Topics in ML
or
Introduction to Machine Learning

Akito Sakurai
Professor, YNU
Professor Emeritus, Keio University

Course information

- Course Name: Topics in Machine Learning
- Code: WAT0118
- Instructor: Prof. Akito Sakurai
- Time: Thursday, 3rd Lecture (13:00-14:30)
- Location: General Research Building, S9-1, Room E205
- TA: Dr. Mehboob Rasul
- Please note: For this class please bring your laptops.
- Class notes: pdf files will be posted somewhere.
- Prerequisites:
  - Basic (some) understandings of statistics/probability
  - Some experience of any computer languages

Class overview

- Class Schedule

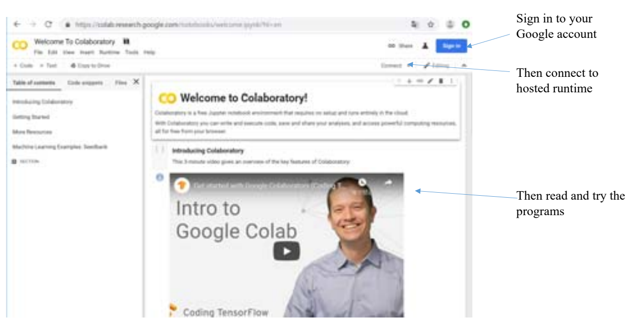
no.	date	topic
1	2019/10/10	introduction
	2019/10/17	no class
2	2019/10/24	nearest neighbor
3	2019/10/31	decision tree and overfitting
4	2019/11/7	naïve bayes and baysian method
5	2019/11/14	model selection
6	2019/11/21	support vector machine
7	2019/11/28	boosting
8	2019/12/5	k-means and em
9	2019/12/12	(buffer)
10	2019/12/19	neural networks and bp
11	2020/1/9	deep learning: conv net
12	2020/1/16	autoencoder and GAN
13	2020/1/23	visualization and grad cam
14	2020/1/30	word embedding

Organization of this lecture

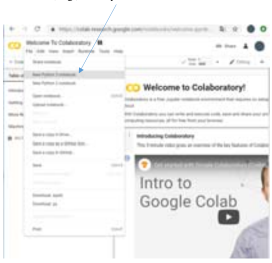
- The lecture includes two sets of topics
  - First half: Machine learning (other than deep learning)
  - Second half: Deep learning
- Machine learning
  - Basic concepts
    - Generalization / Model complexity
  - Methods
    - Decision tree / Nearest neighbor / Support vector machine / Boosting
- Deep learning
  - Basic concepts
    - Neural network / backpropagation
  - Methods
    - Convolutional .

Exercise Environment

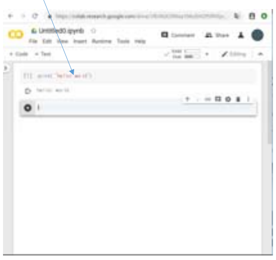
- In the lecture, Google Colaboratory will be used.
  - You are requested to have a Google account.
  - You do not need to prepare e.g. Ubuntu and GPU environments
  - It is free of charge.
  - The language to be used is R (not Python).
    - See the next slide.
    - To use the language R, you need a couple of techniques.
  - Google Colaboratory is a good choice when you do not have a PC with GPU. If you do have, it would be nice to set up R/RStudio and Anaconda environment by yourself.



Select, e.g., New Python 3 notebook



Run a Python program in the notebook by ctrl+enter or shift+enter



A way to use R with Google Colaboratory

1. Create a notebook
2. Download it by
   File → Download .ipynb,
3. Rewrite two lines it as:
4. Upload it by
   File → Upload notebook
5. Or just upload R.ipynb

```
"kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
 }
```

```
"kernelspec": {
   "name": "ir",
   "display_name": "R"
 }
```

https://stackoverflow.com/questions/54595285/how-to-use-r-with-google-colaboratory

A test:
Install a package "keras" and use it.

```
[4] install.packages("keras")

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
also installing the dependencies 'config', 'reticulate', 'tensorflow', 'tfruns'

[5] library(keras)

[6] mnist <- dataset_mnist()
    train_images <- mnist$train$x
    train_labels <- mnist$train$y
    test_images <- mnist$test$x
    test_labels <- mnist$test$y
```


Another way

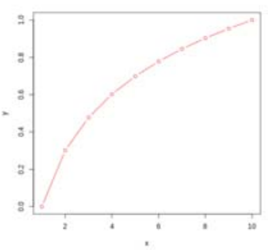
You insert %load_ext rpy2.ipython in the first cell; and then
insert %%R at the first line of *each* cell.
(cells without %%R are Python cells)

Note: For unknown reason, dataset_mnist() crashes kernel after successful installation of package keras. (memory problem?)

```
[1] %load_ext rpy2.ipython

[2] %%R
    x <- 1:10
    y <- log10(x)

[3] %%R
    plot(x,y,type="b",col="red")
```



Other methods to use R

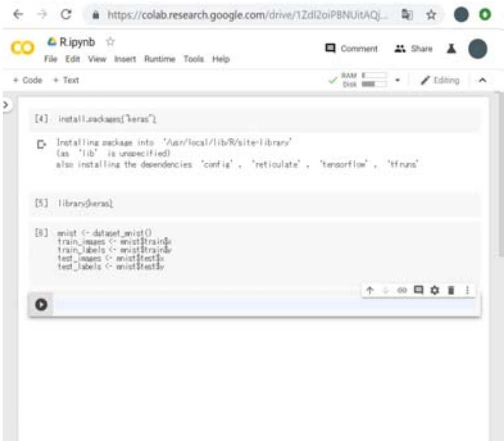
- Install R itself.
  - You could install native R on Windows, Mac, and Unix.
  - The R Project for Statistical Computing https://www.r-project.org/
- Install R with some development environment
  - RStudio https://www.rstudio.com/products/rstudio/download/
  - Anaconda The World's Most Popular Data Science Platform https://www.anaconda.com/

Text books and/or resource

- No textbooks
- References
  - Machine Learning with R https://github.com/SharmaNatasha/Books/blob/master/Machine%20Learning%20with%20R%2C%20Second%20Edition.pdf
  - Introduction to Statistical Learning with Applications in R http://faculty.marshall.usc.edu/gareth-james/
  - François Chollet with J. J. Allaire, Deep Learning with R
    - "Notebooks" in the next slide.

Deep learning examples

- Codes in "Deep Learning with R" https://github.com/jjallaire/deep-learning-with-r-notebooks are converted to Google Colaboratory Jupyter notebook format.
- You may upload and test them.
- Note:
  - Some of them were erroneous. They were corrected by comparing with "Deep Learning with Python" and accompanying notebooks.
  - Some of them crash Google Colaboratory's R kernel because of memory limitation.



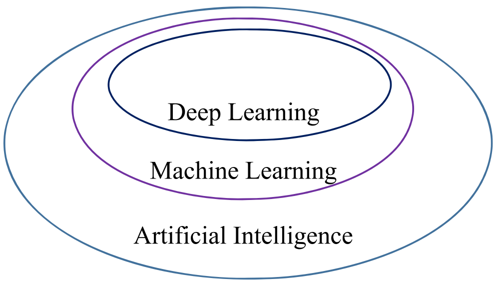
What is machine learning?

- ML is methodology to make machines (computers) become smarter by itself.
- Arthur Samuel (1959):
  … a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program.
  Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.

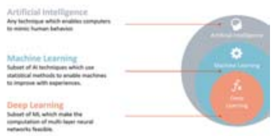
  (the following famous quote is not verified:
  ML: Field of study that gives computers the ability to learn without being explicitly programmed.)

AI, ML, and DL

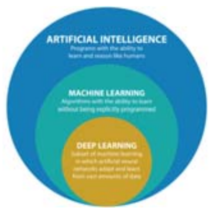
No. 1



https://www.geospatialworld.net/blogs/difference-between-ai%EF%BB%BF-machine-learning-and-deep-learning/



https://rapidminer.com/blog/artificial-intelligence-machine-learning-deep-learning/



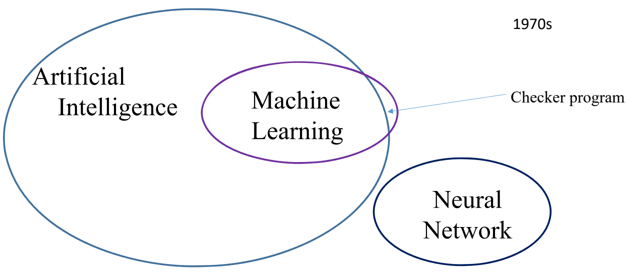
https://www.qubole.com/blog/deep-learning-the-latest-trend-in-ai-and-ml/

https://blogs.oracle.com/bigdata/difference-ai-machine-learning-deep-learning

AI, ML, and DL (DNN)

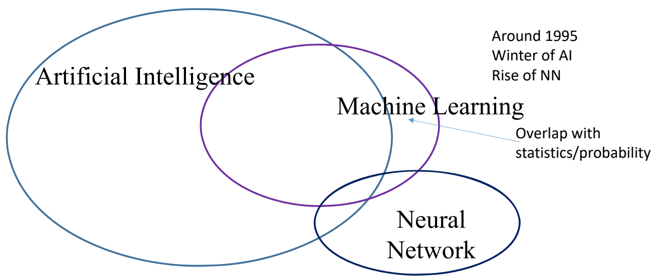
No. 2-1

1970s



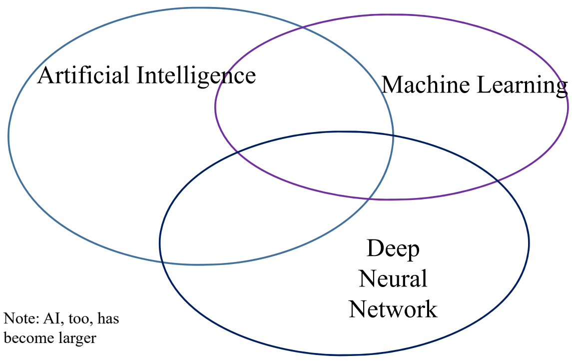
AI, ML, and DL (DNN)

No. 2-2



AI, ML, and DL (DNN)

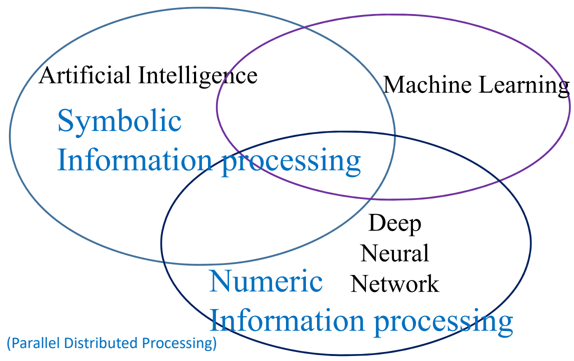
No. 2-3



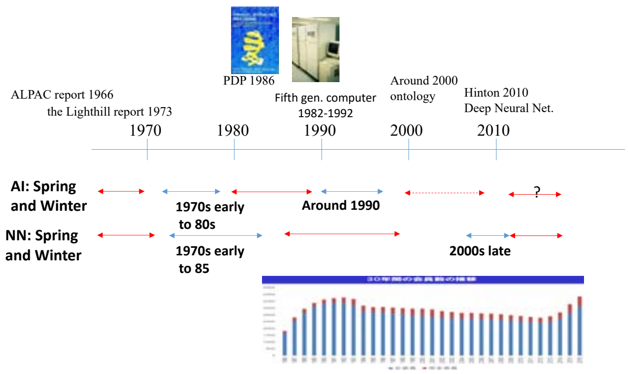
Note: AI, too, has become larger

From information viewpoint
AI, ML, and DL (DNN)

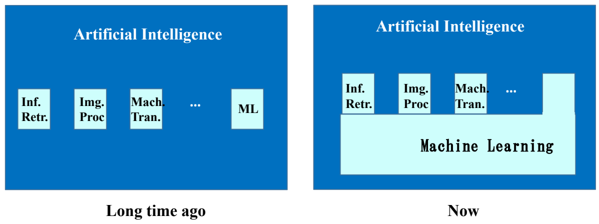
No. 2-3



Symbolic AI vs. Numerical AI, or AI vs. NN



Positioning of ML



Framework of ML

To write a learning program strong in Go,

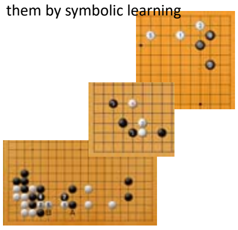
Rule-based (symbolic/discrete) :

A programmer should study Go so deeply that he/she could understand and distinguish good and bad moves.

Write programs to play, or collect Go scores as many as possible and learn them by symbolic learning

Non-symbolic (continuous value) :

A programmer should collect Go scores as many as possible, without need of knowing much of Go.



Framework of ML (cont.)

- Supervised learning
  - Given: Labeled data (pairs of input and output value (label))
  - Gives: estimated output for (possibly unseen) input
  - Problems: regression, classification, identification

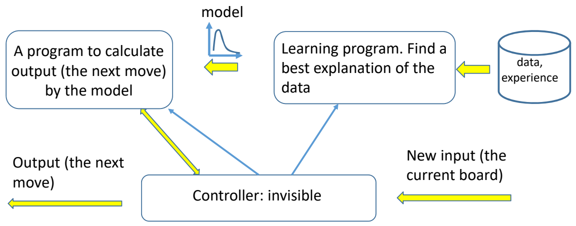
  (,1),(,4),... ⟹ (,),

- Unsupervised learning
  - Given: Unlabeled data (i.e., only "input")
  - Gives: Clustering, outlier detection

- Semi-supervised learning
  - Given: unlabeled and labeled data (the latter is in smaller number)
  - Gives: as supervised learning

- Reinforcement learning
  - Given: not given. Data should be collected by oneself
  - Gives: A best policy to reach a goal

Overview of a whole setup



Frameworks

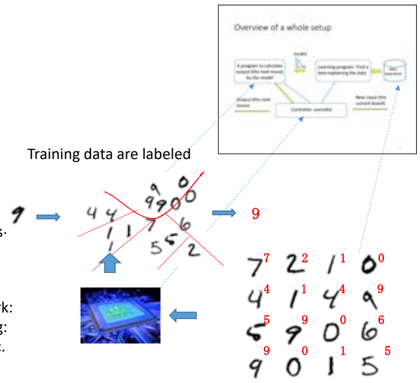
Supervised learning:

Unsupervised learning:

Semi-supervised learning.

Reinforcement learning:

Other learning framework:
representation learning:
correlation learning etc.



Frameworks

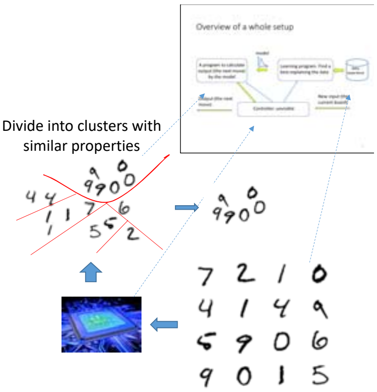
Supervised learning:

Unsupervised learning:

Semi-supervised learning:

Reinforcement learning:

Other learning framework:
representation learning:
correlation learning etc.



Frameworks

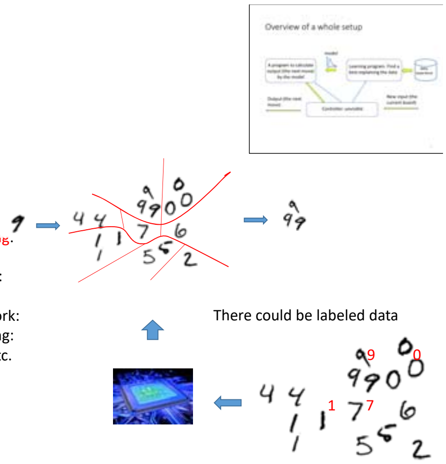
Supervised learning:

Unsupervised learning:

Semi-supervised learning.

Reinforcement learning:

Other learning framework:
representation learning:
correlation learning etc.



Frameworks

Supervised learning:

Unsupervised learning:

Semi-supervised learning:

Reinforcement learning:

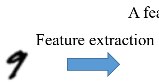
Other learning framework:
representation learning:
correlation learning etc.

Reinforcement learning
Reinforcement: any stimulus applied that will strengthen an organism's future behavior whenever that behavior is preceded by a specific antecedent stimulus

Reinforcement could be delayed (delay is quite common). Reinforcement given long after its cause is, too, considered. It could happen that the reword obtained long after the time of cause.
This means that actions that resulted in good reward is not necessarily correct.

Feature extraction

- To treat images, we need to form a vector form them. The resulted vector or its element is called feature vector or a feature.



$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

- Functions the map images to those feature vector vary from field of study to others.
- Deep learning often construct the feature function in its learning phase.

Formulation of ML problems with loss function.

- Supervised/Unsupervised learning is formulated as to minimize loss incurred by using the learned model.
  - If a parameter $\theta \in \Theta$ is to identify the model (hypothesis), Θ corresponds to a hypothesis space where we search for it; and the learning is to find a good approximation to the true one: $\hat{\theta} \approx \theta$.
  - A loss function measures badness of explaining occurrence of data z by $\hat{\theta}$: $\ell(z; \hat{\theta})$.
  - We need to evaluate $\hat{\theta}$ so that we want to eliminate z from ℓ.
- We consider two types of loss incurred by $\hat{\theta}$.
  - Generalization error (expected error): expected loss over all possible data sampled from the population, i.e., $L_g(\hat{\theta}) = \int \ell(Z; \hat{\theta}) p(Z) dZ$.
  - Empirical error (training error):$L_e(\hat{\theta}) = (1/n) \sum_{i=1}^{n} \ell(z_i; \hat{\theta})$

Generalization error vs. empirical error

- Minimization of generalization error is the goal of learning. But it is impossible because the population is not known.
- What we could do most is to use empirical error. But it is in general under-estimation and over-training/over-learning occurs:
  $\exists \theta_1 \theta_2 \, L_g(\theta_1) < L_g(\theta_2) \, \wedge \, L_e(\theta_1) > L_e(\theta_2)$
- To circumvent it, we use information criteria, cross-validation, regularization, and so forth.

Ex. Loss function for regression

For a sample $z = (x, y)$ and $f = \lambda . xf$

- Square loss: $\ell(z;\, \theta) = (1/2)(y - f(x))2$
- Absolute loss: $\ell(z;\, \theta) = |y - f(x)\,|$
- τ-Quantile loss:
  $\ell(z;\, \theta) = (1 - \tau) \max(f(x) - y, 0) + \tau \max(y - f(x), 0)$
- ε-incensitive loss:
  $\ell(z;\, \theta) = \max(|f(x) - y| - \varepsilon|, 0)$

Ex. Loss function (classification)

- For $y \in \{+1. -1\}$
- Logistic loss: $\ell(z;\, \theta) = \log((1 + \exp(-yf(x)/2)$.
- Hinge loss: $\ell(z;\, \theta) = \max\{\, 1 - yf(x), 0\}$.
- Exponential loss: $\ell(z;\, \theta) = \exp(-yf(x))$.

Over-training/over-learning

$\exists \theta_1 \theta_2 \, L_g(\theta_1) < L_g(\theta_2) \, \wedge \, L_e(\theta_1) > L_e(\theta_2)$

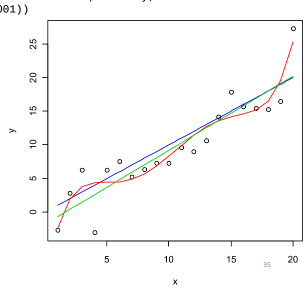
Hypothesis space:
a set of polynomials up to fifth degree
Loss function: Square loss
Parameters: degree, coefficients

```
library(nls2)
set.seed(1234)
x <- 1:20
y <- x+rnorm(20,sd=3)
plot(x,y)
xy <- data.frame(x=x,y=y)
res5 <- nls(y ~ a + b *x + c*x^2 + d *x^3 + e *x^4 + f*x^5, data=xy,
  start=list(a=1,b=1,c=0.5,d=0.1,e=0.05,f=0.001))
curve((x),col=4,add=T)           # blue
lines(x,predict(res5),col=2)     # red

res1 <- nls(y ~ a + b *x , data=xy,
                      start=list(a=1,b=1))
lines(x,predict(res1),col=3)     # green
```

Errors/residuals

```
> # in-sample/empirical error
> mean( (y-predict(res5) )^2)
[1] 5.808777
> mean( (y-predict(res1) )^2)
[1] 8.454419
> # generalization error
> mean( (x-predict(res5) )^2)
[1] 3.544463
> mean( (x-predict(res1) )^2)
[1] 0.8988136
```



Linear Regression

- Suppose that
  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$
- Simple linear regression
  $y = \beta_0 + \beta_1 x_1 + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$
- Multiple linear regression
  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$
  where $d \geq 2$

Ordinary Least Squares

- Suppose: observed samples are $(\mathbf{x}_i, y_i) \in R^d \times R$

$$\mathrm{X} = \begin{bmatrix} 1 & \mathbf{x}_1^\mathrm{T} \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^\mathrm{T} \end{bmatrix} \in R^{n\times(d+1)}, \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in R^n, \epsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \in R^n$$

- Suppose also that β^* is the true coefficients, i.e., $\mathbf{y} = \mathrm{X}\beta^* + \epsilon$

- Then OLS estimator is:

$$\begin{aligned} \hat{\beta} &= \arg\min_{\beta} \textstyle\sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\mathrm{T}\beta\right)^2 \\ &= \arg\min_{\beta} \|\mathbf{y} - \mathrm{X}\beta\|^2 \\ &= \left(\mathrm{X}^\mathrm{T}\mathrm{X}\right)^{-1}\mathrm{X}^\mathrm{T}\mathbf{y} \end{aligned}$$

Simple Examples of Linear Regression

- Simple Regression 1
  - Upload empty_R.ipynb to Google MyDrive and open it with CoLab
  - Copy & paste: install.packages('nls2')
  - Copy & paste the following and run them all.

```
library(nls2)
set.seed(1234)
x <- 1:20
y <- x+rnorm(20,sd=3)
plot(x,y)
xy <- data.frame(x=x,y=y)
res5 <- nls(y ~ a + b *x + c*x^2 + d *x^3 + e *x^4 + f*x^5, data=xy,
  start=list(a=1,b=1,c=0.5,d=0.1,e=0.05,f=0.001))
curve((x),col=4,add=T)            # blue
lines(x,predict(res5),col=2)      # red
res1 <- nls(y ~ a + b *x , data=xy, start=list(a=1,b=1))
lines(x,predict(res1),col=3)      # green
```

- Simple Regression 2
  - Do the same to: 01LinearRegression.R.ipynb
  - Explanation is in https://predictivemodeler.com/2019/02/23/r-basic-regression/

Summary

- Lecture
  - Basics of machine learning and deep learning
  - Exercise environment: Colab and others
  - Language: R and others
- Introduction to ML
  - Positioning of ML in AI
  - Symbolic AI and Numerical AI (not popular)
  - Supervised/unsupervised/semi-supervised/reinforcement
- Schedule
  - No class on Oct 17