Introduction to ML & DL

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning

Outline

1. What's Machine Learning?
2. What's Deep Learning?
3. About this Course...
4. FAQ

Outline

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
  - E.g., to tell if an email is spam or not
  - The correct answer varies in time and from person to person

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
  - E.g., to tell if an email is spam or not
  - The correct answer varies in time and from person to person
- Machine learning algorithms use the ***a posteriori knowledge*** to solve problems

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
  - E.g., to tell if an email is spam or not
  - The correct answer varies in time and from person to person
- Machine learning algorithms use the ***a posteriori knowledge*** to solve problems
  - Learnt from ***examples*** (as extra input)

Example Data $\mathbb{X}$ as Extra Input

- Unsupervised:

$$\mathbb{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}, \text{ where } \boldsymbol{x}^{(i)} \in \mathbb{R}^D$$

  - E.g., $\boldsymbol{x}^{(i)}$ an email

Example Data $\mathbb{X}$ as Extra Input

- Unsupervised:
$$\mathbb{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}, \text{ where } \boldsymbol{x}^{(i)} \in \mathbb{R}^D$$
  - E.g., $\boldsymbol{x}^{(i)}$ an email
- Supervised:
$$\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}, \text{ where } \boldsymbol{x}^{(i)} \in \mathbb{R}^D \text{ and } \boldsymbol{y}^{(i)} \in \mathbb{R}^K,$$
  - E.g., $y^{(i)} \in \{0, 1\}$ a spam label

General Types of Learning (1/2)

- ***Supervised learning***: learn to predict the labels of future data points

$X \in \mathbb{R}^{N \times D}$: 

$\boldsymbol{y} \in \mathbb{R}^{N \times K}$: $[\boldsymbol{e}^{(6)}, \boldsymbol{e}^{(1)}, \boldsymbol{e}^{(9)}, \boldsymbol{e}^{(4)}, \boldsymbol{e}^{(2)}]$

$\boldsymbol{x}' \in \mathbb{R}^{D}$: 

$\boldsymbol{y}' \in \mathbb{R}^{K}$: ?

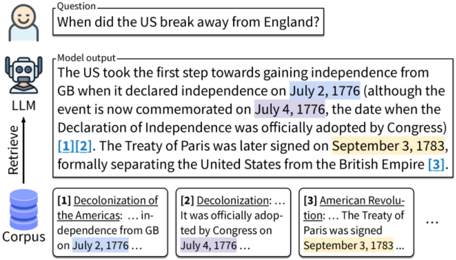
General Types of Learning (1/2)

- ***Supervised learning***: learn to predict the labels of future data points

$\boldsymbol{X} \in \mathbb{R}^{N \times D}$:  $\boldsymbol{x}' \in \mathbb{R}^{D}$: 

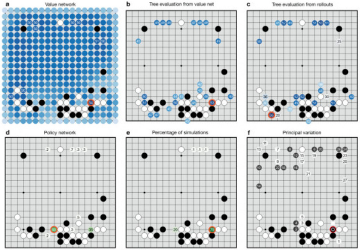
$\boldsymbol{y} \in \mathbb{R}^{N \times K}$: $[\boldsymbol{e}^{(6)}, \boldsymbol{e}^{(1)}, \boldsymbol{e}^{(9)}, \boldsymbol{e}^{(4)}, \boldsymbol{e}^{(2)}]$ $\boldsymbol{y}' \in \mathbb{R}^{K}$: ?

- ***Unsupervised learning***: learn (latent) patterns in $\boldsymbol{X}$, and optionally generate new $\boldsymbol{x}$'s



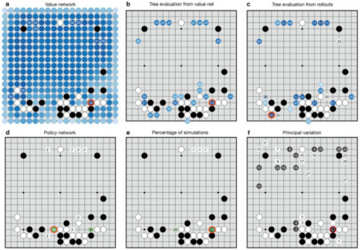
General Types of Learning (2/2)

- ***Reinforcement learning***: learn from "good"/"bad" feedback of actions (instead of correct labels) to maximize the goal



General Types of Learning (2/2)

- ***Reinforcement learning***: learn from "good"/"bad" feedback of actions (instead of correct labels) to maximize the goal



- AlphaGo [1] is a hybrid of reinforcement learning and supervised learning
  - Supervised learning from the game records
  - Then, reinforcement learning from self-play

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
    1. Split a dataset into the training and testing datasets

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
   1. Split a dataset into the training and testing datasets
2. Model development
   1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}$ that is a collection of candidate functions f's (representing posteriori knowledge) we want to discover
      1. f is assumed to be parametrized by $\boldsymbol{w}$

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
    1. Split a dataset into the training and testing datasets
2. Model development
    1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}$ that is a collection of candidate functions f's (representing posteriori knowledge) we want to discover
        1. f is assumed to be parametrized by $\boldsymbol{w}$
    2. Define a ***cost function*** $C(\boldsymbol{w};\mathbb{X})$ (or functional $C[f;\mathbb{X}]$) that measures "how good a particular f can explain the training data"

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
    1. Split a dataset into the training and testing datasets
2. Model development
    1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}$ that is a collection of candidate functions f's (representing posteriori knowledge) we want to discover
        1. f is assumed to be parametrized by $\boldsymbol{w}$
    2. Define a ***cost function*** $C(\boldsymbol{w};\mathbb{X})$ (or functional $C[f;\mathbb{X}]$) that measures "how good a particular f can explain the training data"
3. ***Training***: employ an algorithm that finds the best (or good enough) function $f^*(\cdot;\boldsymbol{w}^*)$ in the model that minimizes the cost function

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X})$$

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
    1. Split a dataset into the training and testing datasets
2. Model development
    1. Assume a ***model*** $\{f(\cdot\,;\boldsymbol{w})\}$ that is a collection of candidate functions f's (representing posteriori knowledge) we want to discover
        1. f is assumed to be parametrized by $\boldsymbol{w}$
    2. Define a ***cost function*** $C(\boldsymbol{w};\mathbb{X})$ (or functional $C[f;\mathbb{X}]$) that measures "how good a particular f can explain the training data"
3. ***Training***: employ an algorithm that finds the best (or good enough) function $f^*(\cdot\,;\boldsymbol{w}^*)$ in the model that minimizes the cost function

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X})$$

4. ***Testing***: evaluate the performance of the learned f^* using the testing dataset

General Machine Learning Steps

1. Data collection, preprocessing (e.g., integration, cleaning, etc.), and exploration
   1. Split a dataset into the training and testing datasets
2. Model development
   1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}$ that is a collection of candidate functions f's (representing posteriori knowledge) we want to discover
      1. f is assumed to be parametrized by $\boldsymbol{w}$
   2. Define a ***cost function*** $C(\boldsymbol{w};\mathbb{X})$ (or functional $C[f;\mathbb{X}]$) that measures "how good a particular f can explain the training data"
3. ***Training***: employ an algorithm that finds the best (or good enough) function $f^*(\cdot;\boldsymbol{w}^*)$ in the model that minimizes the cost function

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X})$$

4. ***Testing***: evaluate the performance of the learned f^* using the testing dataset
5. Apply the model in the real world

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
   1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
   1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$
   2. ***Cost function***: $C(\boldsymbol{w};\mathbb{X}) = \Sigma_i 1(\boldsymbol{w};f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
   1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$
   2. ***Cost function***: $C(\boldsymbol{w};\mathbb{X}) = \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
3. ***Training***: to solve $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
   1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$
   2. ***Cost function***: $C(\boldsymbol{w};\mathbb{X}) = \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
3. ***Training***: to solve $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
4. ***Testing***: accuracy $\frac{1}{|\mathbb{X}'|}\Sigma_i 1(\boldsymbol{x}'^{(i)}, y'^{(i)}; f(\boldsymbol{x}'^{(i)};\boldsymbol{w}^*) = y'^{(i)})$

Example for Spam Detection

1. Random split of your past emails and labels
    1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
    2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
    1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$
    2. ***Cost function***: $C(\boldsymbol{w};\mathbb{X}) = \Sigma_i 1(\boldsymbol{w};f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
3. ***Training***: to solve $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \Sigma_i 1(\boldsymbol{w};f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
4. ***Testing***: accuracy $\frac{1}{|\mathbb{X}'|}\Sigma_i 1(\boldsymbol{x}'^{(i)}, y'^{(i)};f(\boldsymbol{x}'^{(i)};\boldsymbol{w}^*) = y'^{(i)})$
5. Use f^* to predict the labels of your future emails

Example for Spam Detection

1. Random split of your past emails and labels
   1. Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
   2. Testing dataset: $\mathbb{X}' = \{(\boldsymbol{x}'^{(i)}, y'^{(i)})\}_i$
2. Model development
   1. ***Model***: $\{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}\}$
   2. ***Cost function***: $C(\boldsymbol{w};\mathbb{X}) = \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
3. ***Training***: to solve $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \Sigma_i 1(\boldsymbol{w}; f(\boldsymbol{x}^{(i)};\boldsymbol{w}) \neq y^{(i)})$
4. ***Testing***: accuracy $\frac{1}{|\mathbb{X}'|}\Sigma_i 1(\boldsymbol{x}'^{(i)}, y'^{(i)}; f(\boldsymbol{x}'^{(i)};\boldsymbol{w}^*) = y'^{(i)})$
5. Use f^* to predict the labels of your future emails

- See Notation

Outline

Deep Learning

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \to \boxed{f^{(1)}\left(\cdot\,;\mathbf{w}^{(1)}\right)} \to \boxed{f^{(2)}\left(\cdot\,;\mathbf{w}^{(2)}\right)} \to \cdots \ \boxed{f^{(L)}\left(\cdot\,;\mathbf{w}^{(L)}\right)} \to \widehat{\mathbf{y}}$$

Deep Learning

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \longrightarrow \boxed{f^{(1)}\left(\cdot\,;\mathbf{w}^{(1)}\right)} \longrightarrow \boxed{f^{(2)}\left(\cdot\,;\mathbf{w}^{(2)}\right)} \longrightarrow \cdots \boxed{f^{(L)}\left(\cdot\,;\mathbf{w}^{(L)}\right)} \longrightarrow \hat{\mathbf{y}}$$

- Pros:
  - Learns to pre-process data automatically
  - Learns a complex function (e.g., visual objects to labels)

Deep Learning

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \longrightarrow \boxed{f^{(1)}\left(\cdot\,;\mathbf{w}^{(1)}\right)} \longrightarrow \boxed{f^{(2)}\left(\cdot\,;\mathbf{w}^{(2)}\right)} \longrightarrow \cdots \boxed{f^{(L)}\left(\cdot\,;\mathbf{w}^{(L)}\right)} \longrightarrow \hat{\mathbf{y}}$$

- Pros:
  - Learns to pre-process data automatically
  - Learns a complex function (e.g., visual objects to labels)
- Cons:
  - Usually needs large data to train a model well
  - Higher computation costs (for both training and testing)

Outline

Target Audience

- ***Senior undergraduate*** and ***graduate*** CS students
  - Easy-to-moderate level of theory
  - Coding and engineering (in Python)
  - Clean datasets (small & large)

Target Audience

- ***Senior undergraduate*** and ***graduate*** CS students
  - Easy-to-moderate level of theory
  - Coding and engineering (in Python)
  - Clean datasets (small & large)
- No prior knowledge about ML is needed

Topics Covered

- Supervised, unsupervised learning, and reinforcement learning

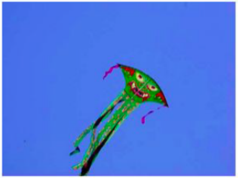
Topics Covered

- Supervised, unsupervised learning, and reinforcement learning
- with ***structural*** output:

A man holding a tennis racquet on a tennis court.

Two pizzas sitting on top of a stove top oven

A group of young people playing a game of Frisbee



A man flying through the air while riding a snowboard

Syllabus (Tentative)

- Part 1: math review (2 weeks)
    - Linear algebra
    - Probability & information theory
    - Numerical optimization

Syllabus (Tentative)

- Part 1: math review (2 weeks)
  - Linear algebra
  - Probability & information theory
  - Numerical optimization
- Part 2: machine learning basics (3 weeks)
  - Learning theory
  - Parametric/non-parametric models
  - Experiment design

Syllabus (Tentative)

- Part 1: math review (2 weeks)
  - Linear algebra
  - Probability & information theory
  - Numerical optimization
- Part 2: machine learning basics (3 weeks)
  - Learning theory
  - Parametric/non-parametric models
  - Experiment design
- Part 3: deep supervised learning (6 weeks)
  - Neural Networks (NNs), CNNs, RNNs

Syllabus (Tentative)

- Part 1: math review (2 weeks)
  - Linear algebra
  - Probability & information theory
  - Numerical optimization
- Part 2: machine learning basics (3 weeks)
  - Learning theory
  - Parametric/non-parametric models
  - Experiment design
- Part 3: deep supervised learning (6 weeks)
  - Neural Networks (NNs), CNNs, RNNs
- Part 4: unsupervised learning (2 weeks)
  - Autoencoders, manifold learning, GANs

Syllabus (Tentative)

- Part 1: math review (2 weeks)
  - Linear algebra
  - Probability & information theory
  - Numerical optimization
- Part 2: machine learning basics (3 weeks)
  - Learning theory
  - Parametric/non-parametric models
  - Experiment design
- Part 3: deep supervised learning (6 weeks)
  - Neural Networks (NNs), CNNs, RNNs
- Part 4: unsupervised learning (2 weeks)
  - Autoencoders, manifold learning, GANs
- Part 5: reinforcement learning (3 weeks)
  - Value/gradient policies, action/critics, reinforce RNNs

Grading (Tentative)

- Prerequisite quiz: ***15%***
    - ***On next Thu (9/21)***
    - ***You have to pass to be able to take this course: >70 or within top-70***
- Contests (x 4): ***40%***
    - At the end of each part
- Assignments: ***20%***
    - Come with the labs
- Final exam: ***25%***
- Bonus: ***6%***
    - Math labs (x 4)
    - Optional ML topics (x 2)

Classes Info

- Lectures on Tue (2 hours)
  - Concepts & theories
  - with companion videos
- Labs on Thu (1 hour)
  - Implementation (in Python) & engineering topics
- TA time: 4:20pm–5:30pm on Thu at Delta 729
- More info can be found in the course website

Outline

FAQ (1/2)

Q: *Should we team up for the contests?*

A: Yes, 2~4 students per team

FAQ (1/2)

Q: *Should we team up for the contests?*

A: Yes, 2~4 students per team

Q: *Which GPU card should I buy?*

A: Nvidia GTX 1060 or above; 1050 Ti (4G RAM) minimal

FAQ (1/2)

Q: *Should we team up for the contests?*

A: Yes, 2~4 students per team

Q: *Which GPU card should I buy?*

A: Nvidia GTX 1060 or above; 1050 Ti (4G RAM) minimal

Q: *Do we need to attend the classes?*

A: No, as long as you can pass. But you have attendance bonus...

FAQ (1/2)

Q: *Should we team up for the contests?*

A: Yes, 2~4 students per team

Q: *Which GPU card should I buy?*

A: Nvidia GTX 1060 or above; 1050 Ti (4G RAM) minimal

Q: *Do we need to attend the classes?*

A: No, as long as you can pass. But you have attendance bonus...

Q: *Is this a light-loading course or heavy-loading one?*

A: Should be ***very heavy*** to most students. Please ***reserve your time***

FAQ (2/2)

Q: *What's the textbook?*

A: No formal textbook. But if you need one, read the Deep Learning book

FAQ (2/2)

Q: *What's the textbook?*

A: No formal textbook. But if you need one, read the Deep Learning book

Q: *Why some sections are marked with "*" or "**" in the slides?*

A: The mark "*" means "can be skipped for the first time reader," and ""**" means "materials for reference only"

TODO

- Assigned reading:
  - Calculus
  - Get your feet wet with Python

Reference I

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al.
Mastering the game of go with deep neural networks and tree search.
Nature, 529(7587):484–489, 2016.