefficients are calculated as:

$$\hat{\beta} = (X^T X)^{-1} \cdot X^T \cdot y$$

1. Gradient Descent: Intro, Motivation, and Derivation | Andrew Ng

:: Linear Regression

Multiple Linear Regression

Multiple Linear regression is also known as Multivariate Linear Regression.

Simple linear regression
only one explanatory variable

Multiple linear regression
more than one explanatory variable



:: Linear Regression

Multiple Linear Regression

Generally one dependent variable depends on multiple factors. For example, the price of a house depends on many factors like the neighborhood it is in, size of it, no. of rooms, attached facilities, distance of nearest station from it, distance of nearest shopping area from it, etc.¹



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

Size (feet ²)	Number of Bedrooms	Number of Baths	Age of Home (years)	Price (\$100k)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

1. Source: <https://www.kaggle.com/andrewbriandata/multiple-linear-regression-challenge>

:: Linear Regression

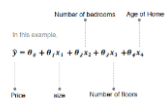
Multiple Linear Regression

This is quite similar to the simple linear regression model we have discussed previously, but with multiple independent variables contributing to the dependent variable. Each training sample has an x made up of multiple input values and a corresponding y with a single value. The equation of multivariate linear regression is as follow,

- Simple Linear Regression: $\hat{y} = h(x) = \theta_0 + \theta_1 x$
- Multiple Linear Regression: $\hat{y} = h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$

Example

Size (feet)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$100k)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...



:: Linear Regression

Multiple Linear Regression

For multiple linear regression model,

We define the **cost function** (aka. Loss function):

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \quad \left| \quad m = \text{number of training instances} \right.$$

$$\text{where } \hat{y} = h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n = \sum_{j=0}^n \theta_j x_j$$

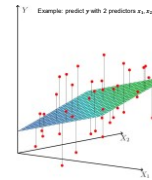
\Downarrow

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (y_i - (\theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \dots + \theta_n x_{in}))^2$$

\Downarrow

How to find the appropriate parameter $\theta_0, \theta_1, \dots, \theta_n$ in order to minimize the cost function/loss function $J(\theta)$?

- Normal Equation
- Gradient Descent



we want the hyperplane that "best" fits the training samples. In other words, we seek the linear function of X that minimizes the sum of squared residuals (error) from Y .

:: Linear Regression

Multiple Linear regression

Use Gradient Descent to find the value of parameters θ that minimize the cost function $J(\theta)$.

Gradient Descent for multiple Linear Regression :

repeat until convergence {

θ Simultaneously update θ_j for every $j=0,1,\dots,n$

$$\theta_j := \theta_j - \alpha \left(\frac{1}{2m} \sum_{i=1}^m \frac{\partial}{\partial \theta_j} J(\theta) \right)$$

}

$\frac{\partial}{\partial \theta_j} J(\theta)$ is the partial derivatives of the cost function with respect to the parameters $\theta_0, \theta_1, \dots, \theta_n$

Cost Function:

$$\text{Hypothesis: } h_{\theta}(x) \triangleq \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

$$\text{Parameters: } \theta = (\theta_0, \theta_1, \dots, \theta_n)$$

$$\text{Cost Function: } J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$\text{Gradient } \nabla_{\theta} J(\theta) :$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$x_j^{(i)}$ is the j th features of i th observation

Gradient Descent:

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right)$$

(Simultaneously update θ_j for every $j=0,1,\dots,n$)

:: Linear Regression

Multiple Linear Regression

For convenience of notation, we can define $x_0=1$. Thus, we simplify the equation of multiple linear regression as follow.

Equation :

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

For convenience of notation, define $x_0=1$, then

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

In the multiple regression setting, because of the potentially large number of predictors, it is more efficient to use matrices to define the regression model and the subsequent analyses.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

$$\hat{y} = \theta^T X$$

\Downarrow

Linear Regression With Multiple Variables (Matrix)

https://youtu.be/Gf2037220s

(by Andrew Ng)

:: Linear Regression

Multiple Linear Regression

Example:, we can use **Normal Equation** to find the value of the coefficients/parameters θ that minimizes the cost function

Example

Size (feet)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$100k)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

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

- The inputs can be represented as an X matrix in which each row is sample and each column is a dimension.
- The outputs can be represented as y matrix in which each row is a sample.

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\hat{y} = \theta^T X$$

$$\theta = (X^T X)^{-1} X^T \cdot y \quad \theta \text{ value of } \theta \text{ that minimizes the cost function can be solved by this equation}$$

:: Linear Regression

Multiple Linear regression

Use Gradient Descent to find the value of parameters θ that minimize the cost function $J(\theta)$

Gradient Descent for multiple Linear Regression :

repeat until convergence {

$$\theta_j := \theta_j - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right)$$

(Simultaneously update θ_j for every $j=0,1,\dots,n$)

}

Video: Gradient Descent For Multiple Variables (Matrix)

https://youtu.be/Gf2037220s

(by Andrew Ng)

Example

Size (feet)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$100k)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

$$\hat{y} = h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

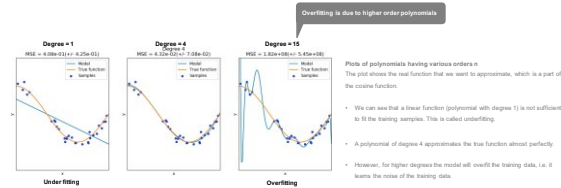
$$\begin{cases} \theta_0 := \theta_0 - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)} \right) \\ \theta_1 := \theta_1 - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) \\ \theta_2 := \theta_2 - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) \\ \theta_3 := \theta_3 - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_3^{(i)} \right) \\ \theta_4 := \theta_4 - \alpha \left(\frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_4^{(i)} \right) \end{cases}$$

:: Linear Regression

In regression, overfitting occurs frequently

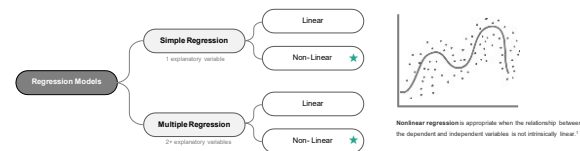
The more parameters or features in your model, the more freedom it has to fit the data. However, the model is also more prone to overfitting the training data.

Example: the models have polynomial features of different degrees.



:: Nonlinear Regression

Type of Regression Model



Logistic Regression

:: Linear Regression

How to reduce overfitting problem ?

There are some strategies to address overfitting problem in Linear regression model

Reduce model's complexity

- It's prone to overfitting if there are lots of features but with very little training data

Throw away some of features. E.g.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \rightarrow \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

- Reduce the degree of polynomial

Try lower degree in case of overfitting. E.g.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4 \rightarrow \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$$

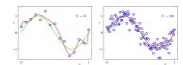
Use L1 / L2 regularization

- Ridge regression** is a regularized version of Linear Regression. It uses L2-norm penalty. Ridge regression includes all of the features in the model. As the value of alpha/lambda increases, the coefficient shrinks.

- Lasso regression** is a regularized version of Linear Regression. It uses L1-norm penalty. L1 will push certain weights to be exactly 0. Lasso Regression automatically performs feature selection and outputs a sparse model.

Get more data

For a given model complexity, the overfitting problem becomes less severe as the size of the data set increases.²



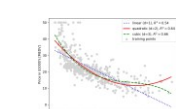
N = 15 data points (left plot) and N = 100 data points (right plot). We see that increasing the size of the data set reduces the overfitting problem.³

:: Nonlinear Regression

What's nonlinear regression?

Nonlinear regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables.¹

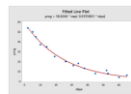
Polynomial regression is NOT considered as non-linear regression



Although this model allows for a nonlinear relationship between Y and X, polynomial regression is still considered linear regression since it is linear in the regression coefficients, $\beta_0, \beta_1, \dots, \beta_n$.²

"Nonlinear" refers to a fit function that is a nonlinear function of the parameters. For example, the equation $y = \frac{a}{b + x}$ is a nonlinear function of the fitting parameters. One simple nonlinear model is the exponential regression model³

$$y_i = \beta_0 + \beta_1 \exp(\beta_2 x_{i1}) + \dots + \beta_p x_{ip} \exp(\beta_2 x_{i1})$$



Some other examples of equation for nonlinear regression models:

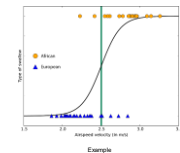
$$\begin{aligned} y_i &= \frac{\beta_0 \beta_1 x_i}{1 + \beta_1 \beta_2 x_i^2} + \epsilon_i \\ y_i &= \frac{\beta_0 + \beta_1 x_i}{1 + \beta_2 x_i^2} + \epsilon_i \\ y_i &= \beta_0 + (\beta_1 + \beta_2 x_i) e^{\beta_3 x_i - \beta_4} + \epsilon_i \end{aligned}$$

:: Logistic Regression

What's logistic regression ?

Logistic regression is a simple classification algorithm that can predict the probability of a binary response belonging to one class or the other.

- Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class. (e.g., what is the probability that this email is spam?). This makes it a binary classifier.¹
 - If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"). For example, if the model infers a value of 0.932 on a particular email message, it implies a 93.2% probability that the email message is spam.²
 - If the estimated probability is less than 50%, it predicts that it does not belong to that class (i.e., it belongs to the negative class, labeled "0").
- Because of its simplicity, logistic regression is commonly used as a starting point for binary classification problems.³ Logistic regression can be used as a baseline for evaluating more complex classification methods



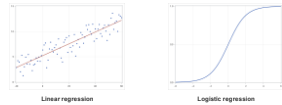
A logistic regression to two-class data with just one feature

:: Logistic Regression

Don't get confused by its name! It's a classification rather than regression algorithm

Although it confusingly includes 'regression' in the name, logistic regression is actually a powerful tool for two-class and multiclass classification.¹

- It's fast and simple. The fact that it uses an 'S'-shaped curve instead of a straight line makes it a natural fit for dividing data into groups.²



Linear regression

Logistic regression

- In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.
- It predicts the probability, its output values lies between 0 and 1 (as expected).

Logistic Regression – Introduction



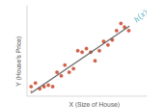
YouTUBE Video (13min)
<https://www.youtube.com/watch?v=343u3K7AQ0>

:: Logistic Regression

Let's recall the Linear regression hypothesis $h_{\theta}(x)$ first

- Univariate linear regression (Simple linear regression)

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



- multivariate linear regression

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = \sum_{i=0}^n \theta_i x_i$$



$$h_{\theta}(x) = \theta^T x \quad \text{if vectorized form}$$

But linear regression is not a good approach for classification problem
 Although we can use linear regression to classify the dataset, but often it's not a good idea to do that. One of the key reason is as follows,

- In linear regression the output value is a continuous range. So linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1).

Andrew Ng explains this in a short video.



Logistic Regression (Classification)
<https://www.youtube.com/watch?v=343u3K7AQ0>

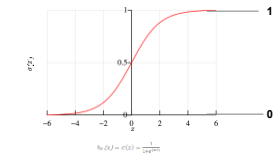
:: Logistic Regression

But what's Logistic/Sigmoid Function ?

"sigmoid" or "logistic" function – it is an S-shaped function that "squashes" the value of $\theta^T x$ into the range [0, 1] so that we may interpret $h_{\theta}(x)$ as a probability.¹ A logistic function or logistic curve is a common "S" shape (sigmoid curve).

- The sigmoid $\sigma(x)$ takes the S-curve shape.

Here is a plot showing $\sigma(x)$



Squash the value x into [0, 1], using sigmoid function

Sigmoid function squashes the value (any value) and gives the value between 0 and 1

- $\sigma(x) \geq 0.5$ when $x \geq 0$
- $\sigma(x)$ tends towards 1 as $x \rightarrow \infty$
- $\sigma(x) \leq 0.5$ when $x \leq 0$
- $\sigma(x)$ tends towards 0 as $x \rightarrow -\infty$

$\sigma(x)$, and hence also $h_{\theta}(x)$, is always bounded between 0 and 1.

:: Logistic Regression

Logistic regression: Calculating a Probability

Many problems require a probability estimate as output. Logistic regression is an extremely efficient mechanism for calculating probabilities.¹ You might forget how the conditional probability is written, don't worry about it. Let's learn it through the following examples.

It tells us how often A happens given that B happens

$$P(A | B)$$

$P(A | B)$ is "Probability of A given B", the probability of A given that B happens.

For example:

- $P(\text{Fire} | \text{Smoke})$ means how often there is fire when we see smoke.
- $P(\text{Smoke} | \text{Fire})$ means how often we see smoke when there is fire.
- $P(\text{bark} | \text{night}) = 0.05$ mean the probability that a dog will bark during the middle of the night is 5%

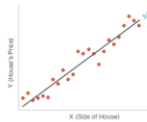
Well, how to calculate the probability? See next slides for the details

:: Logistic Regression

Now we take a look at the logistic regression hypothesis : $h_{\theta}(x)$

Linear regression hypothesis $h_{\theta}(x)$

- Univariate linear regression (Simple linear regression)



$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

- Multivariate linear regression

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = \sum_{i=0}^n \theta_i x_i$$



$$h_{\theta}(x) = \theta^T x \quad \text{if vectorized form}$$

Logistic regression hypothesis $h_{\theta}(x)$

In a binary classification we use a different hypothesis.

We predict the probability that a given example belongs to the "1" class versus the probability that it belongs to the "0" class

- 1 (positive) : e.g. malignant tumor
- 0 (negative) : e.g. non-malignant tumor

We want $0 \leq h_{\theta}(x) \leq 1$

So we need to find a function for our hypothesis so that the output is bounded (0,1). Sigmoid function can help us.

$$h_{\theta}(x) = \theta^T x$$



$$h_{\theta}(x) = \frac{\sigma}{1 + \exp(-\theta^T x)}$$



$$h_{\theta}(x) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$

Use sigmoid function to "transform" the linear regression equation $h_{\theta}(x) = \theta^T x$

Sigmoid/Logistic function: $\sigma(x) = \frac{1}{1 + \exp(-x)}$

:: Logistic Regression

More about Logistic/Sigmoid Function

The benefit of this curve is that the input values can range from $-\infty < x < \infty$ whilst the output ranges from 0 to 1, exactly the range for probability values. Hence its common use as a transfer function.¹

$$\sigma(t) = \frac{1}{1 + e^{(-t)}}$$



e^x is **exponential function**, which is also denoted by $\exp(x)$.

The constant $e = 2.71828\dots$, the base of the natural logarithm.

The value of t

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

if you're not familiar with exponential function $\exp(x)$, you can try to calculate it easily in Excel which provides this function.

:: Logistic Regression

We can treat the output of logistic regression hypothesis $h_\theta(x)$ as estimated probability

"sigmoid" or "logistic" function – it is an S-shaped function that "squashes" the value of $\theta^T x$ into the range $[0, 1]$ so that we may interpret $h_\theta(x)$ as a probability.¹

Linear regression hypothesis

$$h_\theta(x) = \theta^T x$$

Logistic regression hypothesis

When our hypothesis $h_\theta(x)$ outputs some number, we can treat that number as **estimated probability** that $y = 1$ (i.e. positive) on input x .

- For example, if we use size of tumor to predict if the tumor is malignant, and suppose the hypothesis outputs 0.7. (i.e. $h_\theta(x) = 0.7$). \rightarrow then I'm going to tell my patient that the tumor has a 70% chance, or a 0.7 chance of being malignant.²

To write this out in math, I'm going to interpret my hypothesis output as $h_\theta(x) = P(y = 1 | x)$

$$P(y = 1 | x) = h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

¹ e.g., $P(y = 1 | x = 1)$ is the probability of "incoming email x is spam."

$$P(y = 0 | x) = 1 - P(y = 1 | x) = 1 - h_\theta(x)$$

² e.g., $P(y = \text{normal email} | x)$ is the probability of "incoming email x is normal email"

³ note: $P(y = 1 | x) \times P(y = 0 | x) = 1$

<https://www.youtube.com/watch?v=7T6a296u8A>

:: Logistic Regression

How to find the value of θ for fitting the model? -1

Our goal is to search for a value of θ so that model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$). Instead of finding the best fitting line by minimizing the squared residuals as in ordinary least squares, Logistic regression model uses a different approach with logistic – Maximum Likelihood Estimation.

Logistic Regression loss function (log loss)

The loss function for linear regression is squared loss. The loss function for logistic regression is **Log Loss**. The loss function over the whole training set is simply the average loss over all training instances. The loss function (known as cost function) for the θ parameters can be defined as:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Why should we use Log Loss as the cost function?

Andrew Ng explains the reason in his excellent video tutorial.

YouTube Video: Logistic Regression (Cost Function) (7:06:00)

<https://www.youtube.com/watch?v=7T6a296u8A>

- $y^{(i)}$ is the label in a labeled example. Since this is logistic regression, every value of $y^{(i)}$ must either be 0 or 1.
- $h_\theta(x)$ is the predicted value (somewhere between 0 and 1), given the set of features in x .

:: Logistic Regression

Regularization in Logistic Regression

Regularization is extremely important in logistic regression modeling. Without regularization, the asymptotic nature of logistic regression would keep driving loss towards 0 in high dimensions. Consequently, most logistic regression models use one of the following two strategies to dampen model complexity.¹

L₂ regularization

L₂ regularization helps drive outlier weights (those with high positive or low negative values) closer to 0 but not quite to 0. L₂ regularization always improves generalization in linear models.²

$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))) \right] + \left[\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2 \right]$$

¹ Add this penalty term to the original Log loss
You can refer to <https://www.youtube.com/watch?v=7T6a296u8A> for more details

Regularized Logistic Regression by Andrew Ng (2:06:00)

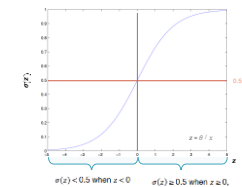
<https://www.youtube.com/watch?v=7T6a296u8A>

- Early stopping, that is, limiting the number of training steps or the learning rate.

:: Logistic Regression

Use the logistic regression model to predict

Once the Logistic Regression model has estimated the **probability** $\hat{y} = h_\theta(x)$ that an instance x belongs to the positive class, it can make its prediction \hat{y} easily.¹



$$\hat{y} = h_\theta(x) = \sigma(x) = \frac{1}{1 + e^{-\theta^T x}} = \frac{1}{1 + e^{-\theta^T x}}$$

Where $x = \theta^T x$

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{y} < 0.5 \\ 1 & \text{if } \hat{y} \geq 0.5 \end{cases}$$

² e.g., classify the email as a normal email if $\hat{y} < 0.5$
³ e.g., classify the email as a spam email if $\hat{y} \geq 0.5$



Video: Logistic Regression - Decision Boundary
<https://www.youtube.com/watch?v=7T6a296u8A>

¹ 2 sources: (1) Andrew Ng, Machine Learning with Decision Trees and Logistic Regression

:: Logistic Regression

How to find the value of θ for fitting the model? -2

We now have a cost function (aka. Loss function) that measures how well a given hypothesis $h_\theta(x)$ fits our training data. We can learn to classify our training data by minimizing $J(\theta)$, to find the best choice of θ .

Minimize the cost function $J(\theta)$ to find the best choice of θ

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

As the cost function is convex, so Gradient Descent can be used to find the global minimum (if the learning rate is not too large and you wait long enough).

Partial derivative of $J(\theta)$

The partial derivative of $J(\theta)$ as given on the left side with respect to θ_j is

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient Descent for logistic regression

As we want to minimize the cost function $J(\theta)$, so

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

} (Simultaneously update θ_0 for every $j=1, 2, \dots, n$)

For more details, –video: Logistic Regression by Andrew Ng. <https://www.youtube.com/watch?v=7T6a296u8A>

:: Logistic Regression

About Multiclass Classification

Whereas binary classifiers distinguish between two classes, multiclass classifiers (also called multinomial classifiers) can distinguish between more than two classes.¹

Binary Classification



In a binary classification problem, a single training or test item (instance) can only be divided into two classes—for example, if you want to train a model to classify an image as a dog, cat, or other animal.²

Multiclass Classification



In a multiclass classification problem, it can be divided into more than two—for example, if you want to train a model to classify an image as a dog, cat, or other animal.³

¹ 2 sources: (1) Andrew Ng, Machine Learning with Decision Trees and Logistic Regression

:: Logistic Regression

Use “one-versus-the-rest ” approach for Multiclass Classification

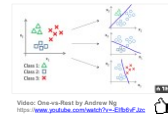
Some algorithms (such as Random Forest classifiers or naive Bayes classifiers) are capable of handling multiple classes directly. Others (such as Support Vector Machine classifiers or Linear classifiers) are strictly binary classifiers. However, there are various strategies that you can use to perform multiclass classification using multiple binary classifiers.

• The most commonly used strategy is One-Vs-The-Rest

Pick a good technique for building binary classifiers (e.g. logistic regression, SVM). Build N different binary classifiers. For the i^{th} classifier, let the positive examples be all the points in class i , and let the negative examples be all the points not in class i . Let f_i be the i^{th} classifier. On a new input x to make a prediction, pick the class i that maximizes $f(x) = \arg \max_i f_i(x)$

For example, one way to create a system that can classify the digits 0-9 into 10 classes (from 0 to 9) is to train 10 binary classifiers, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).

Then when you want to classify an image, you get the decision score from each classifier for that image and you select the class whose classifier outputs the highest score. This is called the one-versus-all (OVA) strategy (also called one-versus-the-rest).



:: Logistic Regression

Recap: Classification with Logistic regression

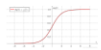
• Train the model

determine best parameters θ_j which can minimize the cost function $J(\theta)$

• Classify new instance using the trained model

• Compute :

$$\hat{\beta} = P(y = 1 | x) = \frac{1}{1 + e^{-(\theta^T x)}}$$



• Classify new instance as:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{\beta} < 0.5 \\ 1 & \text{if } \hat{\beta} \geq 0.5 \end{cases}$$

Key notation

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \text{Sigmoid function}$$

$$\begin{aligned} \theta^T x &= \sum_{i=1}^n \theta_i x_i \\ &= \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n \end{aligned} \quad \text{Weighted sum}$$

$$\sigma(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad \text{Sigmoid function of weighted sum}$$

logistic regression can handle any number of numerical and/or categorical variables

$$P(y = 1 | x) = \sigma(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n)}} = \frac{1}{1 + e^{-\theta^T x}}$$