

# Lecture 2: Preliminaries of Machine Learning

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering  
Nanjing University

Oct. 11th, 2021

# Table of Contents

## 1 Machine Learning

- Supervised Learning, Linear Regression
- Unsupervised Learning, K-means Clustering

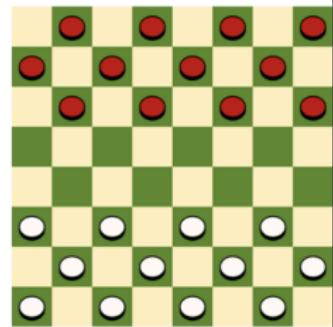
# Definition of Machine Learning

Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.



A. L. Samuel\*

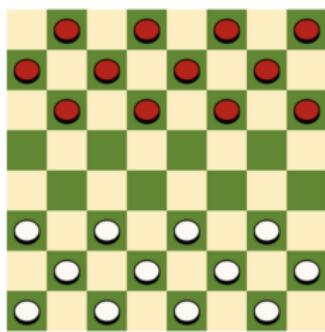
**Some Studies in Machine Learning  
Using the Game of Checkers. II—Recent Progress**



# Definition of Machine Learning

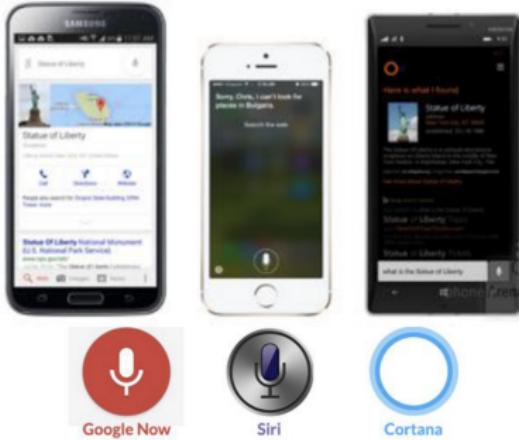
Tom Mitchell (1998): a computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .

- Experience (data): games played by the program (with itself)
- Performance measure: winning rate



# Example: Speech Recognition

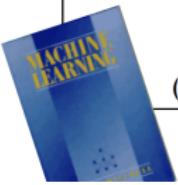
## 1. Learning to recognize spoken words

THEN	NOW
<p>“...the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal...neural network methods...hidden Markov models...”</p> <p>(Mitchell, 1997)</p> 	

Source: <https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults>

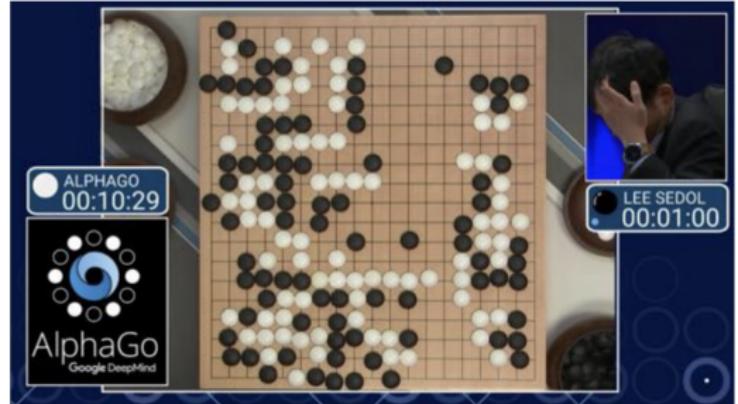
# Example: Robotics

## 2. Learning to drive an autonomous vehicle

THEN	NOW
<p>“...the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars...”</p> <p>(Mitchell, 1997)</p> 	 <p><a href="https://www.geek.com/wp-content/uploads/2016/03/uber.jpg">https://www.geek.com/wp-content/uploads/2016/03/uber.jpg</a></p>

# Example: Games / Reasoning

## 3. Learning to beat the masters at board games

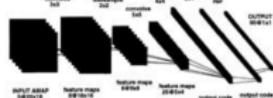
THEN	NOW
<p>“...the world’s top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself...”</p> <p>(Mitchell, 1997)</p> 	

# Example: Computer Vision

## 4. Learning to recognize images

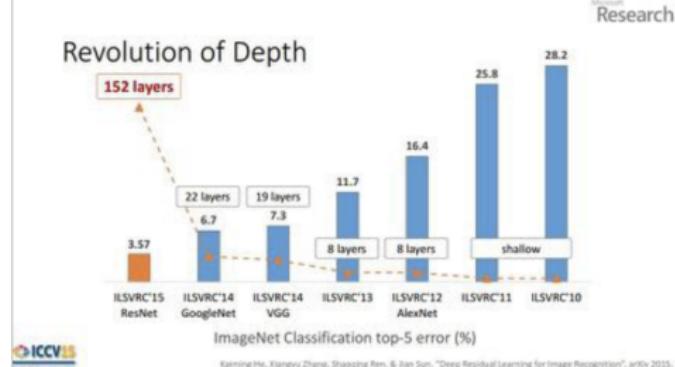
### THEN

“...The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors....”

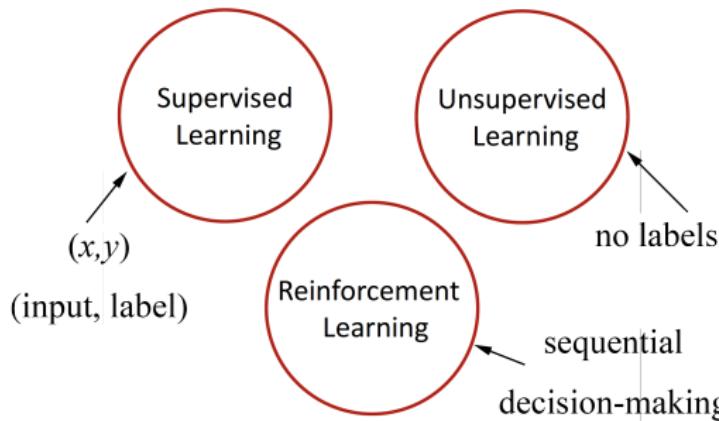


(LeCun et al., 1995)

### NOW



# Taxonomy of Machine Learning



- a simplistic view based on tasks
- can also be viewed as tools/methods

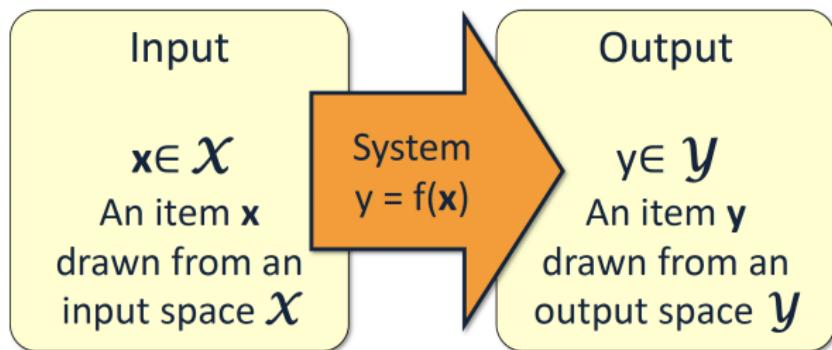
- Yann LeCun's cake analogy at NIPS 2016:
  - If intelligence is a cake, the bulk of the cake is unsupervised learning, the icing on the cake is supervised learning, and the cherry on the cake is reinforcement learning.

# Table of Contents

## 1 Machine Learning

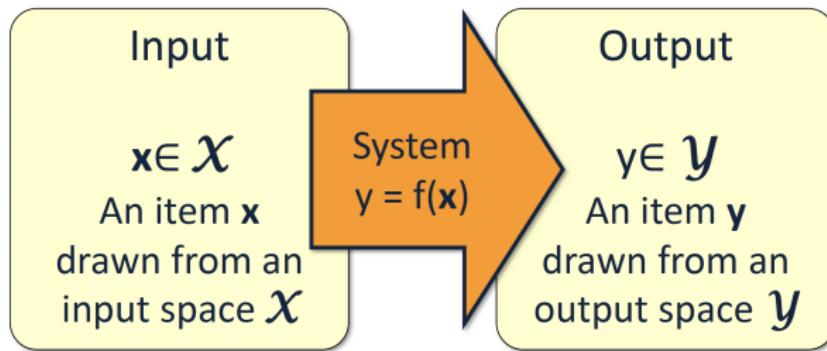
- Supervised Learning, Linear Regression
- Unsupervised Learning, K-means Clustering

# Supervised Learning



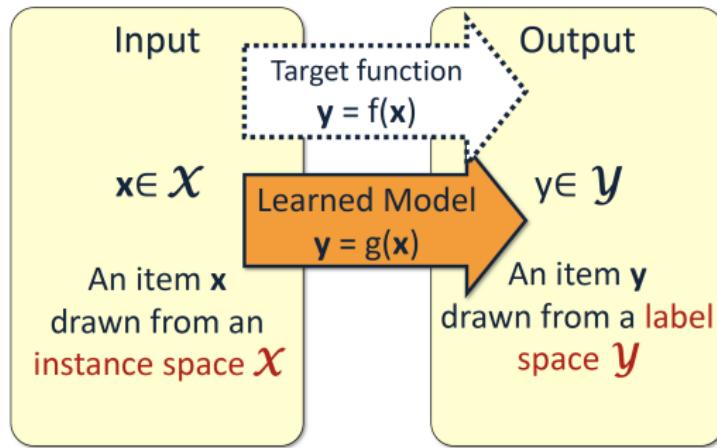
- We consider systems that apply a function  $f(\cdot)$  to input items  $x$  and return an output  $y = f(x)$
- In supervised learning, we deal with systems whose  $f(\cdot)$  is learned from samples  $(x, y)$

# Why Use Learning?



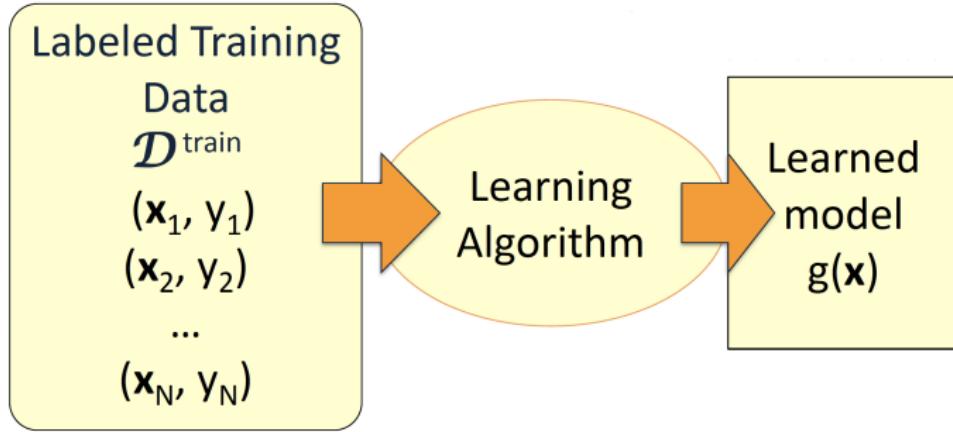
- We typically use machine learning when the function  $f(\cdot)$  we want the system to apply is **unknown** to us, and we cannot “think” about it.
- The function could actually be simple.

# Learned Model - Hypothesis



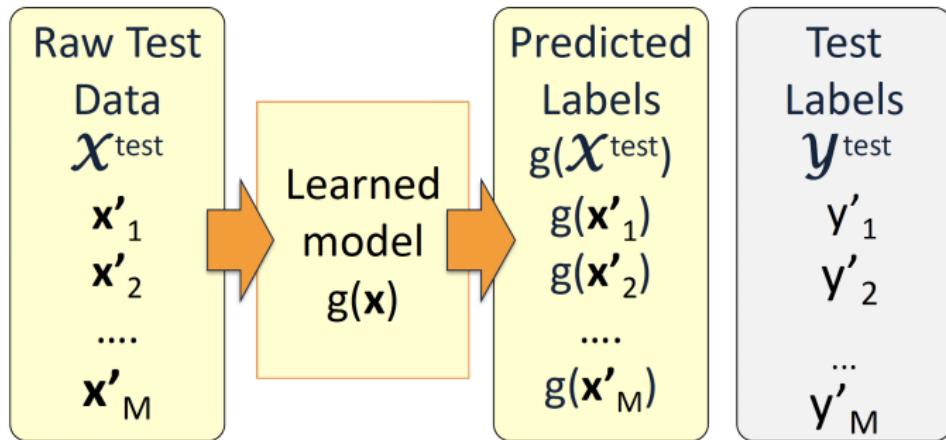
- We need to choose what kind of model we want to learn
  - Linear model, nonlinear model...
  - Parametric model, nonparametric model...
  - Decision trees, neural networks, Gaussian processes...

# Training



- Given the learning samples in  $\mathcal{D}^{\text{train}}$
- The learner returns a model  $g(\cdot)$

# Testing



- Apply the model to raw test data
- Evaluate by comparing predicted labels against the test labels

# Goal of Supervised Learning

- Given: Samples  $(\mathbf{x}, \mathbf{y})$  of some unknown function  
 $f(\cdot) : \mathbf{x} \rightarrow \mathbf{y}$
  - Find: A good approximation of  $f$
- 
- $\mathbf{x}$  provides some representation of the input
    - $\mathbf{x} \in \{0, 1\}^d$  or  $\mathbf{x} \in \mathbb{R}^d$
  - The target function (label)
    - $f(\mathbf{x}) \in \{-1, +1\}$  Binary Classification
    - $f(\mathbf{x}) \in \{1, 2, \dots, k\}$  Multi-class Classification
    - $f(\mathbf{x}) \in \mathbb{R}^k$  Regression

# Linear Regression

The output variable  $y$  as a linear combination of the input variables  $x_1, x_2, \dots, x_d$  plus a noise term  $\epsilon$ :

$$y = \underbrace{\theta_0 + \theta_1 x_1 + \dots + \theta_d x_d}_{g(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}} + \epsilon$$

- $\boldsymbol{\theta} = [\theta_0, \dots, \theta_d]^T$ : model parameters, **to be learned**
- Use linear regression for at least two purposes:
  - **describe** relationships in the data by interpreting  $\boldsymbol{\theta}$
  - **predict** future outputs for inputs that we have not yet seen

## Example: House Price Prediction

$x_1$	$x_2$	$y$
Living area (feet <sup>2</sup> )	#bedrooms	Price (1000\$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
:	:	:

$$g(\mathbf{x}) = \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

# Cost Function

Given a training set, how do we pick, or learn, the parameters  $\theta$ ?



Make  $g(\mathbf{x})$  close to  $y$ , at least for the training examples we have



Mean squared error



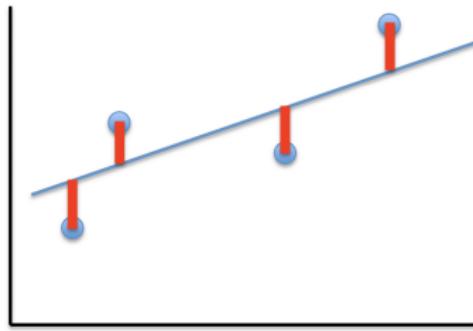
$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (g(\mathbf{x}^i) - y^i)^2$$



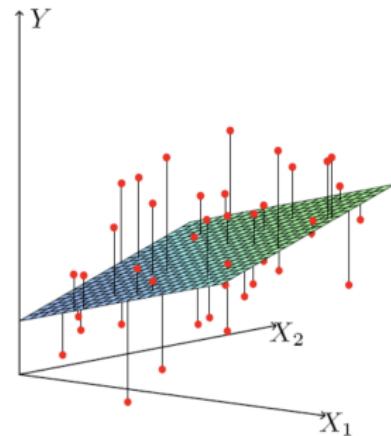
Ordinary least-squares regression model

# Cost Function

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (g(\mathbf{x}^i) - y^i)^2$$



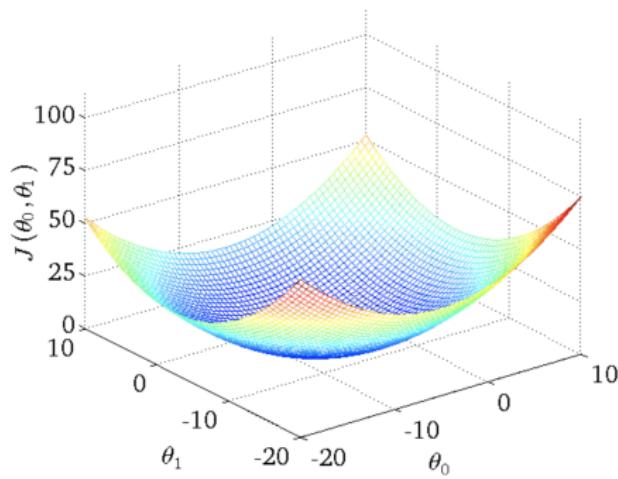
$$\hat{y} = \theta_0 + \theta_1 x$$



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

# Cost Function

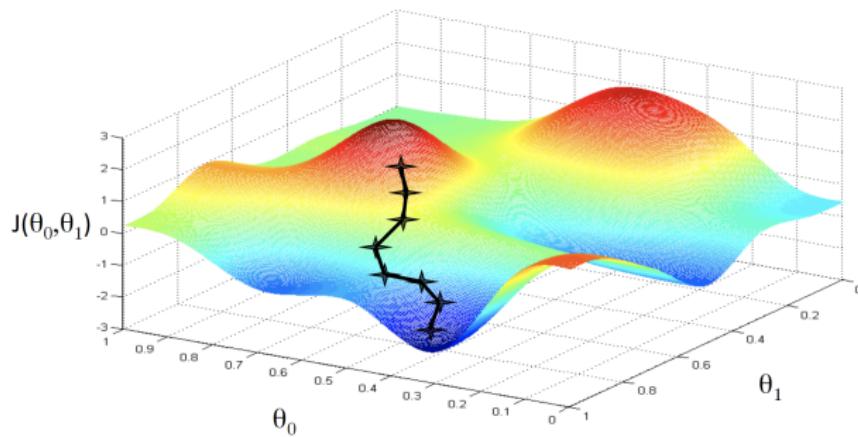
$$\theta^* = \min_{\theta} J(\theta) = \frac{1}{2} \sum_{i=1}^n (g(x^i) - y^i)^2$$



Least-squares cost function is convex!

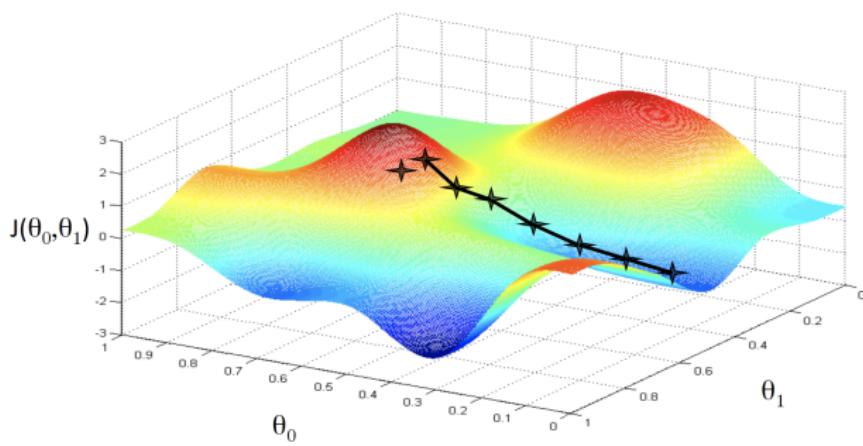
# Gradient Descent

- A search procedure:
  - starts with some “initial guess” for  $\theta$
  - repeatedly changes  $\theta$  to make  $J(\theta)$  smaller
  - until converging to a minimum



# Gradient Descent

- A search procedure:
  - starts with some “initial guess” for  $\theta$
  - repeatedly changes  $\theta$  to make  $J(\theta)$  smaller
  - until converging to a minimum



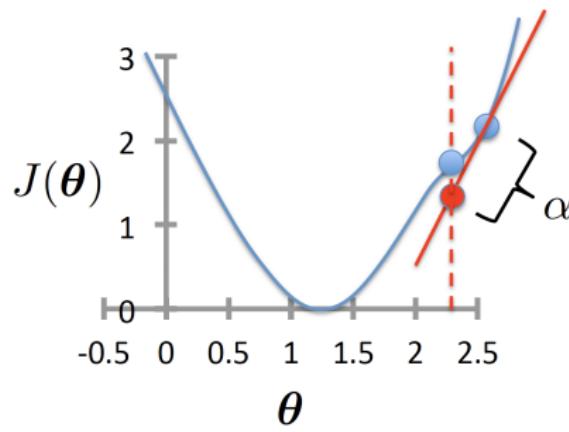
# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update  
for  $j = 0 \dots d$

learning rate (small)  
e.g.,  $\alpha = 0.05$



# Gradient Descent for Linear Regression

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (g(\mathbf{x}) - y)^2 \\&= (g(\mathbf{x}) - y) \cdot \frac{\partial}{\partial \theta_j} (g(\mathbf{x}) - y) \cdot \\&= (g(\mathbf{x}) - y) \cdot \frac{\partial}{\partial \theta_j} \left( \sum_{i=0}^d \theta_i x_i - y \right) \\&= (g(\mathbf{x}) - y) x_j\end{aligned}$$

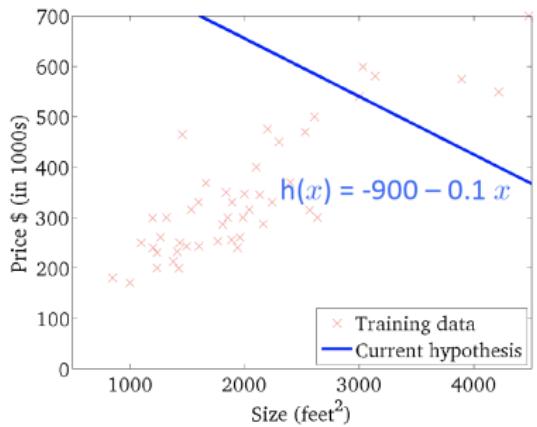
## Least Mean Squares (LMS) update rule

- $\theta_j \leftarrow \theta_j - \alpha \cdot (g(\mathbf{x}^i) - y^i) x_j^i$
- Assume convergence when  $\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$

# Visualization of Gradient Descent

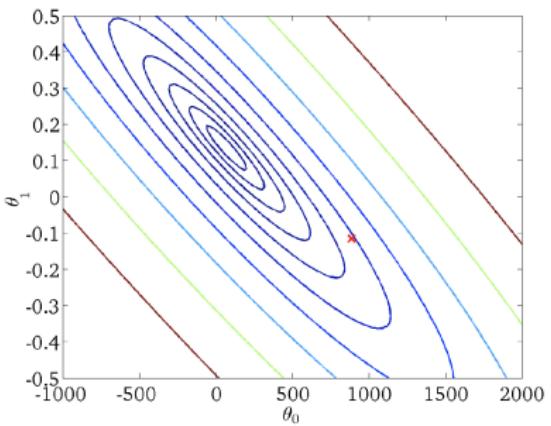
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

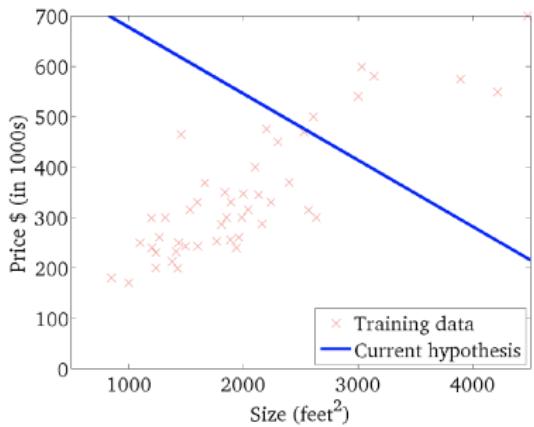
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

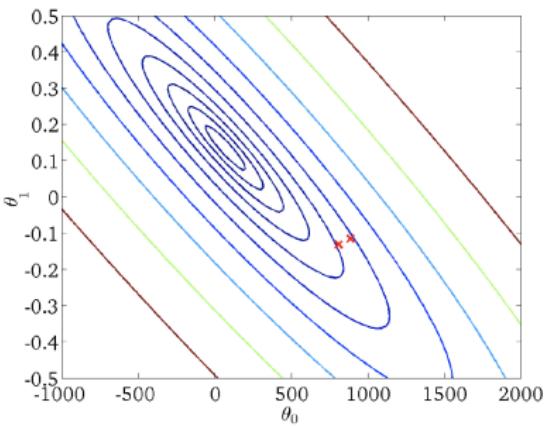
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

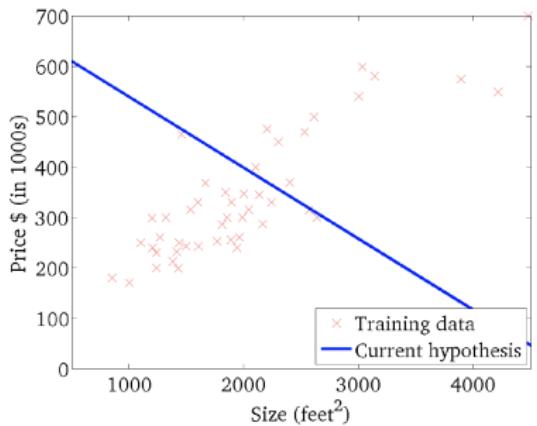
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

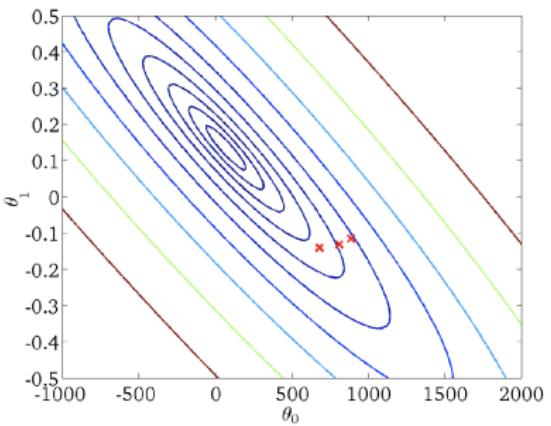
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

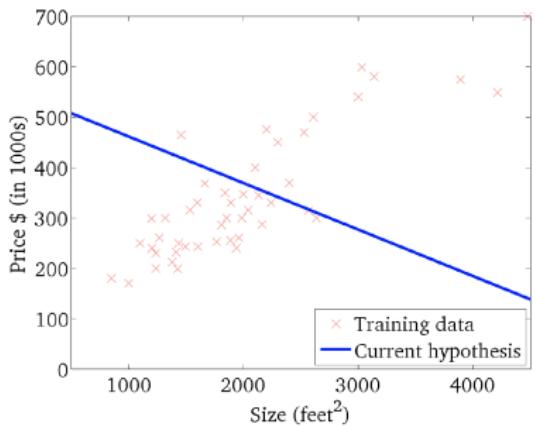
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

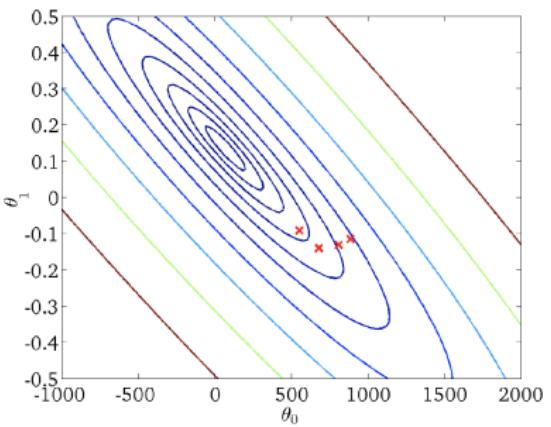
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

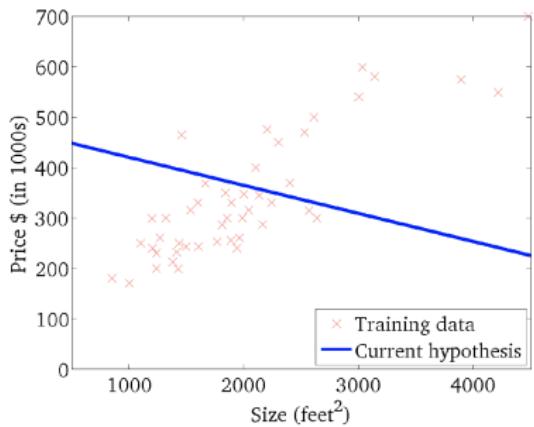
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

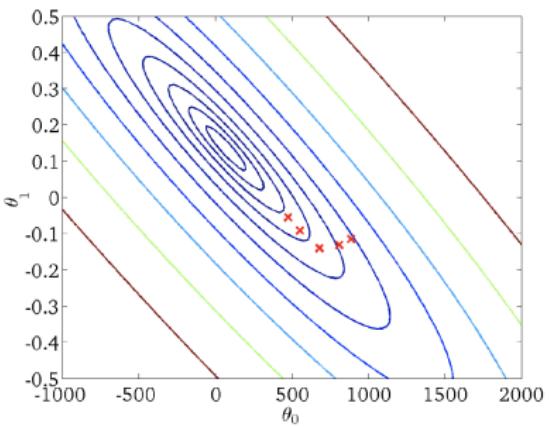
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

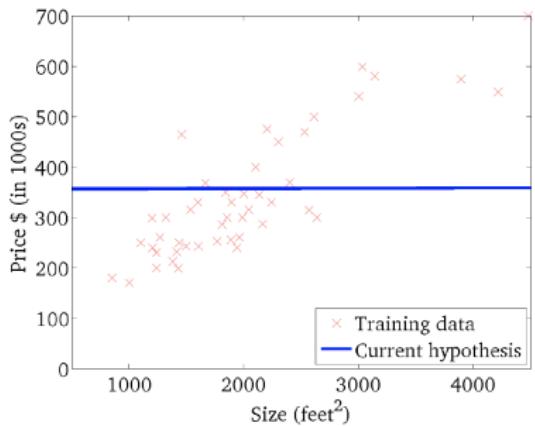
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

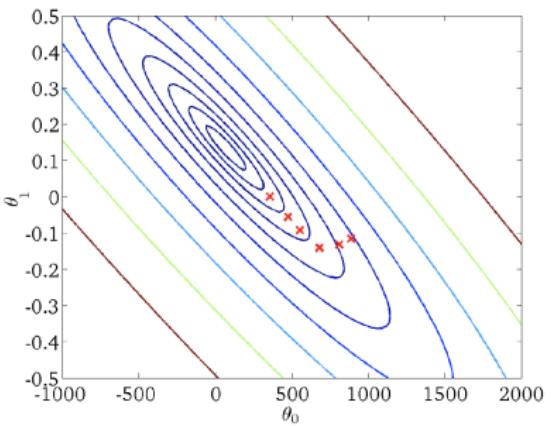
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

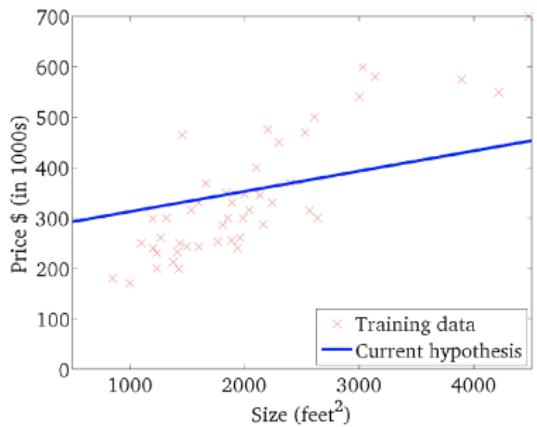
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

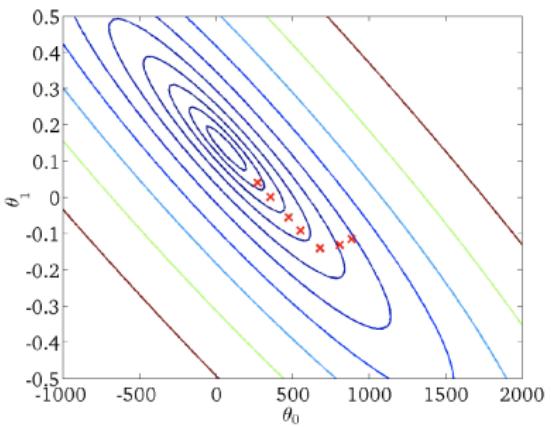
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

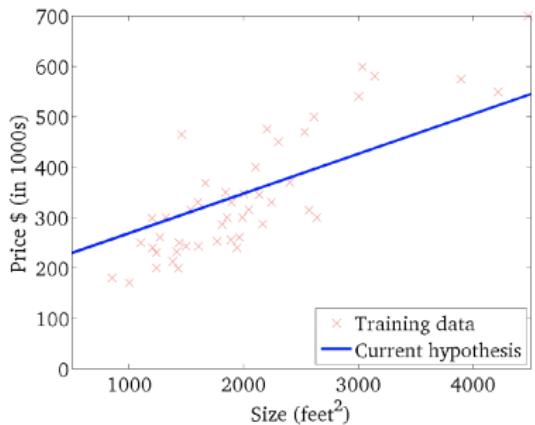
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

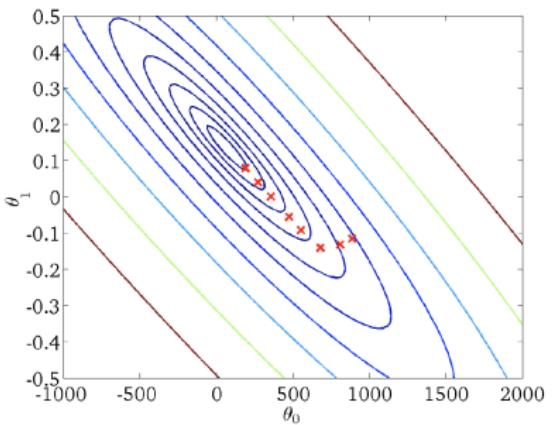
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

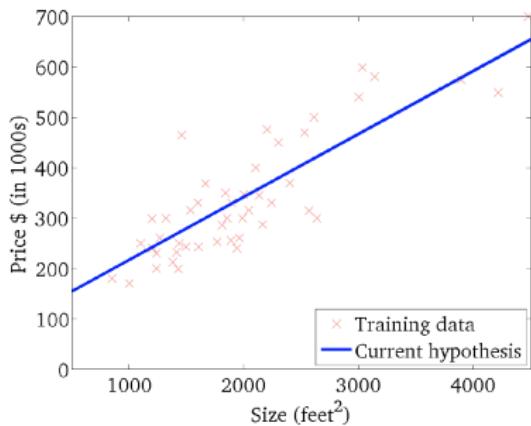
(function of the parameters  $\theta_0, \theta_1$ )



# Visualization of Gradient Descent

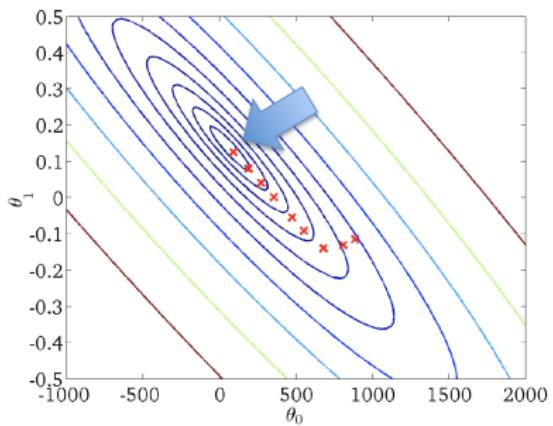
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



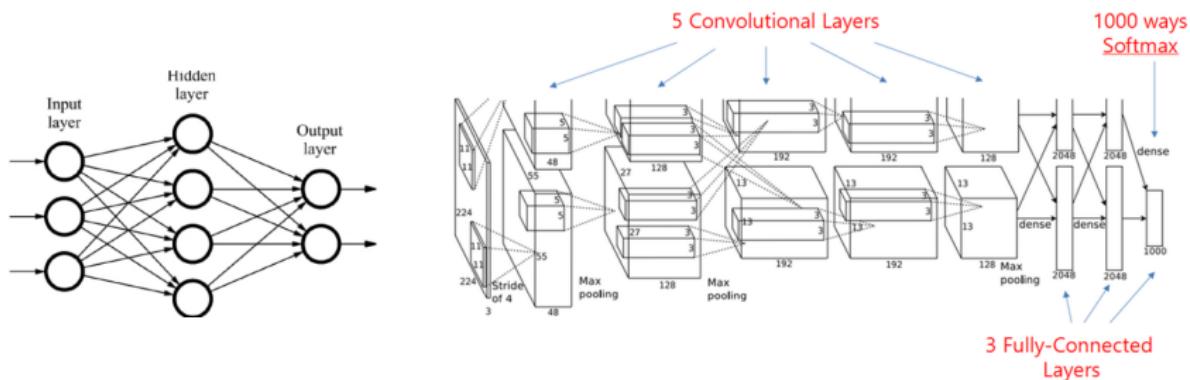
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



# Neural Networks? Learn more after class...

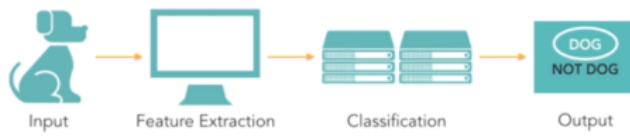
- Input-output pairs  $\{(x_i, y_i)\}$
- Function approximator  $y \approx g(\mathbf{x})$ , using a neural network?
- Cost function, e.g., mean-squared error, cross-entropy loss
- Optimization - gradient descent



# Deep Learning? Learn more after class...

- is part of a broader family of machine learning methods based on artificial neural networks with representation learning
  - deep neural networks, deep belief networks, recurrent neural networks, convolutional neural networks, etc
  - learning can be supervised, semi-supervised or unsupervised
  - in an **end-to-end** manner

## TRADITIONAL MACHINE LEARNING



## DEEP LEARNING



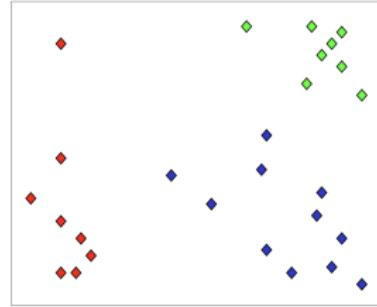
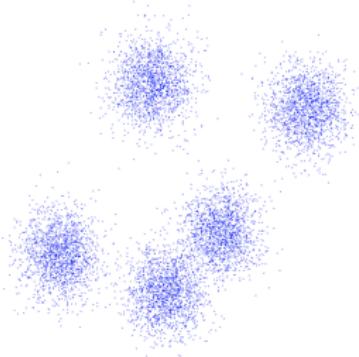
# Table of Contents

## 1 Machine Learning

- Supervised Learning, Linear Regression
- Unsupervised Learning, K-means Clustering

# Unsupervised Learning

- Unsupervised learning is a type of machine learning that **looks for previously undetected patterns** in a data set with **no pre-existing labels** and with a minimum of human supervision.
- Main methods: **Clustering** and **dimensionality reduction**



# Clustering

- **Goal:** Automatically partition **unlabeled** data into groups of similar datapoints.
- **Question:** When and why would we want to do this?
- **Useful for:**
  - Automatically organizing data
  - Understanding hidden structure in data
  - Preprocessing for further analysis, representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes)
- Typical algorithms: K-means, Gaussian mixture models...

# Applications (Clustering comes up everywhere...)

- Cluster news articles or web pages or search results by topic
- Cluster protein sequences by function or genes according to expression profile
- Cluster users of social networks by interest (community detection)
- Cluster customers according to purchase history
- Cluster galaxies or nearby stars (e.g. Sloan Digital Sky Survey)
- And many many more applications...

# K-means Clustering

- **Input:** A set  $S$  of  $n$  points, also a distance/dissimilarity measure specifying the distance  $d(x_i, x_j)$  between pairs  $(x_i, x_j)$ .
- **Goal:** Output a partition of the data.
- **K-means:** Find center points  $c_1, c_2, \dots, c_k$  to minimize

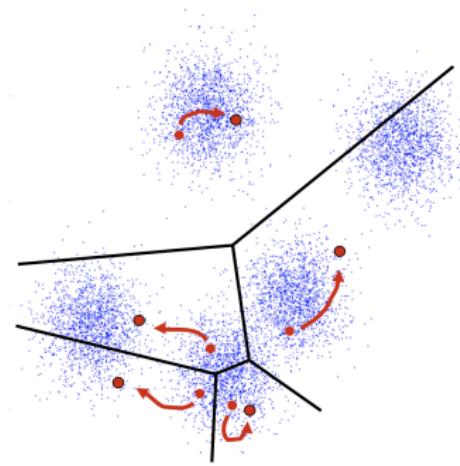
$$\text{minimize} \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} d(x_i, c_j)$$



# Euclidean K-means Clustering

- **Input:** a set of  $n$  datapoints,  
 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  in  $\mathbb{R}^d$ , target number of clusters  $k$ .
- **Output:**  $k$  representatives,  
 $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$
- **Objective:** choose  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$  to minimize

$$\text{minimize} \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|\mathbf{x}_i - \mathbf{c}_j\|^2$$



# The Iterative Method

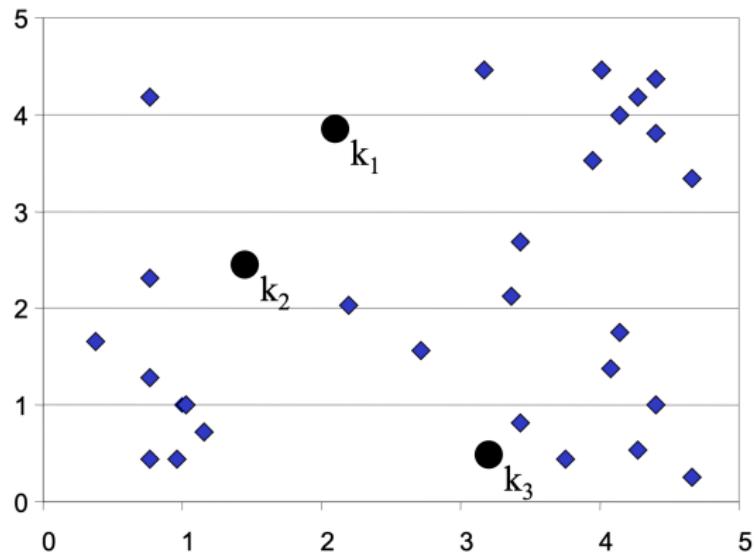
- **Input:** a set of  $n$  datapoints,  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  in  $\mathbb{R}^d$ , target number of clusters  $k$ .
- **Initialize** centers  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \mathbb{R}^d$ , and clusters  $C_1, C_2, \dots, C_k \in \mathbb{R}^d$  in any way
- **Repeat** until there is no further change in the cost
  - for each  $j$  :  $C_j \leftarrow \{ \text{whose closest center is } \mathbf{c}_j \}$
  - for each  $j$  :  $\mathbf{c}_j \leftarrow \text{mean of } C_j$

Holding  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$  fixed,  
pick optimal  $C_1, C_2, \dots, C_k$

Holding  $C_1, C_2, \dots, C_k$  fixed,  
pick optimal  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$

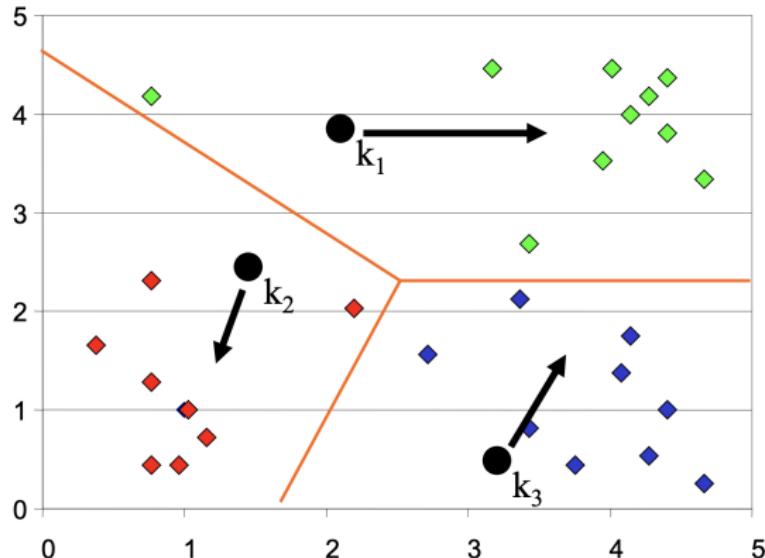
# K-means Clustering: Initialization

- Decide  $k$ , and initialize  $k$  centers (randomly)



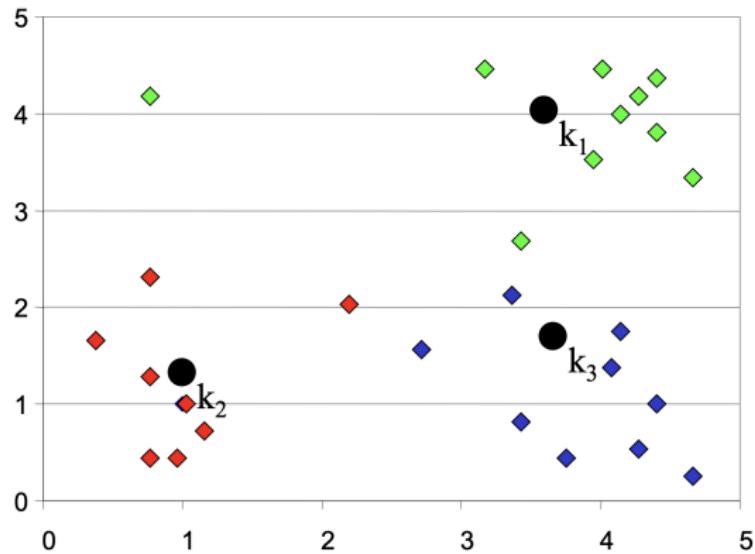
# K-means Clustering: Iteration 1

- Assign all objects to the nearest center. Move a center to the mean of its members.



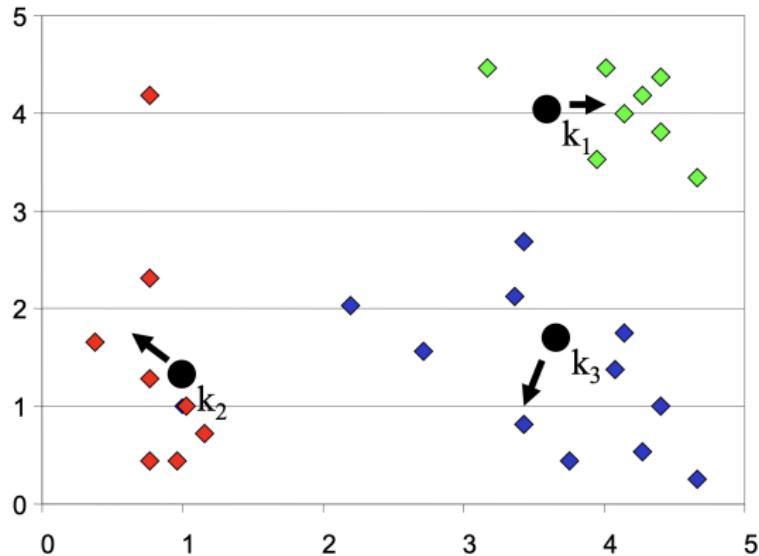
## K-means Clustering: Iteration 2

- After moving centers, re-assign the objects...



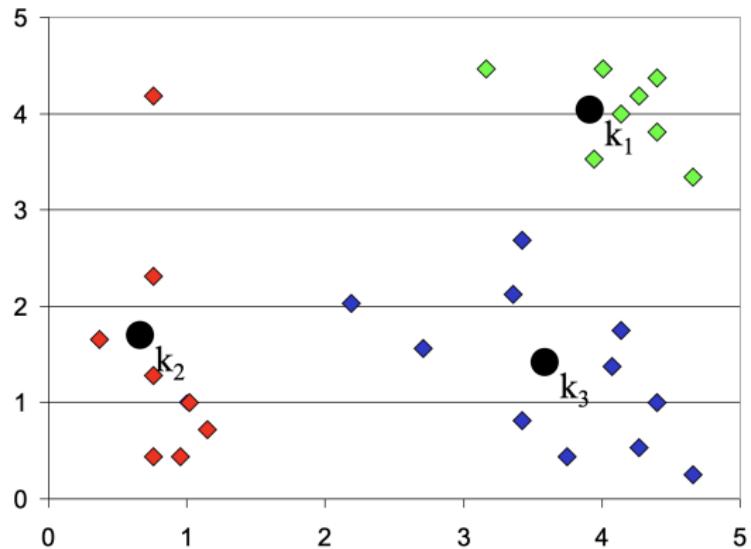
## K-means Clustering: Iteration 2

- After moving centers, re-assign the objects to nearest centers. Move a center to the mean of its new members.

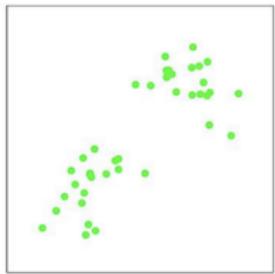


# K-means Clustering: Finished!

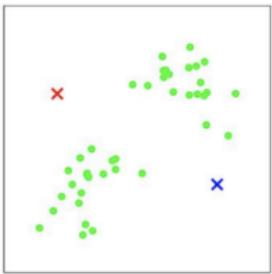
- Re-assign and move centers, until no objects changed membership.



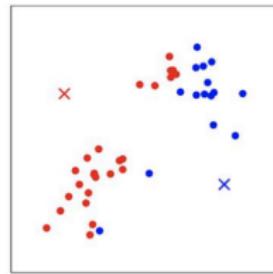
# Example for K-means



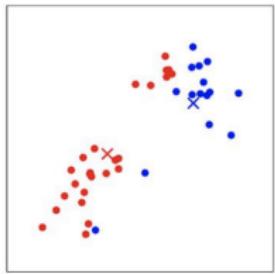
(a)



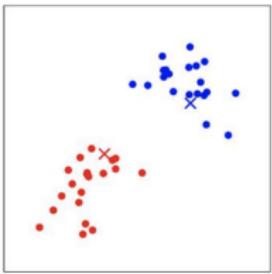
(b)



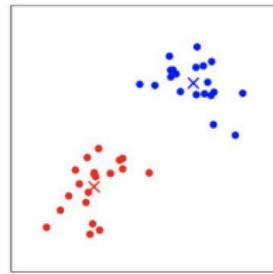
(c)



(d)



(e)



(f)

# Learning objectives of this lecture

You should be able to...

- Know elements of supervised learning and unsupervised learning
- Be familiar with basic algorithms, e.g., linear regression and k-means
- Learn neural networks after class

# References

- Stanford University, *Machine Learning*
  - <http://cs229.stanford.edu/>
- Carnegie Mellon University, *Introduction to Machine Learning*
  - <https://www.cs.cmu.edu/~ninanmf/courses/401sp18/lectures.shtml>

# Homework 1

- Study the following two algorithms after class:
  - Logistic regression for two-class classification, **supervised learning**
  - Gaussian mixture model with expectation-maximization for clustering, **unsupervised learning**
- Write a report (in English) that introduces the two algorithms
  - Theories, explanations, formula derivations, potential applications...
  - Submit the report to [mg20150005@smail.nju.edu.cn](mailto:mg20150005@smail.nju.edu.cn)

# THE END