Lecture 0: Preliminaries of Machine Learning

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

Sept. 17th, 2020

Table of Contents

What we'll cover

- Preliminaries of machine learning
  - Supervised learning: regression, classification
  - Unsupervised learning: clustering, dimensionality reduction
- Finite Markov decision processes (MDPs)
- Tabular RL algorithms:
  - Dynamic programming
  - Monte-Carlo methods
  - Temporal difference learning
- Deep RL algorithms:
  - Policy gradients
  - Actor-critics
  - Value function-based methods
- State-of-the-arts:
  - Transfer RL, meta-RL
  - Evolution strategies for RL
  - Model-based RL
  - Hierarchical RL

Grading

- Homework: 50% (10% per HW x 5 HWs)
  - Preliminaries of machine learning
  - Tabular RL algorithms
  - Policy gradient
  - Actor-critic
  - Deep Q-learning
- Final Project: 50%
  - Make a presentation (with slides in English) of one paper selected from a given list
- Teaching assistant: You Qin
  - All homeworks should be submitted to *MG20150006@smail.nju.edu.cn*

Useful links and references

- Course website
  - http://heyuanmingong.github.io/nju_drl/index.html
- Book
  - Richard S. Sutton, and Andrew G. Barto, *Reinforcement Learning: An Introduction.* 2nd Edition, http://202.119.32.195/cache/2/03/incompleteideas.net/5b682cef88335157bc5542267cf3f442/RLbook2018.pdf
- Famous open classes
  - CS 285 at UC Berkeley, Deep Reinforcement Learning, http://rail.eecs.berkeley.edu/deeprlcourse/
  - CS 234 at Stanford, Reinforcement Learning, http://web.stanford.edu/class/cs234/

Table of Contents

Pre-requisite

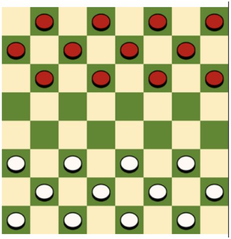
- Probability
  - distribution, random variable, expectation, conditional probability, variance, density
- Linear algebra
  - matrix multiplication
  - eigenvector
- Basic programming (in Python)

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*

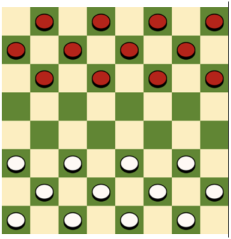
Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress



Definition of Machine Learning

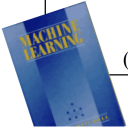
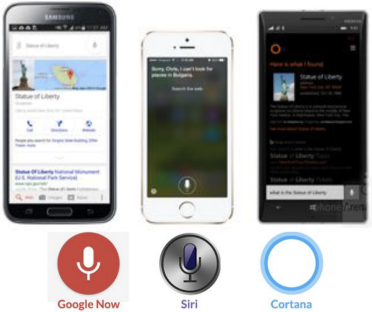
Tom Mitchell (1998): a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

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



Example: Speech Recognition

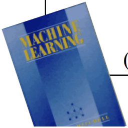
1. Learning to recognize spoken words

THEN	NOW
"…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…"  (Mitchell, 1997)	

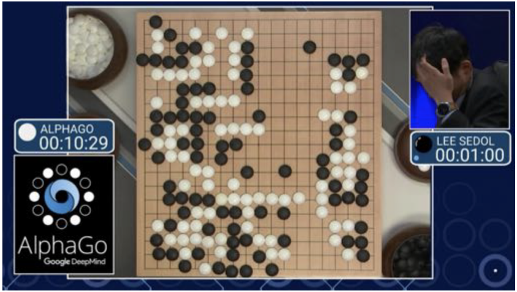
Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults

2. Learning to drive an autonomous vehicle

THEN	NOW
"…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…" (Mitchell, 1997)	https://www.geek.com/wp-content/uploads/2016/03/uber.jpg



3. Learning to beat the masters at board games

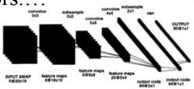
THEN	NOW
"…the world's top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself…" (Mitchell, 1997)	

Example: Computer Vision

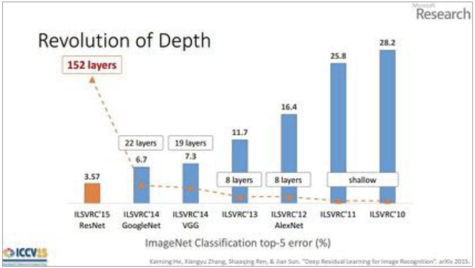
4. Learning to recognize images

THEN	NOW

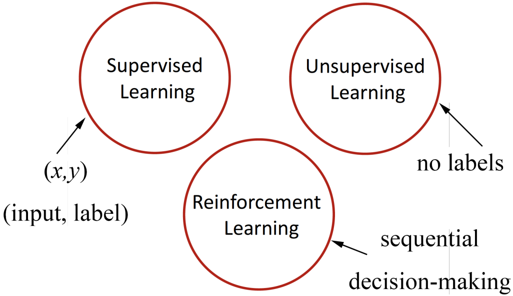
"…The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors…."



(LeCun et al., 1995)



Taxonomy of Machine Learning

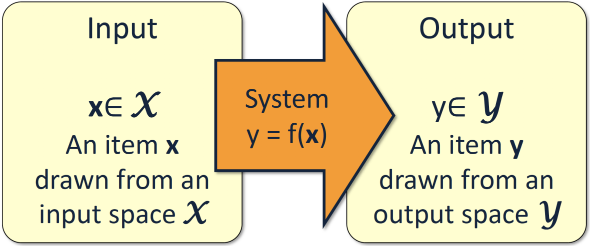


- a simplistic view based on tasks
- can also be viewed as tools/methods

- 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.

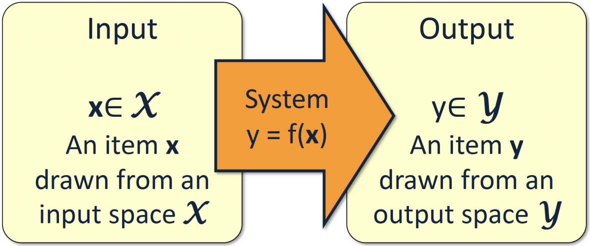
Table of Contents

Supervised Learning



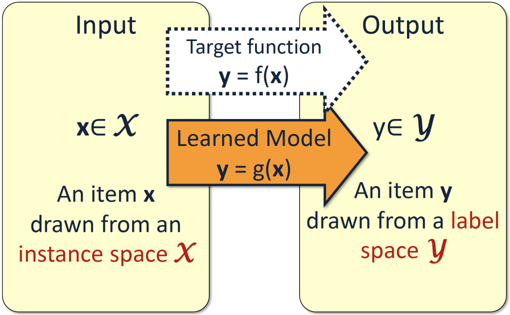
- We consider systems that apply a function $f(\cdot)$ to input items $\boldsymbol{x}$ and return an output $\boldsymbol{y} = f(\boldsymbol{x})$
- In supervised learning, we deal with systems whose $f(\cdot)$ is learned from samples $(\boldsymbol{x}, \boldsymbol{y})$

Why Use Learning?



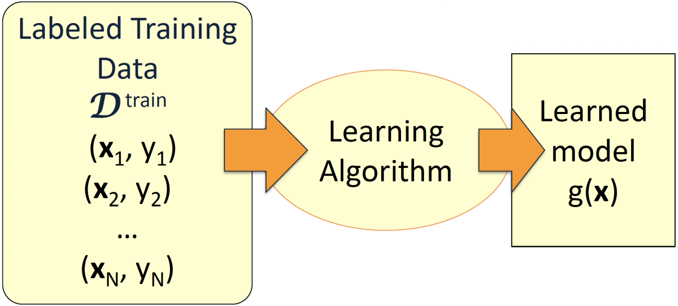
- We typically use machine learning when the function $f(\cdot)$ we want the system to apply is unknown to us, and we cannot "think" about it.
- The function could actually be simple.

Learned Model - Hypothesis



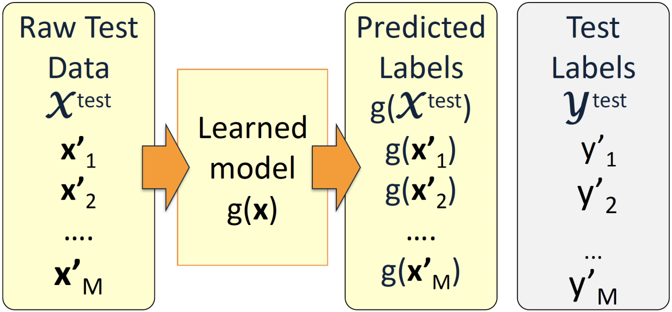
- We need to choose what kind of model we want to learn
  - Linear model, nonlinear model...
  - Parametric model, nonparametric model...
  - Decision trees, neural networks, Gaussian processes...

Training



- Given the learning samples in $\mathcal{D}^{train}$
- 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 $(\boldsymbol{x}, \boldsymbol{y})$ of some unknown function $f(\cdot) : \boldsymbol{x} \rightarrow \boldsymbol{y}$
- Find: A good approximation of f

- $\boldsymbol{x}$ provides some representation of the input
  - $\boldsymbol{x} \in \{0, 1\}^d$ or $\boldsymbol{x} \in \mathbb{R}^d$
- The target function (label)
  - $f(\boldsymbol{x}) \in \{-1, +1\}$ Binary Classification
  - $f(\boldsymbol{x}) \in \{1, 2, ..., k\}$ Multi-class Classification
  - $f(\boldsymbol{x}) \in \mathbb{R}^k$ Regression

Linear Regression

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

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

- $\boldsymbol{\theta} = [\theta_0, ..., \theta_d]^T$: 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

Living area (feet^2) $\to x_1$	#bedrooms $\to x_2$	Price (1000\$s) $\to y$
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

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

Cost Function

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

$$\Downarrow$$

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

$$\Downarrow$$

Mean squared error

$$\Downarrow$$

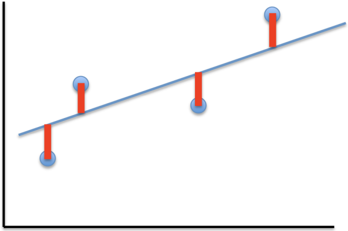
$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(g(\boldsymbol{x}^i) - y^i)^2$$

$$\Downarrow$$

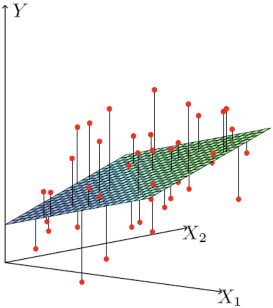
Ordinary least-squares regression model

Cost Function

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



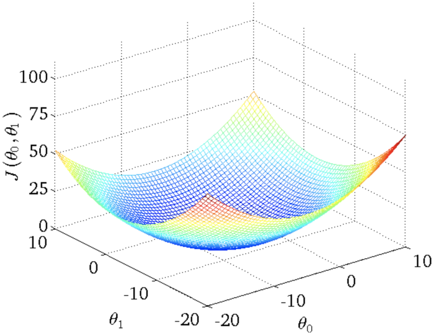
$\hat{y} = \theta_0 + \theta_1 x$



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

Cost Function

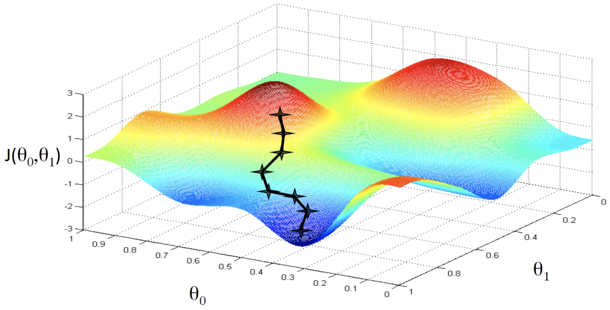
$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(g(\boldsymbol{x}^i) - y^i)^2$$



Least-squares cost function is convex!

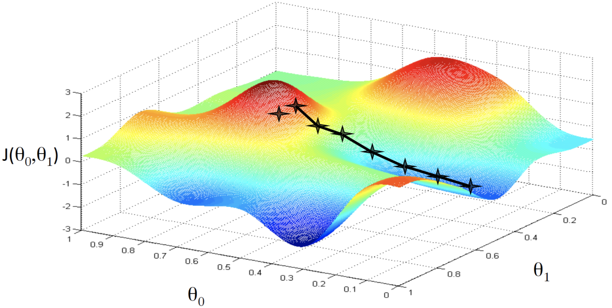
Gradient Descent

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



Gradient Descent

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

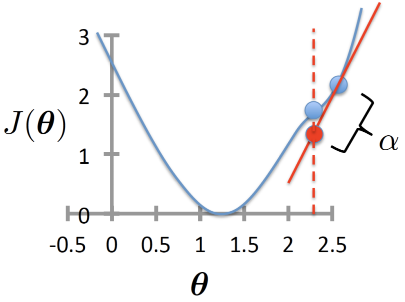


- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

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

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



19

Gradient Descent for Linear Regression

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

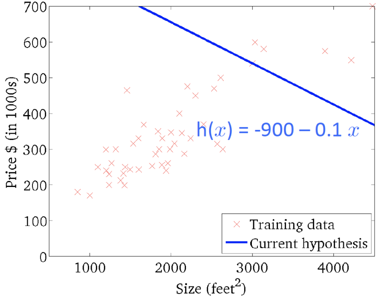
Least Mean Squares (LMS) update rule

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

Visualization of Gradient Descent

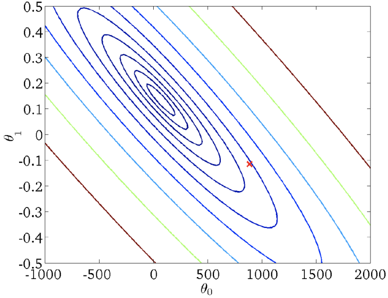
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

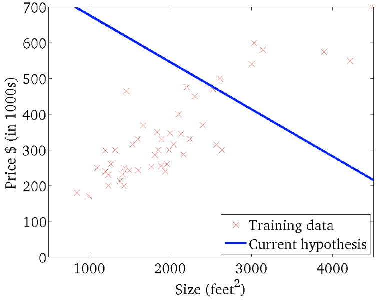
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

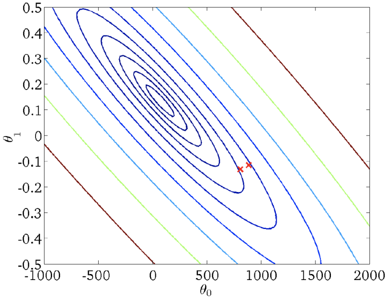
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

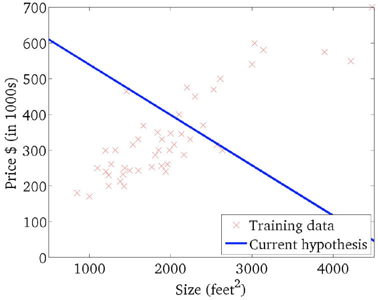
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

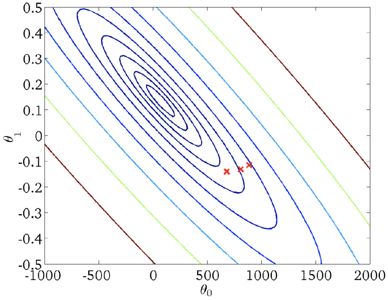
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

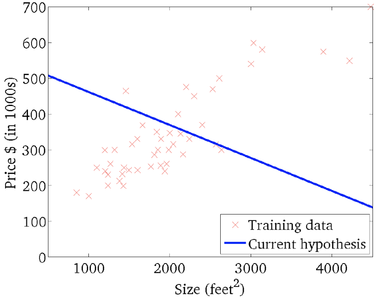
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

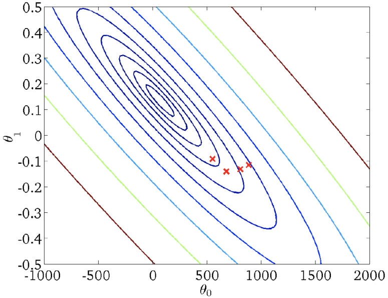
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

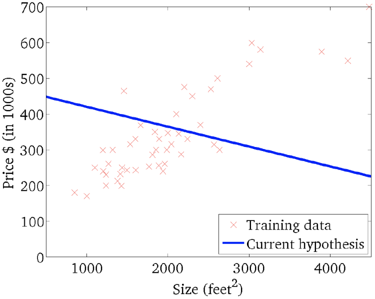
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

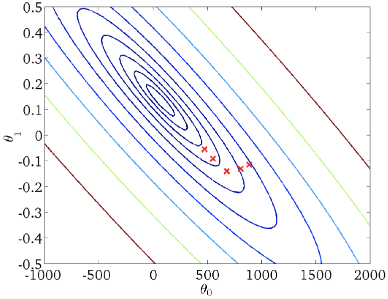
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

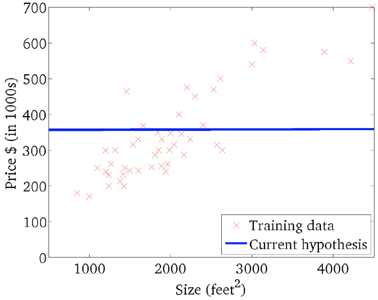
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

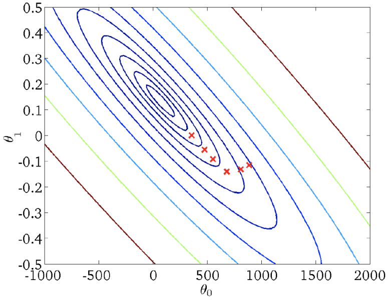
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

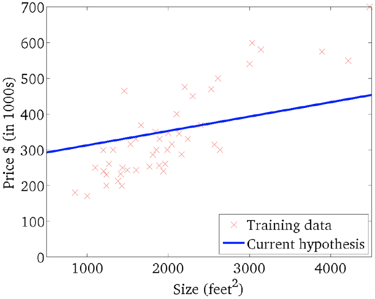
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

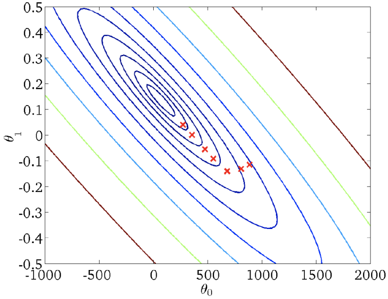
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

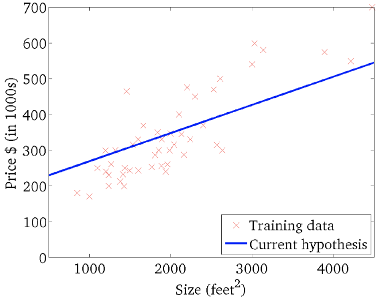
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

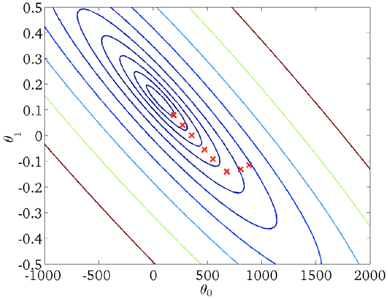
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

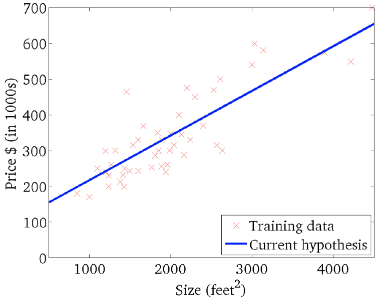
(function of the parameters θ_0, θ_1)



Visualization of Gradient Descent

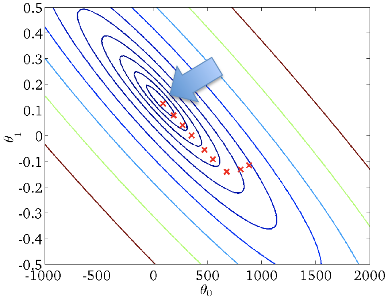
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



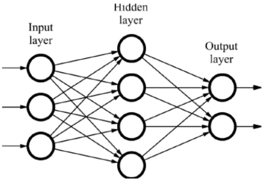
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



Neural Networks? Learn more after class...

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



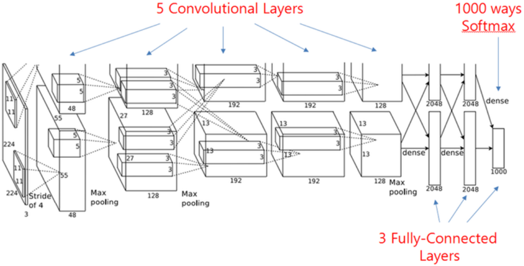
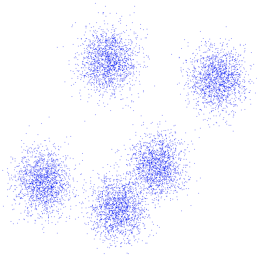
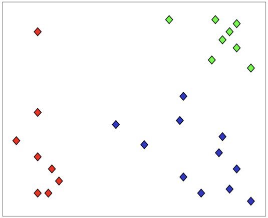


Table of Contents

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

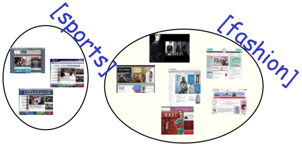
- **Goal**: Automatically partition unlabeled data into groups of similar datapoints.
- **Question**: When and why would we want to do this?
- **Useful for**:
  - Automatically organizing data
  - Understanding hidden structure in data
  - Preprocessing for further analysis, representing high-dimensional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

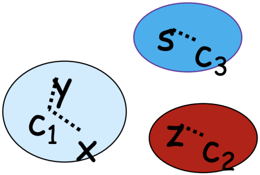
Applications (Clustering comes up everywhere...)

- 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...

- **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 $c_1, c_2, ..., c_k$ to minimize

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

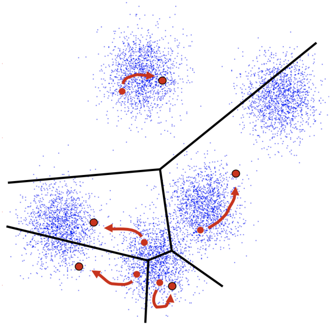




Euclidean K-means Clustering

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

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



The Iterative Method

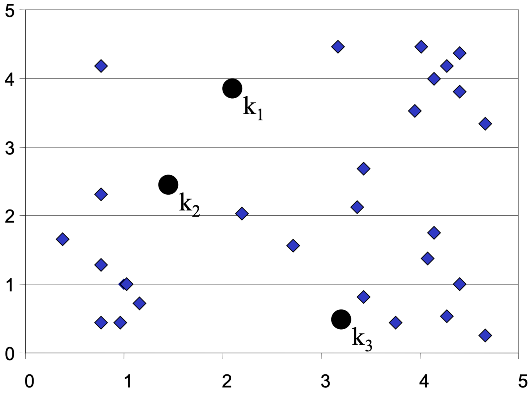
- **Input**: a set of n datapoints, $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ in $\mathbb{R}^d$, target number of clusters k.
- **Initialize** centers $\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k \in \mathbb{R}^d$, and clusters $\boldsymbol{C}_1, \boldsymbol{C}_2, ...,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

Holding $\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k$ fixed, pick optimal $\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_k$

Holding $\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_k$ fixed, pick optimal $\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k$

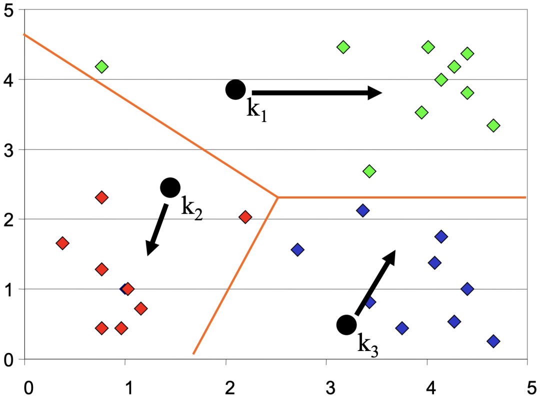
K-means Clustering: Initialization

- Decide k, and initialize k centers (randomly)



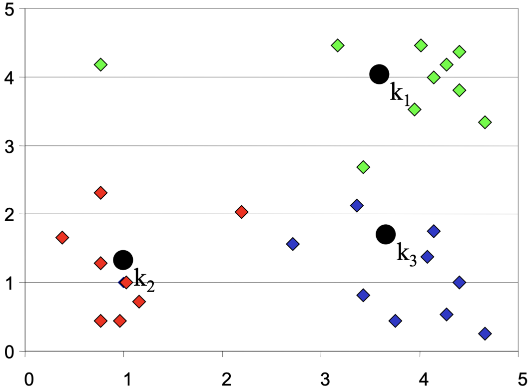
K-means Clustering: Iteration 1

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



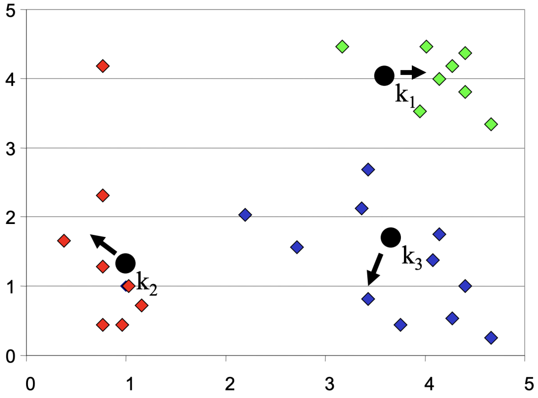
K-means Clustering: Iteration 2

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



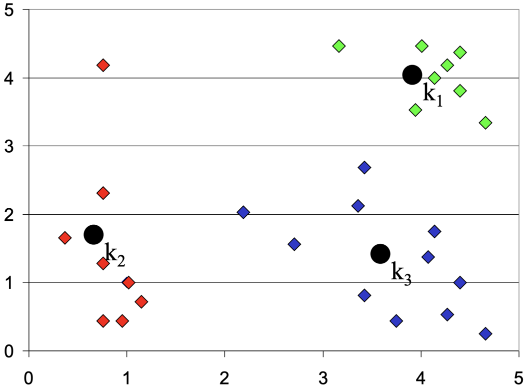
K-means Clustering: Iteration 2

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

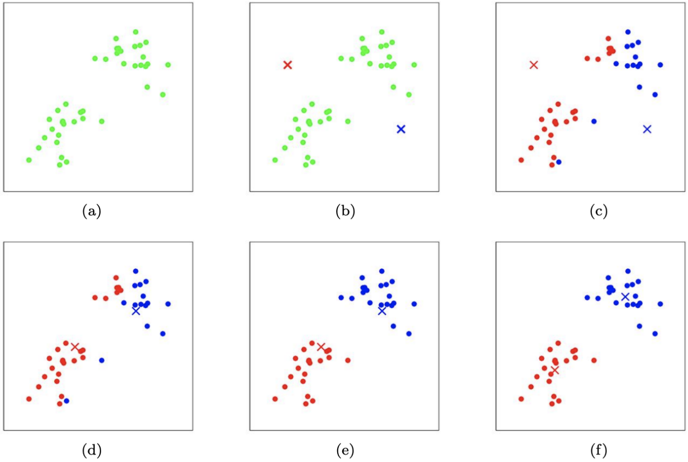


K-means Clustering: Finished!

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



Example for K-means



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/~ninamf/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 that introduces the two algorithms
  - Theories, explanations, formula derivations, potential applications...
  - Submit the report to *MG20150006@smail.nju.edu.cn*

THE END