Machine Learning Class Note

Reference : ① Prof. Andrew Ng's learning class @ Coursera ~~Stanford University~~
② Wikipedia ③ CS231 class @ Stanford University

Topic 1 : Introduction

* What is Machine Learning?

[Wikipedia] Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead.

* can be divided into :

$$\text{ML}\begin{cases}\text{supervised learning}\\\text{unsupervised learning}\\\text{weakly supervised learning}\\\ldots\end{cases}$$

supervised learning : a machine learning task of learning a function that maps an input to an output based on example input-output pairs. ("right answer" given)

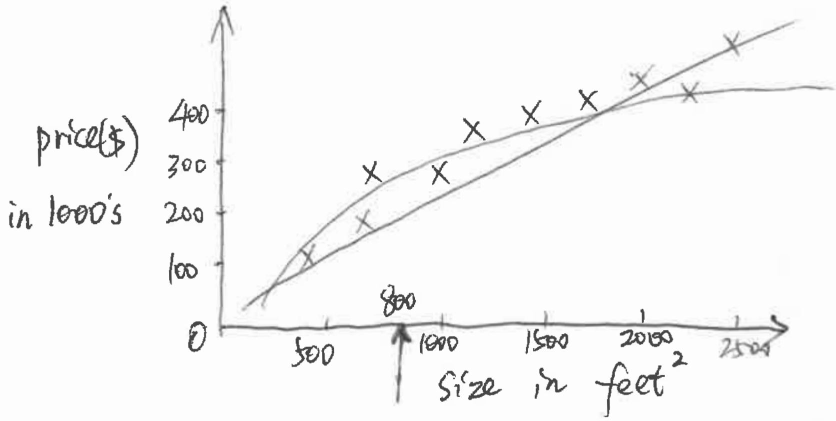
unsupervised learning : a machine learning task without a teacher ("right answer" not given)

Given a data, find the structure of data without "right answer"

weakly supervised learning : similar to supervised learning, but with weak supervision, such as lower-quality labels. ("right answer" weakly given)

Example of Supervised Learning:

① Housing price prediction



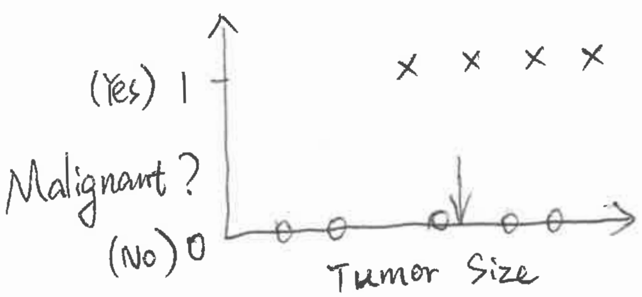
Plot Data in a city.

What is the estimate price for a house with 800 feet^2 in that city?

Supervised learning: "right answer" given

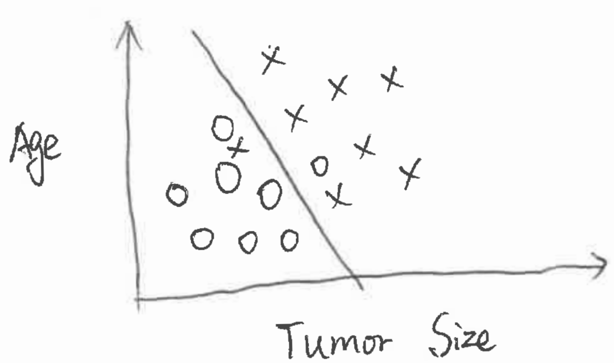
Regression: Predict continuous valued output (price)

② Breast Cancer Classification (malignant, benign)



One feature/attribute

multiple features/attributes



supervised learning: "right answer" given

Classification: Discrete valued output (0 or 1)

↓

Or: 0, 1, 2, 3, ...

benign, cancer type 1, cancer type 2, cancer type 3

Supervised learning:

Data is divided into:

10k	1k		5k
training set	validation set		testing set

⇓

m×n matrix { m: sample #
n: feature dimention/attribute.

example data:

	Age	Tumor Size	Label/gt
Patient 1	20	5	1
Patient 2	25	2.5	0
⋮			⋮
Patient M	45	5	1

training set / validation set / testing set : non-overlapped usually

training set: given "right answer" → train a model

validation set: validata the trained model usually for parameter setting and model selection

testing set → test the trained model; new data: no "right answer".

Take K-NN (k-nearest neighbor) based Image Classification as example:

① feature extraction

400 × 400 I_i → feature extraction (CNN/HOG/color histogram) → 4096D feature vector. f_i

② training set / validation set

10,000 images   1,000 images : each image can be represented with one feature vector

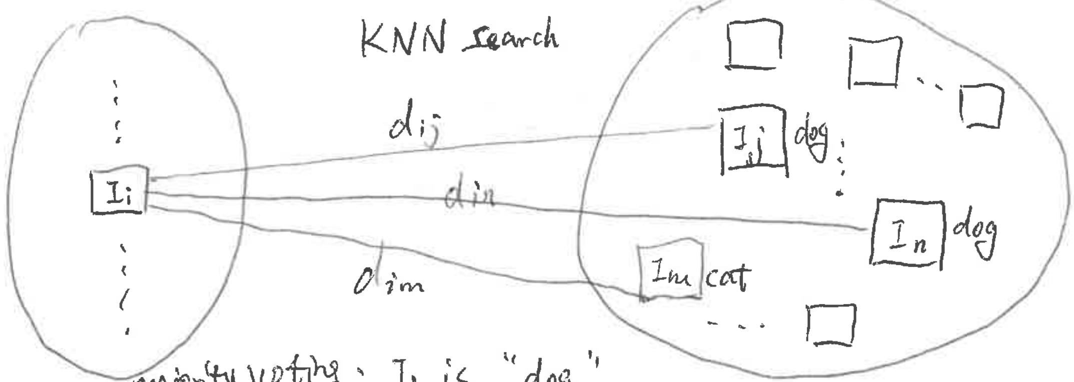
I_i f_i

each image has a classification label by human

I_i dog, cat, monkey, ...
1000 classes

(3.1) naive training (Method1)

Validation set : 1000 images | training set : 10,000 images



majority voting : I_i is "dog".

* distance of Image i and Image j can be the l_2 distance of f_i and f_j feature vectors.
* Given the label of each image on Validation Set, we can know the majority voting result is correct or not ⇒ an accuracy can be obtained
* Using validation Set, Search on Training Set. Suppose that:

$K=1 \Rightarrow$ accuracy $= 50\%$

$K=2 \Rightarrow$ accuracy $= 55\%$

$K=3 \Rightarrow$ accuracy $= 58\%$

$K=4 \Rightarrow$ accuracy $= 60\%$

$K=5 \Rightarrow$ accuracy $= 62\%$ ☆ best validation accuracy obtained when $K=5$

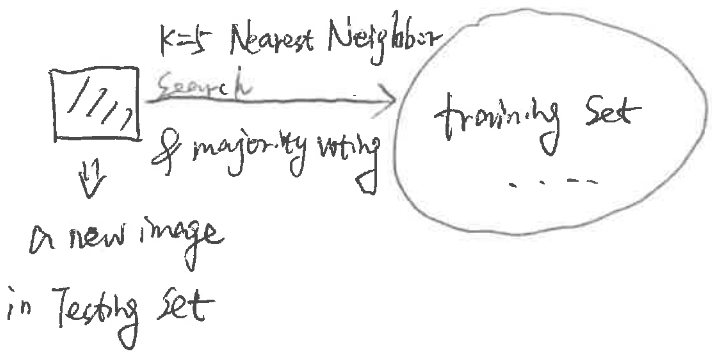
⇓

$K=5$ after training

$K=6 \Rightarrow$ accuracy $= 55\%$

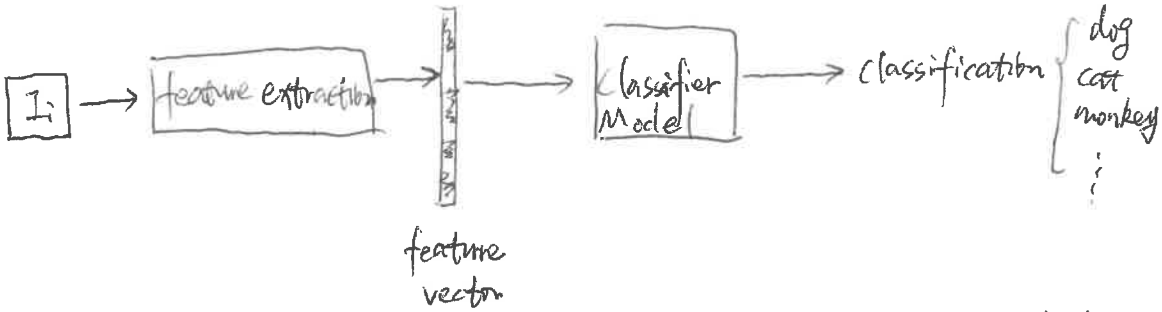
⋮

(4.1) Use the trained model on Testing Set (new 5,000 images).



(3.2) train a classifier, like SVM, NN, CNN, etc. (Method 2)

More Accurate



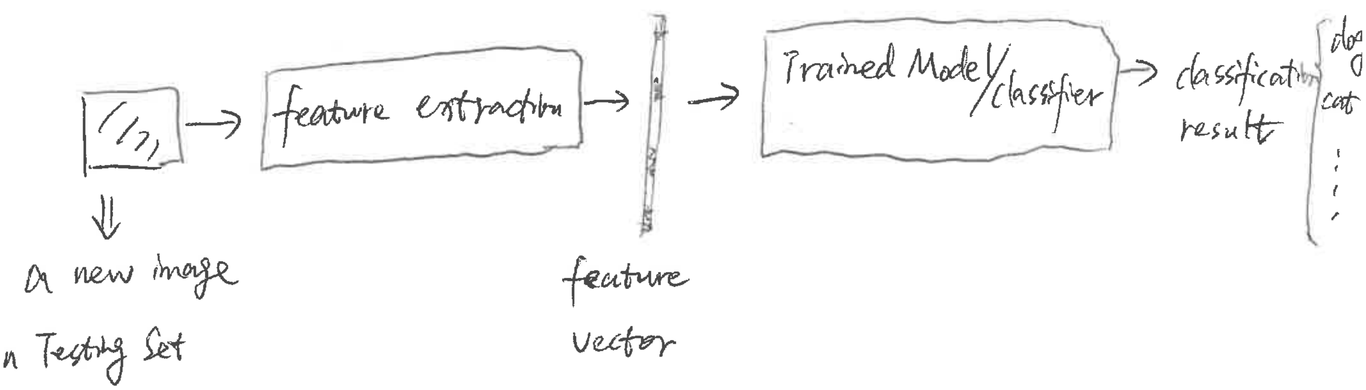
Training: adjust the classifier to make the prediction (output) by the classifier can be more close to the manully labeled result/gt.

On Training Set 10,000 images ⇐ Repeat it for many iterations (for example: 20,000 times).

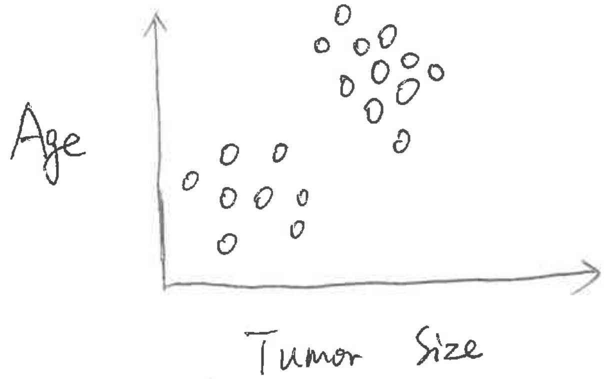
Validation: tune some hyperparameters to select the best classifier that has best performance on validation set.

On Validation Set 1000 images ⇐ hyperparameters: like learning rate, model structure, iteration number, etc

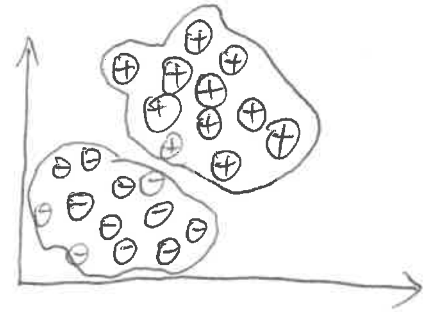
(4.2) use the trained model (classifier) on Testing Set (new 1000 images)



Example of unsupervised learning.



Find the structure of data without "right answer"/gt

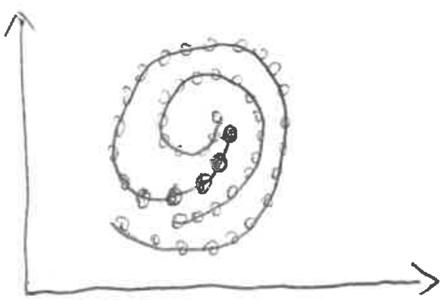


① clustering (Kmeans, ...)

⇓ as above: cluster the data into k groups based on similarity

like Google News: Sports, Technology, Business, ...

for example: l_2 distance

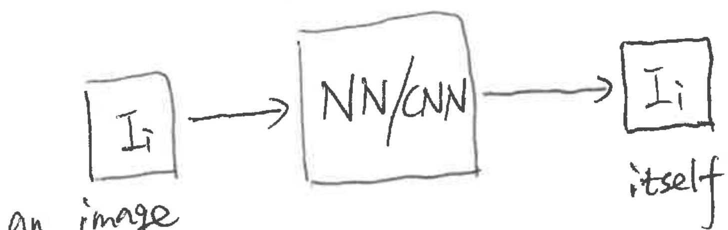


Kmeans cannot solve this clustering problem.

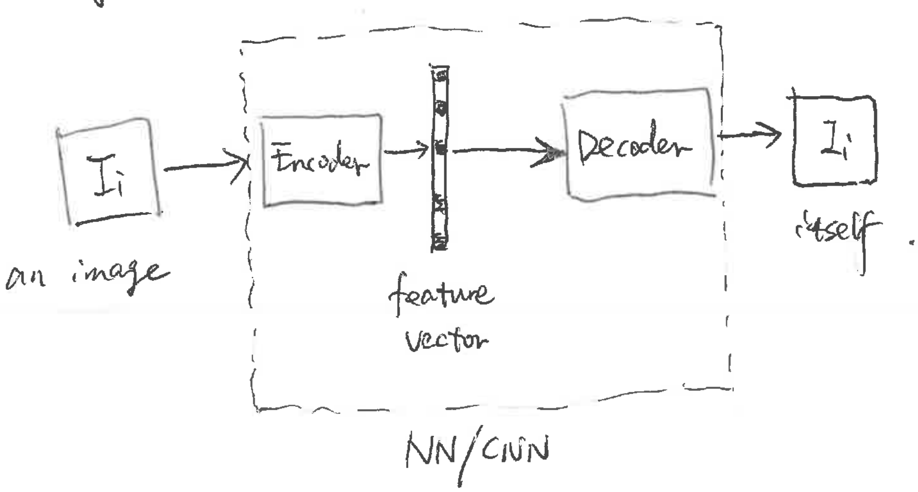
Need more advanced algorithm

② Feature Extraction by Encoder & Decoder.

Feature: color histogram, histogram
HOG, RGB color, ...
learned feature, ---



Train a NN/CNN on Training Set (10,000 images) for many iterations (say, 20,000 times)

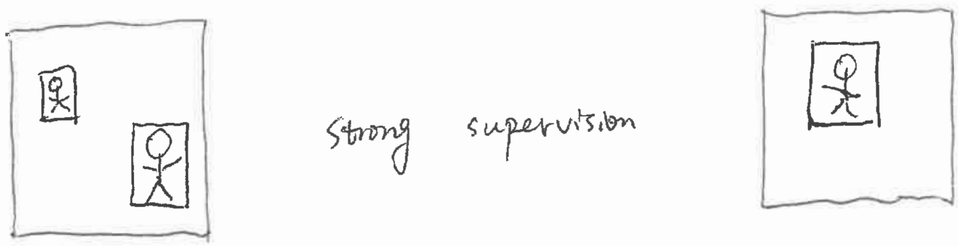


Example of Weakly Supervised Learning.

Similar as supervised learning, but with lower-quality labels.
("right answer" weakly given)

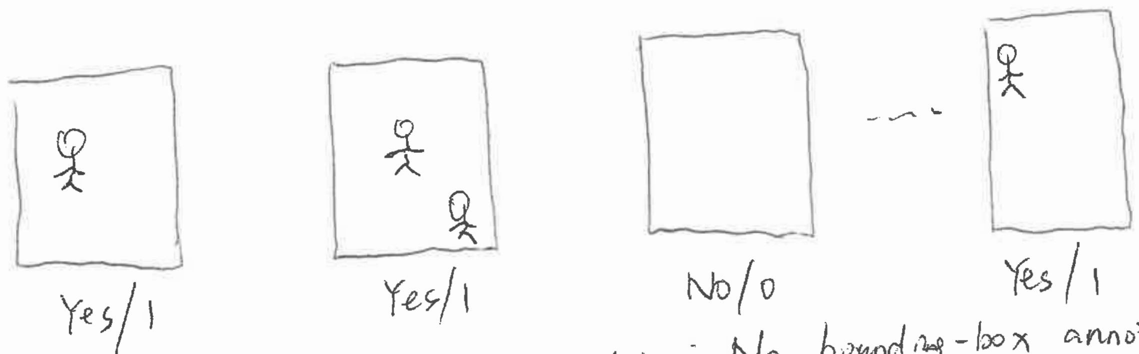
Supervised learning for Human detection:

On Training Set (10,000) images, for each image:



Weakly supervised learning for Human detection:

On Training Set (10,000) images, for each image:



Weak supervision: No bounding-box annotation
but Yes/No annotation.

* Overfitting.

On supervised learning, if the trained model performs very great on training and validation set, but performs poor on testing set. It is called 'overfitting'

training accuracy: 99%

validation accuracy: 98%

testing accuracy: 40%

8

Machine Learning Class Note

03/2019

Reference : ① Prof. Andrew Ng's Machine Learning Class @ Coursera Stanford

② Wikipedia

③ CS 231 class @ Stanford

Topic 2: Linear Regression with one variable

Training set of housing prices

size in feet^2 (x)	Price ($) in 1000's ($y$)
2104	460
1416	232
1534	315
852	178
...	...

$\Rightarrow$ $m = 47$

Notation :

m: Number of training examples

x's: "input" variable / features

y's: "output" variable / "target" variable

$x^{(1)} = 2104$

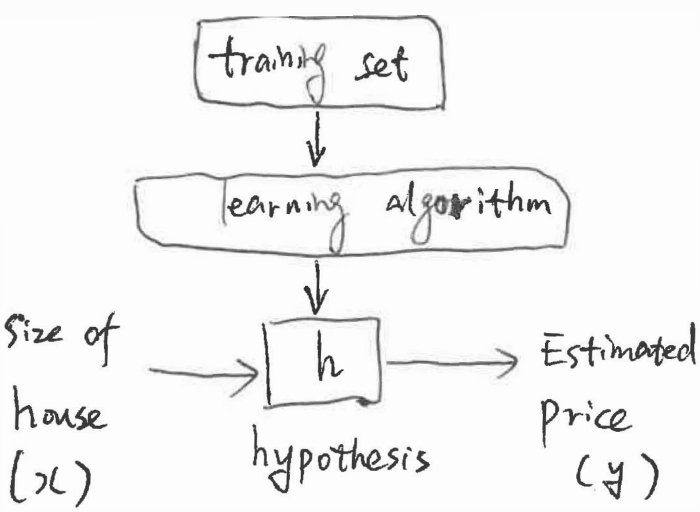
$x^{(2)} = 1416$

$y^{(1)} = 460$

$y^{(2)} = 232$

(x, y) : one training example

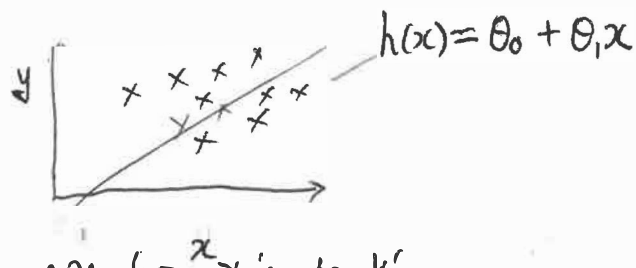
$(x^{(i)}, y^{(i)})$: i-th training example



How to represent h ?

Say $h_\theta(x) = \theta_0 + \theta_1 x$

shorthand : $h(x)$



h maps from x's to y's.

Linear regression with one variable

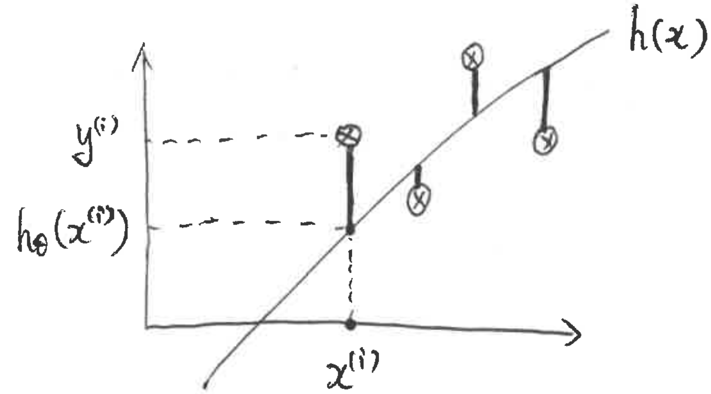
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

parameters: θ_0, θ_1

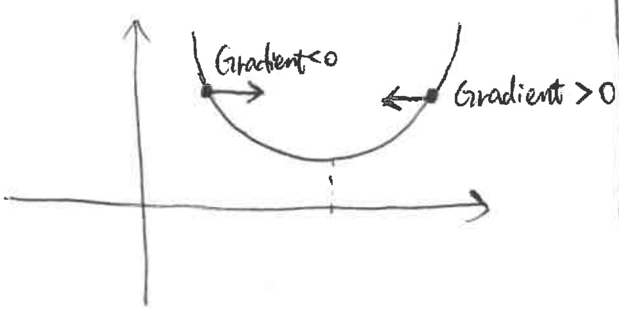
cost function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

misalignment

Goal: $\underset{\theta_0, \theta_1}{\text{Minimize}}\ J(\theta_0, \theta_1) \Rightarrow$ Find a linear line (θ_0, θ_1) to well fit the training samples so as to minimize the cost function / misalignment.

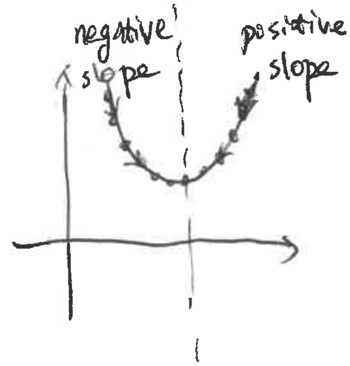


Gradient descent:



Optimization: minimize the cost function

If the cost function is convex, moving to $-1 *$ Gradient direction will find smaller cost function, Repeat this until finding the global minimum.



Gradient descent Algorithm

Repeat until convergence {

$$\theta_j = \theta_j - \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}_{\text{derivative}} \quad \text{for } j=0 \text{ and } j=1$$

}

Correct:

$temp0 = \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$temp1 = \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 = temp0$

$\theta_1 = temp1$

Incorrect:

$temp0 = \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 = temp0$

$temp1 = \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_1 = temp1$

How to compute the gradient?

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

when $j=0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

when $j=1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$

$\frac{\partial}{\partial x} x^2 = 2x$ $\quad \frac{\partial}{\partial x} (x+3)^2 = 2(x+3) \times 1$

$$\frac{\partial}{\partial x} (kx+b)^2 = 2(kx+b) \cdot \frac{\partial}{\partial x} (kx+b) = 2(kx+b) \cdot k$$

$$\frac{\partial}{\partial b} (kx+b)^2 = 2(kx+b) \times 1$$

partial gradient

Topic 3: Linear Regression with multiple variables.

size (feet^2) x_1	Number of bedrooms x_2	number of floors x_3	Age (years) x_4	Price (in \$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
⋮	⋮	⋮	⋮	⋮

Notation: n: number of features

$x^{(i)}$: input (features) of i-th training example.

$x_j^{(i)}$: value of feature j in i-th training example.

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_3 + \theta_4 \cdot x_4$$

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \theta_n \cdot x_n$$

Let us define $x_0 = 1 \Rightarrow x_0^{(i)} = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \Rightarrow h_\theta(x) = \theta^T \cdot x$$

column vector   column vector

$$\Updownarrow$$

$$[\theta_0, \theta_1, \cdots, \theta_n] \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Hypothesis:

$$h_\theta(x) = \theta^T \cdot x = \theta_0 \cdot \underbrace{x_0}_{x_0=1} + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

parameters: θ: $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ $(n+1)$-dimentional vector

cost function:

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

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

New Algorithm

Repeat {

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

% simultaneously update θ_j for $j = 0, \cdots, n$

}

Feature scaling / normalization.

$0 \le x_1 \le 3$ ✓

$-2 \le x_2 \le 0.5$ ✓

$-100 \le x_3 \le 100$ ×

Mean normalization:

$$x_j = \frac{x_j - \mu_j}{s_j}$$

x_j: j-th feature

μ: mean/average value of x_j

s: stand deviation or range (max − min)

Topic 4: clustering / kmeans

Given a set of observations $(x_1, x_2, \cdots, x_n)$, where each observation is a d-dimentional vector.

x_1: n-dimentional feature vector

x_2: n-dimentional

$\vdots$

x_n: n-dimentional

kmeans clustering aims to partition the n observations into k ($k \le n$) sets $S = \{S_1, S_2, \cdots S_k\}$ to minimize the within-cluster sum of squares:

$$\underset{S}{\arg\min} \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

Where μ_i is the mean point/vector in S_i.

Standard Algorithm:

① Initilize K means/clusters randomly

Repeat until convergence:

① Assignment step: Assign each observation to the "nearest" mean/cluster

② Update step: Calculate the new means of the re-assigned observations in the new clusters

* This algorithm has converged when the assignments no longer change.

* This algorithm does not guarantee to find the optimum.

How to define K? Some heuristic algorithms:

$$\text{Loss}(K) = \frac{d_{intra}}{d_{inter}}$$

where d_{intra} is the average distance between each observation to the center within each cluster

d_{inter} is the average distance of each inter-cluster center.

$\underset{(K)}{\text{argmin}}\ \text{Loss}(K)$: eg., Find the minimum loss when $K = 2, \cdots, 10$.

eg. = for example.