

# Tools Seminar

## Week 7 - Machine Learning

Hongzheng Chen

Mar 9, 2020

- 1 Introduction
- 2 Some Introductory Books & Courses
- 3 Machine Learning Basis
- 4 Data Processing
- 5 Pandas & Sklearn
- 6 Summary

1

# Introduction

# Machine Learning

*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  $E$ .*

— Mitchell, 1997

Take exam as an example

- $T$ : To obtain high score
- $E$ : Do exercise
- $P$ : Accuracy of exercise

# Machine Learning Branches

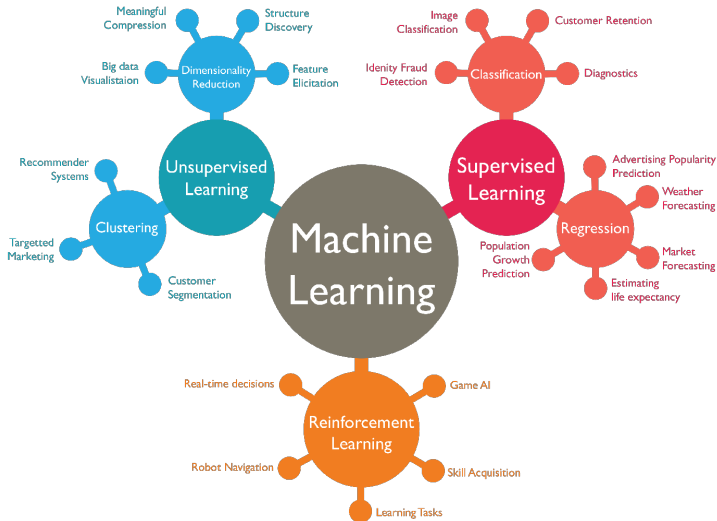


Fig source: <https://askdatascience.com/13/>

# Some Ideas to Clarify

**Artificial Intelligence (AI)**  
    > **Machine Learning (ML)**  
        > **Deep Learning (DL)**

- ML is not the only way to achieve AI
- DL is just a method of ML
- Neural Network (NN) is the core of DL
- Computer Vision (CV) & Natural Language Processing (NLP) are two applications of DL

## 2

# Some Introductory Books & Courses

# Machine Learning

Prerequisite: [Linear algebra](#), multivariable calculus, [probability theory](#)

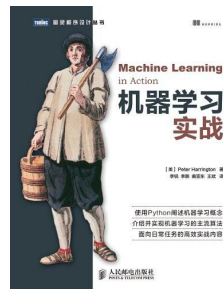
- Andrew Ng, [Stanford CS229: Machine Learning](#)
- Hsuan-Tien Lin, [NTU: Machine Learning Foundations](#)

You can find them on Bilibili!

\* YJango, [《学习观》](#)



# Chinese Books



# Other books

- Christopher M. Bishop, [Pattern Recognition & Machine Learning \(PRML\)](#)
- Stephen Boyd, Lieven Vandenberghe, [Convex Optimization](#)
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, [Deep Learning](#) (flower book)
- Richard S. Sutton, Andrew G. Barto, [Reinforcement Learning](#)

# Top-tier Conferences in AI Area

- General AI: AAAI, IJCAI
- NLP: ACL, EMNLP, NAACL
- CV: CVPR, ICCV, ECCV
- ML: NeurIPS/NIPS, ICML, ICLR
- Data mining: KDD, SIGMOD, ICDE, VLDB, SIGIR, WWW
- System: SysML

3

# Machine Learning Basis

# Machine Learning Basis

For **supervised learning** (classification & regression), we have

- Training set

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$$

- Sample:  $(\mathbf{x}_i, y_i)$
- Input/Feature/Attribute:  $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$
- Output/Label/Target:  $y_i \in \mathcal{Y} \subset \mathbb{R}$
- Model: Want to learn a mapping from  $\mathcal{X}$  to  $\mathcal{Y}$

$$f : \mathcal{X} \mapsto \mathcal{Y}$$

For example, consider face recognition

- Input: Many students' faces in 2D figures
- Output: The name of the student
- Model:  $f(\text{face}) = \text{student}$

Differences between different ML alg. are how to determine  $f$

# Linear Regression

Consider sample with  $d$  features

$$\mathbf{x} = (x_1, x_2, \dots, x_d)$$

Use **linear combination** of features as model

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_dx_d + b = \mathbf{w}^T \mathbf{x} + b \simeq y$$

where  $\mathbf{w}$ ,  $b$  are the parameter needs to be learned

**Loss function**  $J(\mathbf{w}, b)$ : measure the performance of the model, we use **mean square error (MSE)** here

$$J(\mathbf{w}, b) = \|f(\mathbf{x}_i) - y_i\|_2^2$$

Then **objective** is to optimize

$$\min_{\mathbf{w}, b} \left( \sum_{i=1}^m J(\mathbf{w}, b) \right) = \min_{\mathbf{w}, b} \left( \sum_{i=1}^m \|f(\mathbf{x}_i) - y_i\|_2^2 \right)$$

# Linear Regression

The best parameters are what we want

$$(\mathbf{w}^*, b^*) = \arg \min_{\mathbf{w}, b} \left( \sum_{i=1}^m \|f(\mathbf{x}_i) - y_i\|_2^2 \right)$$

Commonly, we use **gradient descent** to optimize, which is an iterative process

$$\begin{cases} \mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{w}} & = \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}, b) \\ b^{(k+1)} = b^{(k)} - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial b} & = b^{(k)} - \alpha \nabla_b J(\mathbf{w}, b) \end{cases}$$

If loss cannot be reduced more (converged), we find the optimal  $\mathbf{w}^*$  and  $b^*$   
The final model becomes

$$f(\mathbf{x}) = (\mathbf{w}^*)^T \mathbf{x} + b^*$$

# Linear Regression

But for MSE of linear regression, you can easily find close-form solution by **least square method**

- Stack the samples

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^T & 1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \hat{\mathbf{w}} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ b \end{bmatrix}$$

- Optimize

$$\hat{\mathbf{w}}^* = \arg \min_{\hat{\mathbf{w}}} (\mathbf{y} - X\hat{\mathbf{w}})^T (\mathbf{y} - X\hat{\mathbf{w}})$$

- Make derivative as 0

$$\frac{\partial J(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} = 2X^T(X\hat{\mathbf{w}} - \mathbf{y}) = 0$$

- Solve for best  $\hat{\mathbf{w}}$  (if  $X^T X$  has inverse)

$$\hat{\mathbf{w}}^* = (X^T X)^{-1} X^T \mathbf{y}$$



# Summary

- ① Obtain required training and testing data
- ② Determine the objective of the task
- ③ Select a machine learning model to train
- ④ Use pre-trained model to predict

## 4

## Data Processing

# Machine Learning Today

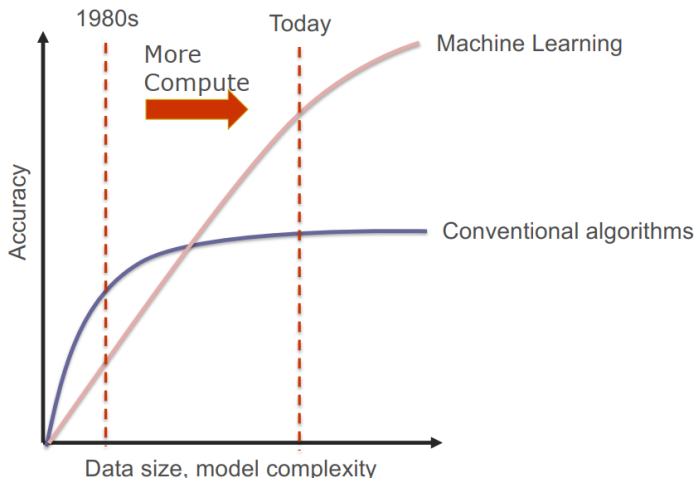
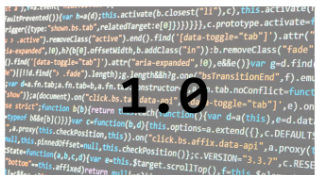


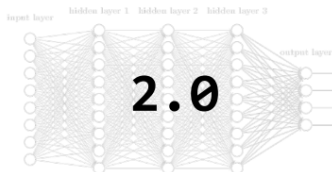
Fig source: Jeff Dean, HotChips 2017

## Software 2.0 Era

Training data: The new input to software 2.0



- Input: Algorithms in code
- Compiled to: Machine instructions



- Input: Training data
- Compiled to: Learned parameters

Fig source: Kunle Olukutun, ISCA, 2018

Thus, **quality & quantity** of the data determine performance!

# Common Data Science Steps

- ① Data collection
- ② Feature engineering (Data cleaning/preprocessing)
- ③ Model selection
- ④ Training
- ⑤ Evaluation

# Data Collection

In competitions, we commonly have official datasets  
But what if no collected datasets available in some specific tasks?  
(e.g. face mask recognition)

# Data Collection

In competitions, we commonly have official datasets

But what if no collected datasets available in some specific tasks?  
(e.g. face mask recognition)

## Design a web crawler!

- 1 Crawl the webpages (requests, urllib)
- 2 Parse html (bs4)
- 3 Retrieve useful data (text, figure, or other specific content)
- 4 Organize the data
- 5 Store them into files (mysql, json)

# A Dataset Example

	Feat 1	Feat 2	...	Feat $d$	Label
Sample 1	$x_{11}$	$x_{12}$			$y_1$
$\vdots$					$\vdots$
Sample $m$					$y_m$



# Data Cleaning

Data cleaning or feature engineering is **the first step** of most ML tasks!  
Most datasets are troublesome: (think about questionnaires)

- Data missing
- Redundant data
- Data not in same scale
- Useless features
- Too many features
- . . .

ML is also called **representative learning**

Good data and good features benefit learning process

5

# Pandas & Sklearn

# Some Hints

Use jupyter notebook for interactive operations, see demo

- You should carefully deal with the data which may be very dirty
- For models with hyperparameters, use [grid search](#) to fine-tune

6

# Summary

# Summary

- Introduction
- Machine learning basis: Linear regression
- Data processing: pandas
- Training & Testing: sklearn