Uncertainty – probability

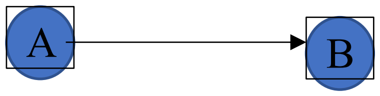
P(A) – marginal probability, formally P(A = a)
P(A, B) - joint probability, formally P(A = a, B = b), P(A) >= P(A, B)
P(A|B) – conditional probability, given B, what is the probability of A,
P(A|B) >= P(A, B), but we don't know P(A), P(B), P(A|B)

P(A, B) = P(A|B) P(B) = P(B|A) P(A) Bayes Rule

Bayesian Network

Why? A way to simplify the storage or computation (training) when we have lots of variables (events) involved through graph models. Assuming each variable has 3 possible values, there are P(A, …, I) has power(3, 9) combinations.

A	B	C	D	E	F	G	H	I
0	0	0						
1	0							
2	0							



The arrow edge indicates the causal relationship

Both of the two terms P(A|B) and P(B|A) are valid!

During training stage, most of the time P(B|A) is more convenient to obtain.

For example, A is a season variables {0, 1, 2, 3}, B is a weather variable {0, 1, 2}

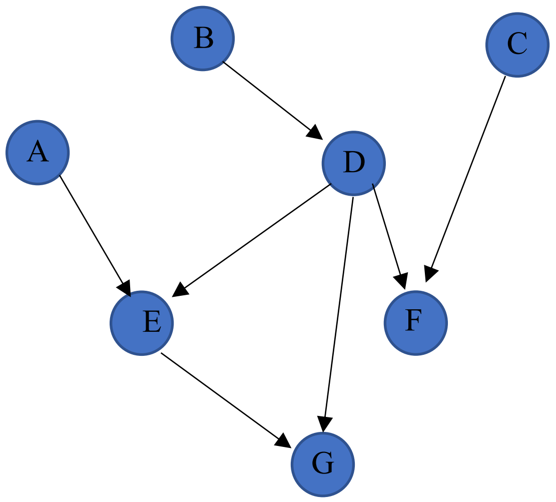
P(B|A) is easier to be trained, e.g. when A = 0, which is Spring season, then you can train how many days of rain, sun shine, and wind. But P(A|B) can be still valid, especially during testing stage. P(B|A)*P(A) = P(A|B) * P(B)

If variables B and E are independent from each other, their joint probability

P(B, E) = P(B) P(E) - factorization (decompose the large joint table into multiple products format)

If variables B and E are not independent: P(B, E) = P(B)P(E|B) = P(E)P(B|E)

$P(X1, X2, \ldots, Xn) = \text{Product}\{ P(Xi \mid \text{parents}(Xi)) \}_{i = 1, \ldots, n}$


$$\begin{tikzcd}
 & B \arrow[dr] & & C \arrow[d] \\
A \arrow[d] & & D \arrow[dl] \arrow[d] \arrow[r] & F \\
E \arrow[r] & & G &
\end{tikzcd}$$

1. P(A, B, C, …, G) what is the joint probability simplified format?
   P(A, …, G) = P(A)*P(B)*P(C)*P(D|B)*P(E|A, D)*P(F|D, C)* P(G|D, E)
2. How many sub-tables do you need to build to store the probabilities?
   7 small tables
3. If each variable has 3 possible values, how many combinations totally from the multiple sub-tables?
   Without Bayesian: 3 ^ 7
   With this Bayesian structure: 3 + 3 + 3 + 3 * 3 + 3 * 3 * 3 + 3 * 3 * 3 + 3 * 3 * 3

4. P(D = d1) = P(D = d1|B = b1)?

   $P(D = d1) = \text{sum}\{P(D = d1 \mid B = bi)\}_{bi} = P(D|B = b1) + P(D|B = b2) + P(D|B = b3)$
   No P(D= d1) >= P(D=d1|B=b1)

5. P(F|D) = P(F|D = d1, C = c2) ?

   P(F|D) = P(F|D, C = c1) + P(F|D, C = c2) + P(F|D, C= c3)
   No P(F|D) >= P(F|D, C)
6. P(G|E, D) = P(G|E, D, A)? Yes
7. P(G|D) = P(G|F, C, E)?
   P(G|F, C, E) = P(G|E)
   P(G|E) = P(G|D) ? It is hard to say (No)

8. P(F|D, C) = P(F|C, D, A, B)? Yes

(Put your answers in a word or .txt file and submit it to the Isidore)

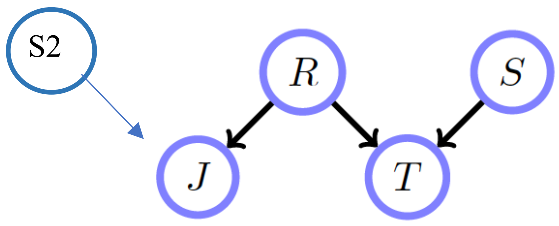
What is the role of Bayesian Network?
It aims to simplify the joint probability, e.g. $P(A, B, C, D, E) = P(A)P(B)P(C|A, B)P(D|C)P(E|C, D)$

What is the motivation of doing this simplification?
It help to make the marginal or joint probability easier.

$$P(A) = \sum_{B,C,D,E} P(A, B, C, D, E) = \sum_{B,C,D,E} P(A)P(B)P(C|A, B)P(D|C)P(E|C, D)$$
$$= P(A)\sum_{B} \{P(B) \sum_{C} P(C|A, B) \sum_{D} P(D|C)\ 1\}$$

One morning Tracey leaves her house and realizes that her grass is wet. Is it due to overnight rain or did she forget to turn off the sprinkler last night? Next she notices that the grass of her neighbor, Jack, is also wet. This explains away to some extent the possibility that her sprinkler was left on, and she concludes therefore that it has probably been raining.

T -	Tracey's grass (0 – dry; 1 – wet)
J -	John's grass (0 – dry; 1 – wet)
S -	Tracey's Sprinkler (0 – off; 1 – on)
R -	Raining (0 – no rain; 1 – rains)
S2 -	John's Sprinkler (0 – off; 1 – on) (Optional)



Case 1:
$P(J, R, S, T) = P(R) P(S) P(J|R) P(T|R, S)$
Case 2:
$P(J, R, S, S2, T) = P(S2) P(R)P(S)P(J|S2, R)P(T|R, S)$

(1) Define the remaining possible variables in this problem following the above example.
(2) Draw a Bayesian network model of these variables
(3) Write down your simplified joint probability according to your Bayesian Network

(4) What is the probability that Tracey's sprinkler was on last night? Write a probability equation.

$P(S = 1|T = 1)$
$= P(S = 1, T = 1) / P(T=1)$

Why do we need to convert the conditional probability in such way?

Answer: This is depends on how much probability information do we know after training. If we can only get the joint probability directly based on some background knowledge, then we have do this conversion based on Bayes rule.

$p(R = 1) = 0.2 \quad p(R=0) = 0.8$
$p(S = 1) = 0.1$
$p(J = 1|R = 1) = 1$, $p(J = 1|R = 0) = 0.2$ (because some other factors can cause John's grass wet)
$p(T = 1|R = 1, S = 0) = 1$
$p(T = 1|R = 1, S = 1) = 1$
$p(T = 1|R = 0, S = 1) = 0.9$
$p(T = 1|R = 0, S = 0) = 0$

$= \sum_{J,R} P(S = 1, T = 1, J, R) / \sum_{J,R,S} P(S, T = 1, J, R)$
$= \sum_{J,R} P(R)\, P(S=1)\, P(J|R)\, P(T=1|R, S=1)$
$/ \sum_{J,R,S} P(R)\, P(S)\, P(J|R)\, P(T=1|R, S)$

$\sum_{J,R} P(R)\, P(S=1)\, P(J|R)\, P(T=1|R, S)$
$= \sum_{R} P(R)\, P(S=1)\, P(T=1|R, S)$ ~~$\sum_{J} P(J|R)$~~
$= P(S=1) \sum_{R} P(R)\, P(T=1|R, S)$
$= 0.1 * \{P(R=0)P(T=1|R=0, S=1) + P(R=1)P(T=1|R=1, S=1)\}$
$= 0.1 * \{0.8 * 0.9 + 0.2 * 1\}$
$= 0.1 * 0.92$
$= 0.092$

$\sum_{J,R,S} P(R)\, P(S)\, P(J|R)\, P(T=1|R, S)$
$=\sum_{R,S} P(R)\, P(S)\, P(T=1|R, S)$
$= P(R=0)P(S=0)P(T=1|R = 0, S=0)$
$+ P(R=0)P(S=1)P(T=1|R=0, S=1)$
$+ P(R=1)P(S=0)P(T=1|R=1, S=0)$
$+ P(R=1)P(S=1)P(T=1|R=1, S=1)$

The answer: $P(S=1|T=1) = 0.3382$

$\sum_{J} P(J|R) = P(J=0|R) + P(J=1|R)$
$=(P(J=0|R=0) + P(J=1|R=0))P(R=0)$
$+(P(J=0|R=1) + P(J=1|R=1))P(R=1)$
$= 1.0 * 0.3 + 1.0 * 0.7$
$= 1.0$

$P(R=1) = 0.3$

P(R=0) = 0.7

Question: based on the above equations and probability table. Calculate what is the result of P(S=1|T=1).

Marginalization:

$\sum_{A,B,C} P(A,B)P(B|C)P(C)P(B)$
$= \sum_{A,B} P(A,B)P(B) \sum_{C} P(B|C)P(C)$
$= \sum_{B,C} P(B|C)P(C)\ P(B) \sum_{A} P(A,B)$

A –> B <- C -> D
C->E

Quiz:
Sally comes home to find that the burglar alarm is sounding (A = 1). Has she been burgled (B = 1), or was the alarm triggered by an earthquake (E = 1)? She turns the car radio on for news of earthquakes and finds that the radio broadcasts an earthquake alert (R = 1).

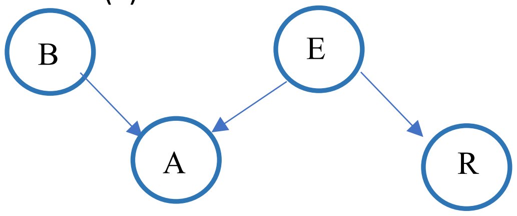
Alarm = 1	Burglar	Earthquake
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

Radio = 1	Earthquake
1	1
0	0

$p(B = 1) = 0.01$ and $p(E = 1) = 0.000001$

P(B=1|A=1, R=1)?

(1) Draw a Bayesian graphic model.
(2) Convert this equation into the form made of joint and marginal probability probabilities.
(3) Simplify the equation further using marginalization.
(4) Calculate the final result.



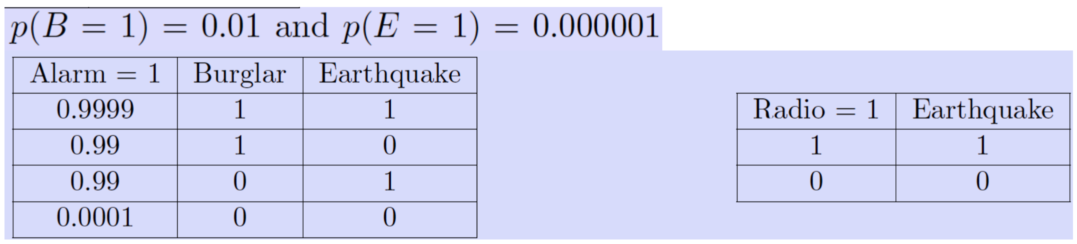
P(A, B, E, R) = P(B)P(A|B, E) P(E)P(R|E)

$P(B=1|A=1, R=1) = P(B=1, A=1, R=1) / P(A=1, R=1)$

$P(B=1, A=1, R=1) = \sum_E P(A, B, E, R)$
$= \sum_E P(B=1)P(A=1|B=1, E)\ P(E)P(R=1|E)$
$= P(B=1) \sum_E P(A=1|B=1, E)\ P(E)P(R=1|E)$
$= 0.01^* \{P(A=1|B=1, E=0)P(E=0)P(R=1|E=0) + P(A=1|B=1, E=1)P(E=1)P(R=1|E=1)\}$
$= 0.01 * \{0.99*0.999999*0 + 0.9999 * 0.000001 * 1\}$

$P(A=1, R=1) = \sum_{E,B} P(A=1, B, E, R=1)$
$= \sum_{E,B} P(B)P(A=1|B, E)\ P(E)P(R=1|E)$
$= (E=0, B=0) + (E=0, B=1) + (E=1, B=0) + (E=1, B=1)$

$(E=0, B=0)$
$= P(B= 0)P(A=1|B=0, E= 0)\ P(E=0)P(R=1|E=0) = 0$



$p(B = 1) = 0.01$ and $p(E = 1) = 0.000001$

Alarm = 1	Burglar	Earthquake
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

Radio = 1	Earthquake
1	1
0	0

Classification -> Neural Network -> Deep Learning

Classification

- A broad/popular type of problem in computer science
- Neural Network/Deep learning is just a method or a strategy to solve classification problem.
- Definition: Given some **background knowledge**, you/computer will assign a label for a particular input data to a certain **group/category.**

e.g. the camera capture many images, you can sort these images by labeling them as "people", "building", "jungle", "animal", "food", …

Background Knowledge: a model that is trained before you do the task.

Different from **Clustering** problem, which separate the input data into multiple groups/categories without any background knowledge.

Classification – training on other (previous) data, supervised learning
Clustering – no training or training only on current data, unsupervised learning, "relative information in current data"

(1) Given a lot of images, let computer find out those images that have faces.
classification
(2) …… , let computer separate the images into brighter ones and darker ones.
Clustering

(3) Computer find a particular pattern from a single image.
classification

(4) Given a single image, group all the pixels into 4 parts according to their color distribution. Initially, get a rough estimation about the group assignment of this image and perform training on these groups.
Still Clustering/unsupervised

Linear Regression

Data (1) existing data
(2) future data

Linear Regression Big picture: use existing data (input and output are known) to predict future data (only knows input). Machine use the existing data to extract some model (knowledge): $\{\theta_0, \theta_1, \ldots, \theta_n\}$, where n is the dimension of input data.

Gradient descent is a technique to help find optimal $\{\theta_0, \theta_1, \ldots, \theta_n\}$ through iteratively refinement.

Input: X (x_1, x_2, … x_n)
Output: y (also called "label")
Goals: We hope we can find a "**linear**" relationship between the input X and output data y.
Assumptions:

(1) The true relationship might not be "linear", but we just use linear to simplify the maths.

(2) Output may not be 1D, however, again to make the problem equation simpler, we choose 1D. Even the output is multi-dimensional, we can line them up in row-wise to convert it into 1D.

Course 1	Course 2	Course 3
[A, B, C, F]	[A, B, C, F]	[A, B, C, F]

Now y?

Original data: AAA ABA ABB AAB
y: [1, 2, 3, 4, …]

(3) Is the output data y known or not known?

Linear regression has two stages. - "**Prediction**"

Price of a house depends on many factors: x_1: area (size), x_2: year, x_3: location, x_4: floors, x_5: bedroom #...
X = [x_1, x_2, x_3, x_4, x_5, …]
y = price

stage I: Training Stage (both X and y are known, consider both as input)
You need to do some homework on existing houses to give you some idea about how much the house be cost depending on these factors. During this stage, you need collect those existing houses as training data. So for training data, the output (label) y is known! So in fact, it's input for training.
What is the output? It's the learned parameters $\{\theta_0, \theta_1, \ldots, \theta_n\}$, each parameter corresponds to the coefficient of [x_1, x_2, x_3, x_4, x_5, …].

Stage II: Testing/Prediction Stage (X is known, y is unknown)
Is y known? Not known. For a new input data X:

$$f_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \longrightarrow \quad y'$$

We hope y' $(= f_\theta(X))$ is as close to y as possible!

$y = f(x)$ if for the function f(), the exponential part of x is at most 1, then we can call this function as linear function.

$y = 5x$
$y = 3x + 7$
$y = 2 \quad (y = 2 + 0*x)$

We can extend dimension of X into multi-dimensional data, e.g. x is 3D: (x1, x2, x3)

$X = (x_1, x_2, x_3)$, we use the lower case x with a number index as the value for the corresponding dimension.
y as the output data, we usually prefer it to be 1D for maths convenience.

Question: write an example of linear function between y and X

$y = 5x_1 + 2x_2 – 7x_3 + 10$
$y = 9 x_2$

y – label (e.g. "face" = 1, "food" = 2, "building" = 3)
$X = (x_1, x_2, x_3)$ - is the input image

The goal of training is to learn the values for the coefficients ($\theta_0, \theta_1, \theta_2, \theta_3$):

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

For the whole classification problem domain, most of all the methods, including neural network, deep learning, …, is just to find an optimal values for the coefficients ($\theta_0, \theta_1, \theta_2, \theta_3, \ldots$) through training, an optimal estimation of these coefficients would produce an accurate prediction on y; otherwise, the result maybe not satisfactory.

Definition of Linear Regression
Below shows all about training stage!

Input Data: X (x$_1$, x$_2$, …) - it can be multidimensional
Output Data: y - by default, the output is 1D for mathematic computation reason. Sometimes the output data is also called "label"
Equation: between y and X is linear combination.
Goal: to learn an optimal model, more specifically, to learn those optimal parameters, $\theta_0, \theta_1, \theta_2, \dots$ How many such parameters $\{\theta_j\}$ equals to the number of X dimension + 1. "j" indicates the dimension index.
Optimal:
Given a set of data pairs $\{<X^{(i)}, y^{(i)}>\} = <X^{(1)}, y^{(1)}>, <X^{(2)}, y^{(2)}>, \dots, <X^{(m)}, y^{(m)}>$, coming in pairs. We will learn the parameters $\{\theta_0, \theta_1, \dots, \theta_n\}$ that minimize the following equation:

One sample <X, y>: $\theta_0 X_0 + \theta_1 X_1 + \dots + \theta_n X_n \xrightarrow{\text{predict}}$ y', we hope that y' is as to y as possible.

$$\boxed{\underset{\{\theta_0, \theta_1, \dots, \theta_n\}}{\text{minimize}}\Big(\frac{1}{m} \sum_{i=1,\dots,m} (\underbrace{(\theta_0 X_0^{(i)} + \theta_1 X_1^{(i)} + \dots + \theta_n X_n}_{\text{Prediction}}\underbrace{^{(i)})}_{\text{Truth}} - y^{(i)})^2 \Big)} \longrightarrow \text{Cost function } J(\theta)$$

The reason of use square instead of "absolute operator" is to allows conveniently compute the derivative, i.e. df()/dx $\quad (5x^2 + 7x + 5)' = 10x + 7$
$\quad (12x + 9)' = 12$

m – the number of training data
n – the number of input data X's dimension
i – indicates the input training data (sample) index, placing as superscript
j – indicates the dimension index, placing as subscript

Gradient Descent

This is a popularly used method to find the optimal parameters $\{\theta_0, \theta_1, \dots, \theta_n\}$ to minimize the equation above. During the training stage, the above equation has the changing variables are $\{\theta_0, \theta_1, \dots, \theta_n\}$ instead of X.

$$J(\theta) = \frac{1}{2m} \sum_{i=1,\dots,m} ((\theta_0 X_0^{(i)} + \theta_1 X_1^{(i)} + \dots + \theta_n X_n^{(i)}) - y^{(i)})^2$$

The reason we let J(θ) defined as 1/2m for maths reason that make the result of first derivative easier.

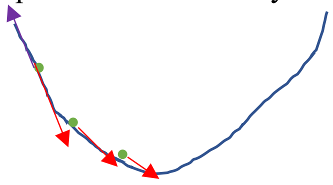
Idea: give a random guess of the possible values for $\{\theta_0, \theta_1, \dots, \theta_n\}$, then plug in these values into J(θ) and get the error from the equation. Then try to modify these guessed values $\{\theta_0, \theta_1, \dots, \theta_n\}$ by adjusting each individuals, denoted as $\{\Delta\theta_0, \Delta\theta_1, \dots, \Delta\theta_n\}$, s. t. they are updated as:

$$\theta_0 = \theta_0 + \Delta\theta_0$$
$$\theta_1 = \theta_1 + \Delta\theta_1 \dots$$
$$\theta_n = \theta_n + \Delta\theta_n$$

Hopefully, this update will allow the overall error decrease. Then we can repeat this process until the error gets small enough.

The Key is to compute $\{\Delta\theta_i\}$

Simple case: If there is only one theta to predict $\{\theta_0\}$, the error equation is just a quadratic function. The equation is similar to $y = ax^2 + bx + c$, where y is just the $J(\theta)$, and x is θ_0.

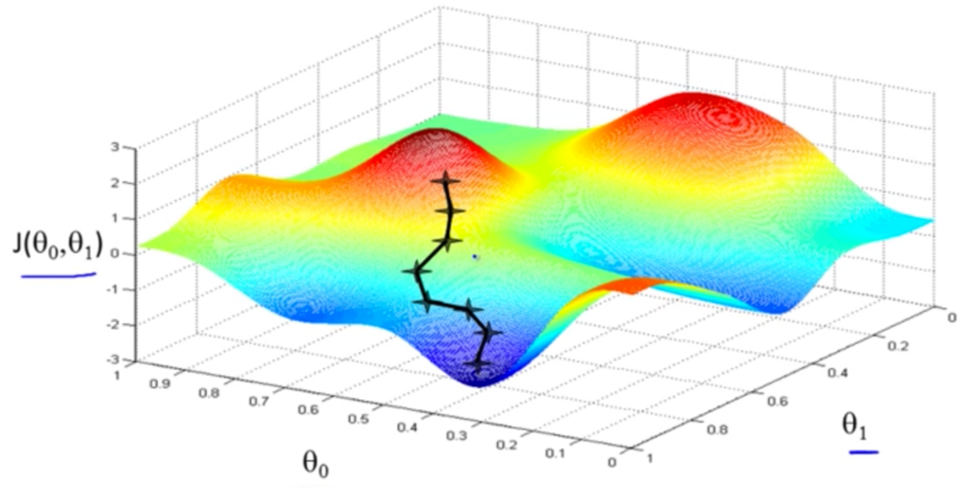


Repeat {

$$\theta_0 = \theta_0 - \frac{dJ(\theta)}{d\theta}$$

}

More complex case: there are two parameters to predict $\{\theta_0, \theta_1\}$, which means the input data X is 1D. The above parabola curve is extended to a curly surface.



Repeat {

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

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

}

α - speed controller

Attention: You need to make sure for each iteration, the two parameters θ_0 and θ_1 are updated by using the partial derivative on the cost function $J(\theta_0, \theta_1)$ based on the (θ_0, θ_1) values in last iteration, i.e. at iteration i, update θ_0 using the value of θ_0 and θ_1 from iteration (i-1); similarly update the θ_1 using the value of θ_0 and θ_1 from iteration (i-1). Don't used the currented updated θ_0

For a problem, we have the initial guess $\theta_0 = 5$, $\theta_1 = 12$. We also know the partial derivatives:

$\partial J(\theta_0, \theta_1) / \partial \theta_0 = 0.3\,\theta_0 + 0.6\,\theta_1$

$\partial J(\theta_0, \theta_1) / \partial \theta_1 = -0.2\,\theta_0 + 0.5\,\theta_1$

What are the (θ_0, θ_1) after 2 iterations? Assuming $\alpha = 1$

Iteration 1:

Theta_0 = theta_0 – partial_derivative(theta_0) = 5 – (0.3 * 5 + 12 * 0.6) = -3.7

Theta_1 = theta_1 – partial_derivative(theta_1) = 12 – (-0.2 * 5 + 12 * 0.5) = 7

Iteration 2:

Theta_0 = theta_0 – partial_derivative(theta_0) = -3.7 – (0.3 * (-3.7) + 0.6 * 7) = -4.248

Theta_1 = theta_1 – partial_derivative(theta_1) = 7 – ((-0.2) * (-3.7) + 0.5 * 7) = 2.76

…

Consider a problem, let's try to predict the price of a car. The price can depends on many factors, such as brand name, model, years, …. We just make a very simple case, let's just consider the car speed as the main factor for the price perdition. We have training data:

input		output
	(100m/h,	$30,000)
	(120m/h,	$45,000)
	(160m/h,	$60,000)

Assume, we only have three data for training. Now, (1) could you write down the cost equation for this prediction? (2) How many parameters should I predict? (3) If I initially predict **θ** as {2, 5}, then what are the new **θ** in the next iteration?

X – 1D, which is just the speed

y – 1D, which is the price

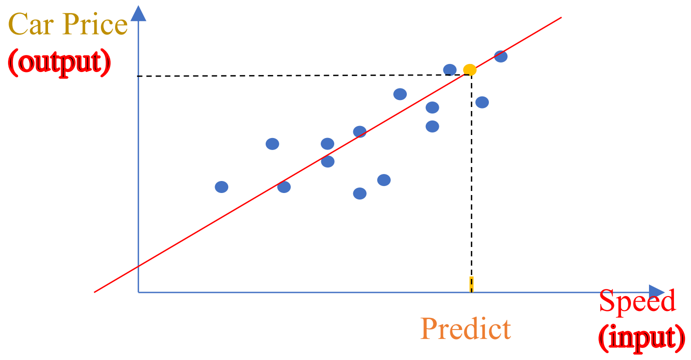
θ – 2D, which are $\{\theta_0, \theta_1\}$

$$J(\theta_0, \theta_1) = (1/6)\left[(\theta_0 + 100\theta_1 - 30000)^2 + (\theta_0 + 120\theta_1 - 45000)^2 + (\theta_0 + 160\theta_1 - 60000)^2\right]$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = (1/6)\,[\,2 * \theta_0 + 2 * \theta_0 + 2 * \theta_0] = \theta_0 = 2$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = (1/6)[2 * 100 * \theta_1 + 2 * 120 * \theta_1 + 2 * 160 * \theta_1]$$

The visualization of Linear Regression – fit a hyper line on the distributed points.



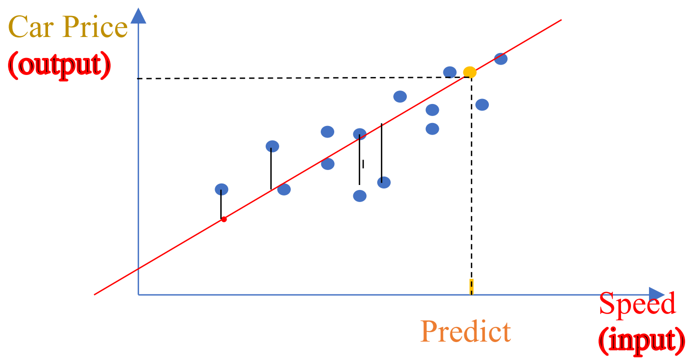
Goal of linear regression:

Prediction

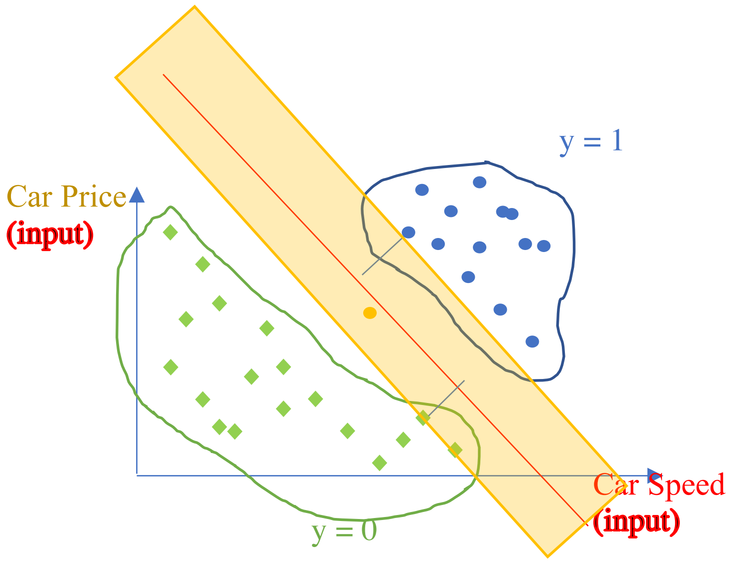
Visually the training procedure can be also considered as “fitting”.

Classification

- Separating data into different groups
- Given an input data, assign a group for it.
- Similar to the “linear regression”,
  - It also has training stage and testing/group assignment stage
  - During the training stage, it has both input (X) and output (y) data, also the output data y is usually converted into 1D
  - During the testing/group assignment stage, the output data y is unknown. y is just a group label/assignment.
  - To solve the problem, it also aims to find a set of parameters $\{\theta_0, \theta_1, \ldots, \theta_n\}$, where n is the dimension of input data X. The relationship between $\{\theta_0, \theta_1, \ldots, \theta_n\}$ and X(x_1, x_2, …, x_n) is also in linear combination.
- Different from the “linear regression” ,
  - Linear regression is for prediction, visually it’s training stage is to perform line or hyper-line fitting.
  - Classification is for group assignment, visually it’s training stage is to perform grouping or data separation.



Linear Regression



Classification Example above shows that we want to tell if a car is "good" or "bad" according to two factors (input): the car's price, and the car's speed.

How can we generate such a different line for the classification?
The only difference is depending on the parameters **θ** = $\{\theta_0, \theta_1\}$, which means during classification, we find different optimal parameters through training. Both problems will use gradient descent (iterative partial derivative refinement) to find the optimal parameters $\{\theta_0, \theta_1, \ldots, \theta_n\}$. So the only possible difference between these two problems is on their **cost function**!

Consider an individual (X, y) for their cost function (error):
linear regression: $(\boldsymbol{\theta}^T\mathbf{X} - y)^2$ (attention, X is converted as $[1, \mathbf{X}^T]^T$)
Classification regression: $h(\boldsymbol{\theta}^T\mathbf{X})$ vs {0, 1}

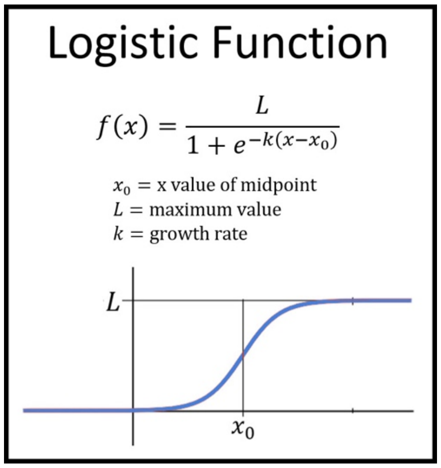
$(h(\boldsymbol{\theta}^T\mathbf{X}) - y)^2$

Label

For classification, we use a mapping function h() to map the output $\boldsymbol{\theta}^T\mathbf{X}$ into a value between 0 and 1, noted [0, 1], in such a way, it can be used as a grouping function to label the input data with either 0 or 1, e.g. if h() produces a value less than 0.5, it can be considered as group 0; otherwise it is assigned as group 1.

Which function h() we can choose to do this mapping job?

- Sigmoid function or Logistic function


Logistic Function
$f(x) = \frac{L}{1+e^{-k(x-x_0)}}$
x_0 = x value of midpoint
L = maximum value
k = growth rate
L
x_0

h() = 1 / (1 + $e^{-(\theta' X)}$) produces an output between 0 and 1. Sometimes this function called activation function, which converts a linear relationship into a non-linear relationship.

How can we measure the error?

For example, for an input sample X(50, 100), and we know its label is 1. Then if we use activation function h() on the estimated parameters $\boldsymbol{\theta}^T$, we get the value $h(\boldsymbol{\theta}^T\mathbf{X}) = 0.72$, we think this estimation is correct; if we get $h(\boldsymbol{\theta}^T\mathbf{X}) = 0.51$, we still think the estimation is correct, but the error is relatively high; if we get $h(\boldsymbol{\theta}^T\mathbf{X}) = 0.39$, we think the estimation is wrong, meanwhile the error is larger.

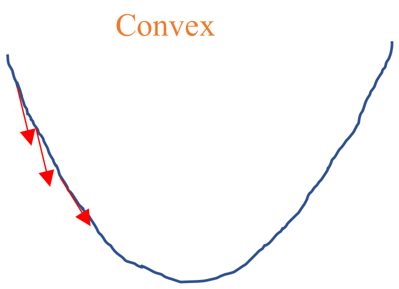
Individual error: $(h(\boldsymbol{\theta}^T\mathbf{X}) - y)^2$ (not suggested)

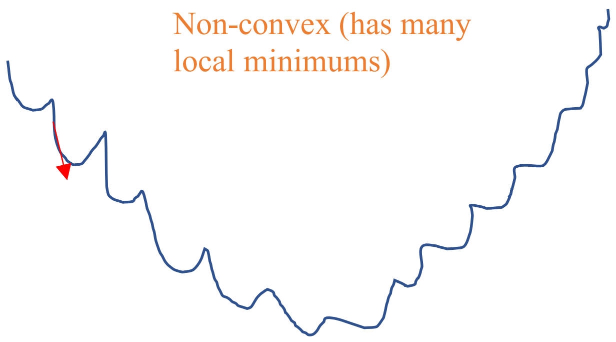
We may put this error into the same cost function $J(\boldsymbol{\theta})$:

$$J(\theta) = \frac{1}{2m} \sum_{i=1,\dots,m} \boxed{(h(\boldsymbol{\theta}^T\mathbf{X}) - y^{(i)})^2}$$

- **Replace this error by the log()**

But the problems of using this error measurement involves (1) It is not straigt forward to compute the partial derivatives (2) Using the logistic function with squared sum operation, a non-convex function will be generated.

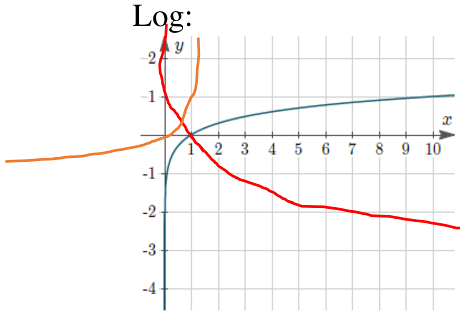


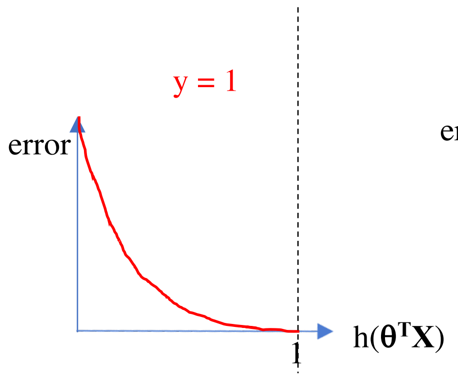


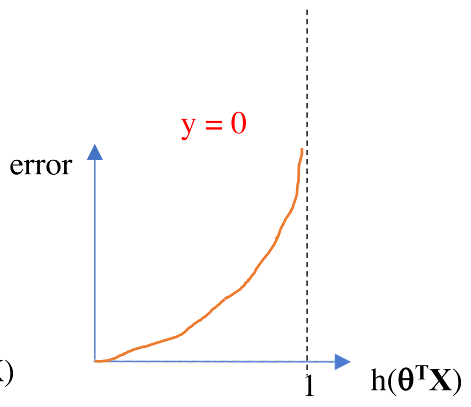
To avoid generating local minimums, we will apply log() on top of the activation function (h()). The beauty is the combination between log() and logistic functions will generate a convex function.

$$\text{Error}(h(\boldsymbol{\theta}^T\mathbf{X}), y) \begin{cases} \text{If } y = 1, & -\log(h(\boldsymbol{\theta}^T\mathbf{X})) \\ \text{If } y = 0, & -\log(1 - h(\boldsymbol{\theta}^T\mathbf{X})) \end{cases}$$

Log:







Individual Error$(h(\boldsymbol{\theta}^T\mathbf{X}), y) = (1-y)(-\log(1 - h(\boldsymbol{\theta}^T\mathbf{X}))) + (-y)(-\log(h(\boldsymbol{\theta}^T\mathbf{X})))$ is used to replace the sum of squared measurement:

$$J(\theta) = \underline{\frac{1}{m}\sum_{i=1}^{m}\text{Cost}(h_\theta(x^{(i)}), y^{(i)})}$$

$$= \underset{\uparrow}{-}\frac{1}{m}\underline{[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1 - h_\theta(x^{(i)}))]}$$

How can we find the best parameters $\boldsymbol{\theta}$ that can minimize $J(\boldsymbol{\theta})$?

Gradient Descent

Different Cost Function

Repeat {

$$\theta_j \begin{cases} \theta_0 = \theta_0 - \dfrac{\partial\, \boxed{J(\theta_0, \theta_1, \ldots, \theta_n)}}{\partial\, \theta_0}\alpha \\ \theta_1 = \theta_1 - \dfrac{\partial\, J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial\, \theta_1}\alpha \\ \ldots \\ \theta_n = \theta_n - \dfrac{\partial\, J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial\, \theta_n}\alpha \end{cases}$$

}

$$\frac{\partial\, J(\boldsymbol{\theta})}{\partial\, \theta_j} = 1/m * \sum_{i=1,\ldots,m}(h(\theta^T X^{(i)}) - y^{(i)})X_j^{(i)})$$

h() is the logistic function

$h(x) = 1/(1 + e^{-x})$

θ^T is 1 x (n+1)

X is (n+1) x 1

$\theta^T X$ = 1x1

Assume we have 5D data as input X:
X1 = [1, 5, 2, 1, 4, 3],   y1 = 0
X2 = [1, 0, 5, 2, 3, 4],   y2 = 1
X3 = [1, 5, 2, 1, 3, 7],   y3 = 0
…

What is the dimension of θ^T ?

Assuming my initial guess of θ^T is [0.2, 0.5, 1.2, 0, -5, 3.2]
If I apply this θ^T to the samples X1, X2, X3 what is the partial derivative for the cost function based on these samples?

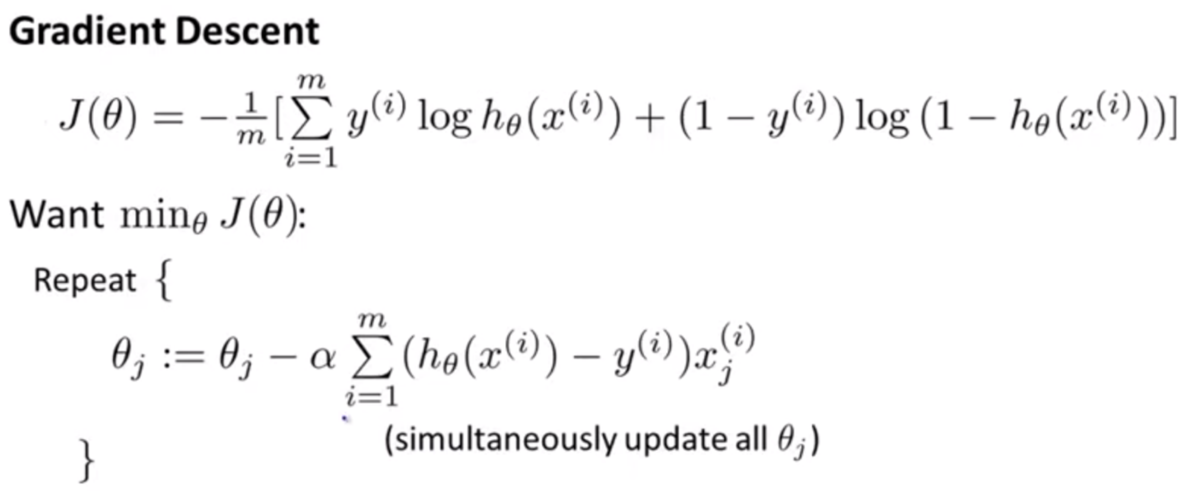
$1/m * \sum_{i=1,\dots,m}(h(\theta^T X^{(i)}) - y^{(i)})X_j^{(i)})$

Assuming we are updating θ_3 ?
X1: $\theta^T X^{(i)}$ = [0.2, 0.5, 1.2, 0, -5, 3.2] * $[1, 5, 2, 1, 4, 3]^T$
= 0.2 * 1 + 0.5 * 5 + 1.2 * 2 + 0 * 1 + (-5) * 4 + 3.2 * 3
= -5.3
$h(-5.3) = 1/(1 + e^{-5.3}) = 0.99$
$(h(-5.3) - y^{(1)})X_3^{(1)} = (0.99 - 0) * 1 = 0.99$

X2: -0.5 (randomly guess)
X3: 0.2   (randomly guess)

(0.99 + (-0.5) + 0.2)/3 = 0.69/3 = 0.23   this is the value for updating θ_3
$\theta_3 = \theta_3$ – (alpha) * partial derivative = 0 – 1 * 0.23 = -0.23

$X_2^{(3)}$ is used for updating θ_2



Gradient Descent

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

Want $\min_\theta J(\theta)$:

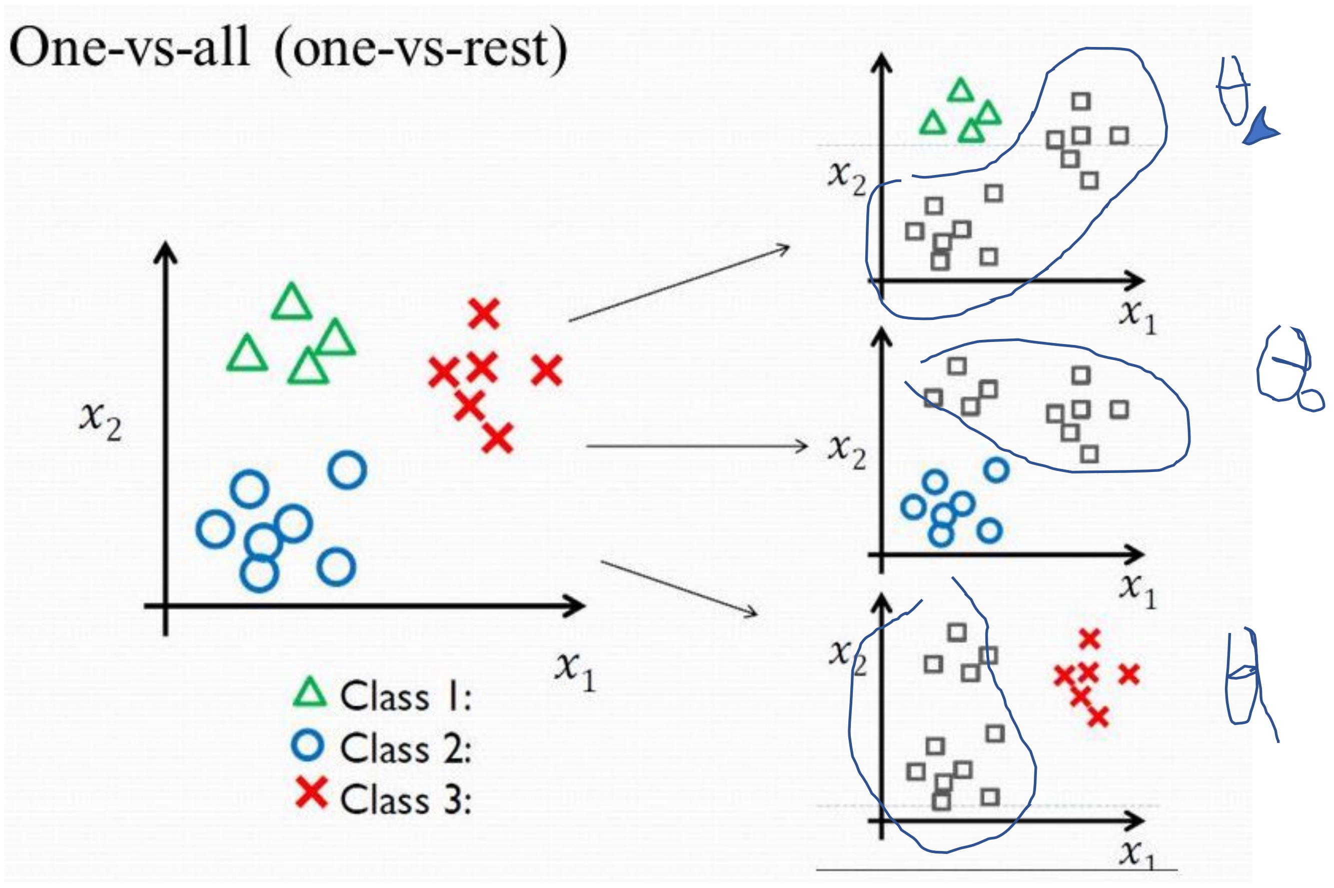
Repeat {

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

(simultaneously update all θ_j)

}

Many classes? One-vs-All



Final Project

Use gradient descent with logistic function to perform a simple classification problem. You will be provided with some training data to perform a prediction. The data is provided in the format of .txt file. You need to write a program to load this data and use gradient descent to iteratively find an optimal parameters. After you find the parameters, you can apply these parameters to a set of testing data (provided in .txt file). You write down a report of how accurate of your estimation.

You need to use C/C++ to write the code. Don't use Python. Don't use any gradient descent package. You need to implement this idea by yourself using a loop and put the partial derivative formula, which we learned in the class, into the loop. After serval iterations, you will get the optimal parameters. For the logistic function, you need to type the equation yourself; functions such as log(), e^x can use "maths" package of C library.

Turn in your C/C++ code.
Turn in your report on the accuracy.