CS260: Machine Learning Algorithms
Lecture 1: Overview

Cho-Jui Hsieh
UCLA

Jan 7, 2019
Course Information

- My office: EVI 284
- Office hours: Wednesday 11am–noon
- Online office hour: TBD
- TA: Patrick Chen (patrickchen@g.ucla.edu)
- TA for online course: Minhao Cheng (mhcheng@ucla.edu)
Course Information

- There is no textbook. Most of the topics are covered in “Deep Learning” (by Goodfellow, Bengio, Courville)
Course Information

- There is no textbook. Most of the topics are covered in “Deep Learning” (by Goodfellow, Bengio, Courville)

- Part I (basic concepts):
  - Linear models (regression, classification, clustering, dimension reduction)
  - Basic learning theory (overfitting, regularization)

- Part II (Nonlinear models):
  - Kernel methods
  - Tree-based methods
  - Deep networks
  - Applications in computer vision and NLP
Grading Policy

- Midterm exam (30%)
- Homework (30%)
  - 3 homeworks
- Final project (40%)
Final project

- Group of \( \leq 4 \) students.
- Work on some research projects:
  - Solve an interesting problem
  - Develop a new algorithm
  - Compare state-of-the-art algorithms on some problems
  - …
- I’ll recommend some topics in the course. Feel free to discuss with me in advance.
Machine Learning: Overview
From learning to machine learning

- What is learning?

  observations → Learning → **Skill**

- **Skill**: how to make decision (action)
  - Classify an image
  - Translate a sentence from one language to another
  - …
From learning to machine learning

- What is learning?

  observations → Learning → Skill

- Skill: how to make decision (action)
  - Classify an image
  - Translate a sentence from one language to another
  - ...

- Machine learning:

  data → Machine Learning → Skill (decision rules)

Automatic the learning process!
Credit Approval Problem

- Customer record (features):
  
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

- To be learned:
  
  “Should we approve the credit card application?”
Credit Approval Problem

- Customer record (features):

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

- To be learned:

  “Should we approve the credit card application?”

- Data: A collection of feature-label pairs:

  (customer1 feature, Yes), (customer2 feature, No), …
Credit Approval Problem

- Customer record (features):
  
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

- To be learned:
  
  "Should we approve the credit card application?"

- Data: A collection of feature-label pairs:
  
  (customer1 feature, Yes), (customer2 feature, No), …

- Learned model: Some decision rule
  
  e.g., salary > 1M
Formalize the Learning Problem

- **Input:** $x \in \mathcal{X}$ (customer application)
  
  e.g., $x = [23, 1, 1000000, 1, 0.5, 200000]$

- **Output:** $y \in \mathcal{Y}$ (approve/disapprove)

- **Target function to be learned:**
  
  $f : \mathcal{X} \to \mathcal{Y}$ (ideal credit approval formula)

- **Data (historical records in bank):**
  
  $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$

- **Hypothesis (model):**
  
  $g : \mathcal{X} \to \mathcal{Y}$ (learned formula to be used)
Basic Setup of Learning Problem

(Figure from “Learning from Data”)

UNKNOWN TARGET FUNCTION
\( f: X \rightarrow Y \)

(ideal credit approval function)

TRAINING EXAMPLES
\((x_1, y_1), \ldots, (x_N, y_N)\)

(historical records of credit customers)

LEARNING ALGORITHM
\( A \)

HYPOTHESIS SET
\( \mathcal{H} \)

(set of candidate formulas)

FINAL HYPOTHESIS
\( g \approx f \)

(final credit approval formula)
A learning model has two components:

- The hypothesis set $\mathcal{H}$:
  - Set of candidate hypothesis (functions)
- The learning algorithm:
  - To pick a hypothesis (function) from the $\mathcal{H}$
  - Usually optimization algorithm (choose the best function to minimize the training error)
Perceptron

- Our first ML model: perceptron (1957)
  - Learning a linear function
  - Single layer neural network
- Next, we introduce two components of perceptron:
  - What’s the hypothesis space?
  - What’s the learning algorithm?
Define the hypothesis set $\mathcal{H}$

- For input $x = (x_1, \ldots, x_d)$ “attributes of a customer”

  Approve credit if $\sum_{i=1}^{d} w_i x_i > \text{threshold}$,

  Deny credit if $\sum_{i=1}^{d} w_i x_i < \text{threshold}$

- Define $\mathcal{Y} = \{+1(\text{good}), -1(\text{bad})\}$

- Linear hypothesis space $\mathcal{H}$: all the $h$ with the following form

  $$h(x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \text{threshold}\right)$$

  (perceptron hypothesis)
Introduce an artificial coordinate \( x_0 = -1 \) and set \( w_0 = \text{threshold} \)

\[
h(x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \text{threshold}\right) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) = \text{sign}(w^T x)
\]

(vector form)

Customer features \( x \): points on \( \mathbb{R}^d \) \( (d \text{ dimensional space}) \)

Labels \( y \): +1 or −1

Hypothesis \( h \): linear hyperplanes
Select the best one from $\mathcal{H}$

- $\mathcal{H}$: all possible linear hyperplanes
- How to select the best one?

$\mathcal{H}$: all possible linear hyperplanes

How to select the best one?
Select the best one from $\mathcal{H}$

- $\mathcal{H}$: all possible linear hyperplanes
- How to select the best one?

Find $g$ such that $g(x_n) \approx f(x_n) = y_n$ for $n = 1, \cdots, N$
Select the best one from $\mathcal{H}$

- $\mathcal{H}$: all possible linear hyperplanes
- How to select the best one?

Find $g$ such that $g(x_n) \approx f(x_n) = y_n$ for $n = 1, \cdots, N$

- Naive approach:
  Test all $h \in \mathcal{H}$ and choose the best one minimizing the "training error"

$$
\text{training error} = \frac{1}{N} \sum_{n=1}^{N} I(h(x_n) \neq y_n)
$$

($I(\cdot)$: indicator)

- Difficult: $\mathcal{H}$ is of infinite size
Perceptron Learning Algorithm

Initial from some $\mathbf{w}$ (e.g., $\mathbf{w} = \mathbf{0}$)
For $t = 1, 2, \ldots$
  
  - Find a **misclassified** point $n(t)$:
    
    $$\text{sign} (\mathbf{w}^T \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

  - Update the weight vector:
    
    $$\mathbf{w} \leftarrow \mathbf{w} + y_{n(t)} \mathbf{x}_{n(t)}$$

---

**Perceptron Learning Algorithm (PLA)**

Initial from some $\mathbf{w}$ (e.g., $\mathbf{w} = \mathbf{0}$)
For $t = 1, 2, \ldots$
  
  - Find a **misclassified** point $n(t)$:
    
    $$\text{sign} (\mathbf{w}^T \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

  - Update the weight vector:
    
    $$\mathbf{w} \leftarrow \mathbf{w} + y_{n(t)} \mathbf{x}_{n(t)}$$
Iteratively

- Find a misclassified point
- Rotate the hyperplane according to the misclassified point
Perceptron Learning Algorithm

- Converge for “linearly separable” case:
  - Linearly separable: there exists a perceptron (linear) hypothesis $f$ with 0 training error
  - PLA is guaranteed to obtain $f$
    (Stop when no more misclassified point)

![Diagram](image-url)

(linear separable) (not linear separable) (not linear separable)
Binary classification

- **Data:**
  - Features for each training example: \( \{x_n\}_{n=1}^{N} \), each \( x_n \in \mathbb{R}^d \)
  - Labels for each training example: \( y_n \in \{+1, -1\} \)

- **Goal:** Learn a function \( f : \mathbb{R}^d \rightarrow \{+1, -1\} \)

- **Examples:**
  - Credit approve/disapprove
  - Email spam/not-scam
  - Patient sick/not sick
  - ...
Other types of output space - Regression

Regression: \( y_n \in \mathbb{R} \) (output is a real number)

Example:

- Stock price prediction
- Movie rating prediction
- ...
Multi-class classification:

- \( y_n \in \{1, \cdots, C\} \) (C-way classification)
- Example: Coin recognition
  - Classify coins by two features (size, mass) \((x_n \in \mathbb{R}^2)\)
  - \( y_n \in \mathcal{Y} = \{1c, 5c, 10c, 25c\} \)
    \((\mathcal{Y} = \{1, 2, 3, 4\})\)
- Other examples: hand-written digits, \cdots
Other types of output space - Multi-label prediction

- Multi-class problem: Each sample only has one label
- Multi-label problem: Each sample can have multiple labels

Examples:
- Document categorization (news/sports/economy/···)
- Document/image tagging
- Extreme classification (large output space problems): Millions of billions of labels (but usually each sample only has few labels)
- Recommendation systems: Predict a subset of preferred items for each user
- Document retrieval or search: Predict a subset of related articles for a query
Multi-class problem: Each sample only has one label
Multi-label problem: Each sample can have multiple labels

Example:
- Document categorization (news/sports/economy/⋯)
- Document/image tagging
- ⋅⋅⋅
Multi-class problem: Each sample only has one label
Multi-label problem: Each sample can have multiple labels

Example:
- Document categorization (news/sports/economy/⋯)
- Document/image tagging
- ⋮

Extreme classification (large output space problems):
- Millions of billions of labels (but usually each sample only has few labels)
- Recommendation systems: Predict a subset of preferred items for each user
- Document retrieval or search: Predict a subset of related articles for a query
Other types of output space - structure predict

- Output as exponential

  ![Diagram of pronoun, verb, noun labels]

- Multiclass classification for each word (word ⇒ word class)
  (not using information of the whole sentence)

- Structure prediction problem:
  sentence ⇒ structure (class of each word)

- Other examples: speech recognition, image captioning, machine translation, ...
Machine Learning Problems

Machine learning problems can usually be categorized into

- **Supervised learning**: every $x_n$ comes with $y_n$ (label)
  
  (semi-supervised learning)

- **Unsupervised learning**: only $x_n$, no $y_n$

- **Reinforcement learning**:
  
  Examples contain (input, some output, grade for this output)
Unsupervised Learning (no $y_n$)

- Clustering: given examples $x_1, \ldots, x_N$, classify them into $K$ classes
- Other unsupervised learning:
  - Outlier detection: $\{x_n\} \Rightarrow \text{unusual}(x)$
  - Dimensional reduction
  - ...
Semi-supervised learning

- Only some (few) $x_n$ has $y_n$
- Labeled data is much more expensive than unlabeled data
Reinforcement Learning

- Used a lot in game AI, robotic controls
  - Agent observe state $S_t$
  - Agent conduct action $A_t$
    (ML model, based on input $S_t$)
  - Environment gives agent reward $R_t$
  - Environment gives agent next state $S_{t+1}$
- Only observe “grade” for a certain action (best action is not revealed)
- Ads system: (customer, ad choice, click or not)
Conclusions

- Basic concept of learning:
  - Set up a hypothesis space (potential functions)
  - Define an error measurement (define the quality of each function based on data)
  - Develop an algorithm to choose a good hypothesis based on the error measurement (optimization)
- A perceptron algorithm (linear classification)
- Binary classification, multiclass, multilabel, structural prediction
- Supervised vs unsupervised learning

Questions?