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

1. What’s Machine Learning?
2. What’s Deep Learning?
3. About this Course...
4. FAQ
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 $X$ as Extra Input

- Unsupervised:
  \[ X = \{x^{(i)}\}_{i=1}^{N}, \text{ where } x^{(i)} \in \mathbb{R}^D \]
  - E.g., $x^{(i)}$ an email
Example Data $\mathbf{X}$ as Extra Input

- **Unsupervised:**
  \[
  \mathbf{X} = \{ \mathbf{x}^{(i)} \}_{i=1}^{N}, \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^D
  \]
  - E.g., $\mathbf{x}^{(i)}$ an email

- **Supervised:**
  \[
  \mathbf{X} = \{ (\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \}_{i=1}^{N}, \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^D \text{ and } \mathbf{y}^{(i)} \in \mathbb{R}^K
  \]
  - E.g., $\mathbf{y}^{(i)} \in \{0, 1\}$ a spam label
General Types of Learning (1/2)

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

\[
X \in \mathbb{R}^{N \times D} : \begin{bmatrix} 6 & 1 & 9 & 4 & 2 \end{bmatrix} \quad x' \in \mathbb{R}^D : \begin{bmatrix} \end{bmatrix}
\]

\[
y \in \mathbb{R}^{N \times K} : \begin{bmatrix} e^{(6)}, e^{(1)}, e^{(9)}, e^{(4)}, e^{(2)} \end{bmatrix} \quad y' \in \mathbb{R}^K : \begin{bmatrix} ? \end{bmatrix}
\]
General Types of Learning (1/2)

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

\[
X \in \mathbb{R}^{N \times D} : \begin{bmatrix} 6 & 1 & 9 & 4 & 2 \end{bmatrix} \quad x' \in \mathbb{R}^D : \begin{bmatrix} \star \end{bmatrix} \\
y \in \mathbb{R}^{N \times K} : [e^{(6)}, e^{(1)}, e^{(9)}, e^{(4)}, e^{(2)}] \quad y' \in \mathbb{R}^K : \begin{bmatrix} ? \end{bmatrix}
\]

- **Unsupervised learning**: learn patterns or latent factors in \( X \)
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. First, supervised learning from the game records, then, reinforcement learning from self-play.
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
   - Split a dataset into the training and testing datasets

2. Model development
   - Assume a \textit{model} \{f(\cdot;w)\} that is a collection of candidate functions \(f\)'s (representing posteriori knowledge) we want to discover
   - \(f\) is assumed to be parametrized by \(w\)

---

Shan-Hung Wu (CS, NTHU)
General Machine Learning Steps

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

2. Model development
   - Assume a model \( \{f(\cdot; w)\} \) that is a collection of candidate functions \( f \)'s (representing posteriori knowledge) we want to discover
     - \( f \) is assumed to be parametrized by \( w \)
   - Define a cost function \( C(w; X) \) (or functional \( C[f; X] \)) 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
   - Split a dataset into the training and testing datasets

2. Model development
   - Assume a model \( \{ f(\cdot; w) \} \) that is a collection of candidate functions \( f \)'s (representing posteriori knowledge) we want to discover
     - \( f \) is assumed to be parametrized by \( w \)
   - Define a cost function \( C(w; X) \) (or functional \( C[f; X] \)) 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 \( f^*(\cdot; w^*) \) in the model that minimizes the cost function
   \[
   w^* = \arg \min_w C(w; X)
   \]
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 \( \{f(\cdot; w)\} \) that is a collection of candidate functions \( f \)'s (representing posteriori knowledge) we want to discover
   2. \( f \) is assumed to be parametrized by \( w \)
   3. Define a cost function \( C(w; X) \) (or functional \( C[f; X] \)) 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 \( f^*(\cdot; w^*) \) in the model that minimizes the cost function

   \[
   w^* = \arg \min_w C(w; X)
   \]

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
   - Split a dataset into the training and testing datasets

2. Model development
   - Assume a model \{f(\cdot; w)\} that is a collection of candidate functions \(f\)'s (representing posteriori knowledge) we want to discover
     - \(f\) is assumed to be parametrized by \(w\)
   - Define a cost function \(C(w; X)\) (or functional \(C[f; X]\)) 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 \(f^*(\cdot; w^*)\) in the model that minimizes the cost function

\[
    w^* = \arg \min_w C(w; X)
\]

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

5. Apply the model in the real world
Random split of your past emails and labels

1. Training dataset: $X = \{(x^{(i)}, y^{(i)})\}_i$
2. Testing dataset: $X' = \{(x'^{(i)}, y'^{(i)})\}_i$
Example for Spam Detection

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

2. Model development
   1. Model: \( \{ f : f(x; w) = w^\top x \} \)
Example for Spam Detection

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

2. Model development
   1. **Model**: $\{f : f(x; w) = w^\top x\}$
   2. **Cost function**: $C(w; X) = \Sigma_i 1(w; f(x^{(i)}; w) \neq y^{(i)})$
Example for Spam Detection

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

2. Model development
   1. **Model**: \( \{f : f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}\} \)
   2. **Cost function**: \( C(\mathbf{w}; \mathbf{X}) = \sum_i 1(\mathbf{w}; f(\mathbf{x}^{(i)}; \mathbf{w}) \neq y^{(i)}) \)

3. **Training**: to solve \( \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i 1(\mathbf{w}; f(\mathbf{x}^{(i)}; \mathbf{w}) \neq y^{(i)}) \)
Example for Spam Detection

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

2. Model development
   1. **Model**: \( \{ f : f(x; w) = w^\top x \} \)
   2. **Cost function**: \( C(w; X) = \sum_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

3. **Training**: to solve \( w^* = \arg \min_w \sum_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

4. **Testing**: accuracy \( \frac{1}{|X'|} \sum_i 1(x'^{(i)}, y'^{(i)}; f(x'^{(i)}; w^*) = y'^{(i)}) \)

See Notation
Example for Spam Detection

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

2. Model development
   1. **Model**: \( \{f : f(x; w) = w^\top x\} \)
   2. **Cost function**: \( C(w; X) = \sum_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

3. **Training**: to solve \( w^* = \arg \min_w \sum_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

4. **Testing**: accuracy \( \frac{1}{|X'|} \sum_i 1(x'^{(i)}, y'^{(i)}; f(x'^{(i)}; w^*) = y'^{(i)}) \)

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: \( X = \{(x^{(i)}, y^{(i)})\}_i \)
   2. Testing dataset: \( X' = \{(x'^{(i)}, y'^{(i)})\}_i \)

2. Model development
   1. **Model**: \( \{f : f(x; w) = w^\top x\} \)
   2. **Cost function**: \( C(w; X) = \Sigma_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

3. **Training**: to solve \( w^* = \arg\min_w \Sigma_i 1(w; f(x^{(i)}; w) \neq y^{(i)}) \)

4. **Testing**: accuracy \( \frac{1}{|X'|} \Sigma_i 1(x'^{(i)}, y'^{(i)}; f(x'^{(i)}; w^*) = y'^{(i)}) \)

5. Use \( f^* \) to predict the labels of your future emails
   - See Notation
Outline

1 What’s Machine Learning?

2 What’s Deep Learning?

3 About this Course...

4 FAQ
Deep Learning

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

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

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
- High computation costs (at both training and test time)
Deep Learning

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

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

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

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

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

- 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
  - High computation costs (at both training and test time)
Outline

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

- 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
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

Shan-Hung Wu (CS, NTHU)
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)

- Math quiz: 20%
  - *In the next week*
  - *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: 40%
  - Come with the labs

- 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: 1:20pm - 3:10pm on Mon at Delta 723
- More info can be found in the course website
Outline

1. What’s Machine Learning?
2. What’s Deep Learning?
3. About this Course...
4. FAQ
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
**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: 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
Q: What’s the textbook?
A: No formal textbook. But if you need one, read the Deep Learning book
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”
Assigned reading:

- Calculus
- Get your feet wet with Python
Mastering the game of go with deep neural networks and tree search. 