CSE 417T
Introduction to Machine Learning

Instructor: Chien-Ju (CJ) Ho
Logistics

• HW 0:
  • Due by **11am next Tuesday**
  • Submit via Gradescope
  • Only waitlisted students need to submit
  • No late days can be used
  • The rules on academic integrity apply

• HW 1: Will be announced next week
  • The question in HW0 will appear in HW1 as well
Logistics: Academic Integrity

• Discussion (conceptually) about course content and homework assignments is encouraged.

• How to make sure to not violate academic integrity?
• Rule of thumb:
  • You should write down the answers/codes entirely on your own.
  • Can’t look at the write-up / codes by others.

• Ask if you are not sure!
Recap
Given by the learning problem

Goal of learning

\[ f : \mathcal{X} \mapsto \mathcal{Y} \]

(ideal credit approval formula)

\[ y_n = f(x_n) \]

TRAINING EXAMPLES

\[(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\]

(historical records of credit customers)

LEARNING ALGORITHM

\[ \mathcal{A} \]

FINAL HYPOTHESIS

\[ g \approx f \]

(learned credit approval formula)

HYPOTHESIS SET

\[ \mathcal{H} \]

(set of candidate formulas)
Linear hypothesis space (Perceptron)

- Input $\vec{x} = (x_1, x_2, ..., x_d)$
- Output $y \in \{-1, +1\}$

- A hypothesis $h$ is a linear separator $\vec{w}^T \vec{x} = b$, characterized by
  - weight vector $\vec{w} = (w_1, ..., w_d)$
  - threshold $b$

- $h(\vec{x}) = sign(\sum_{i=1}^{d} w_i x_i - b) = sign(\vec{w}^T \vec{x} - b)$
  - Predict $+1$ if $\vec{w}^T \vec{x} > b$
  - Predict $-1$ if $\vec{w}^T \vec{x} < b$
Linear hypothesis space (Perceptron)

- To simplify $h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x} - b)$, define
  - $x_0 = 1$
  - $w_0 = -b$

- And we implicitly let
  - $\vec{x} = (x_0, x_1, ..., x_d)$
  - $\vec{w} = (w_0, w_1, ..., w_d)$

- A hypothesis can then be written as
  - $h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})$
  - We will interchangeably use $h$ and $\vec{w}$ to express a hypothesis in Perceptron
Perceptron Learning Algorithm (PLA)

- Given a dataset $D = \{(\vec{x}_1, y_1), ..., (\vec{x}_N, y_N)\}$
- Assume the dataset is **linearly separable**
- Want to find a hypothesis that separates data in $D$

Perceptron Learning Algorithm
- Initialize $\vec{w}(0) = \vec{0}$
- For $t = 0, ...$
  - Find a misclassified example $(\vec{x}(t), y(t))$ in $D$
    - That is, $\text{sign}(\vec{w}(t)^T \vec{x}(t)) \neq y(t)$
  - If no such sample exists
    - Return $\vec{w}(t)$
  - Else
    - $\vec{w}(t + 1) \leftarrow \vec{w}(t) + y(t)\vec{x}(t)$

Notation:
We use $\vec{w}(t)$ to denote the value of $\vec{w}$ at step $t$ of the algorithm.
Similarly, we use $(\vec{x}(t), y(t))$ to denote the data point found at step $t$. 
Perceptron Learning Algorithm (PLA)

• Theorem (informal):
  • If a dataset $D$ is linearly separable, PLA find a linear separator that separates the data in $D$ within a finite number of steps.

• HW0: Prove the above theorem
Perceptron

• Graphical Representation

Inspired by neurons:
The output signal is triggered when the weighted combination of the inputs is larger than some threshold

• Deep learning (neural network with many layers)
Lecture Today

The notes are not intended to be comprehensive.
Let me know if you spot errors.
Given by the learning problem

\[ y_n = f(x_n) \]

(ideal credit approval formula)

**Training Examples**

\((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)

(historical records of credit customers)

**Learning Algorithm**

\( A \)

**Final Hypothesis**

\( g \approx f \)

(learned credit approval formula)

**Hypothesis Set**

\( \mathcal{H} \)

(set of candidate formulas)

**Goal of Learning**
How Do We Formally Characterize the Goal?

• Goal of learning: find $g \approx f$
  • $f$: unknown target function
  • $g$: output of the learning algorithm
  • What do we mean by $g \approx f$?

• Main idea: **Generalization**
  • Want $g$ to make predictions similar to $f$ for **unseen data points**

Focus of today’s lecture:
• Feasibility of learning
• Can we achieve generalization?
\[ f(x) = +1 \]

\[ f(x) = -1 \]

Predict for unseen points (Generalization)

\[ f(x) = ??? \]
\[ h(x) = \begin{cases} +1 & \text{if symmetric} \\ -1 & \text{otherwise} \end{cases} \]

Hypothesis 1

\[ h \left( \begin{array}{ccc} \text{black} & \text{white} & \text{white} \\ \text{white} & \text{black} & \text{white} \\ \text{white} & \text{white} & \text{black} \end{array} \right) = +1 \]

Hypothesis 2

\[ h(x) = \begin{cases} +1 & \text{if top left is white} \\ -1 & \text{otherwise} \end{cases} \]

\[ h \left( \begin{array}{ccc} \text{black} & \text{white} & \text{white} \\ \text{white} & \text{black} & \text{white} \\ \text{white} & \text{white} & \text{black} \end{array} \right) = -1 \]

You can come up with many more hypotheses.
Feasibility of Learning

• Is learning feasible (can we generalize the learning)?
  • Cannot know anything \textit{for sure} about $f$ outside the data without assumptions
  • We might need to give up the \textit{“for sure”} and make additional assumptions

• Thought experiments: Which hypothesis would you choose? Why?
Key assumption of ML

*Training* data points and *testing* data points are *i.i.d.* drawn from the same (unknown) distribution

• Remarks
  • Modern ML is built on *probabilistic inference* with this assumption
  • The assumption is a reasonable approximation in many useful scenarios.
  • The assumption might not hold in other cases
    • There are various research efforts on this, but it’s outside of the scope of this course
UNKNOWN TARGET FUNCTION

\[ f : \mathcal{X} \mapsto \mathcal{Y} \]

\[ y_n = f(x_n) \]

TRAINING EXAMPLES
\[ (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \]

LEARNING ALGORITHM
\[ \mathcal{A} \]

HYPOTHESIS SET
\[ \mathcal{H} \]

FINAL HYPOTHESIS
\[ g \]
UNKNOWN TARGET FUNCTION

\[ f : \mathcal{X} \mapsto \mathcal{Y} \]

\[ y_n = f(x_n) \]

TRAINING EXAMPLES

\((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)

LEARNING ALGORITHM

\(\mathcal{A}\)

HYPOTHESIS SET

\(\mathcal{H}\)

UNKNOWN INPUT DISTRIBUTION

\(P(x)\)

\(x_1, x_2, \ldots, x_N\)

\(x\)

\(g(x) \approx f(x)\)

FINAL HYPOTHESIS

\(g\)
Let’s discuss probability first

We’ll then talk about how it connects back to machine learning
A Thought Experiment about Probability

What can we say about \( \mu \) from \( \nu \)?

Law of large numbers
- When \( N \to \infty, \nu \to \mu \)

Hoeffding’s Inequality
- \( \Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \) for any \( \epsilon > 0 \)
Interpretations

\[ \Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \]

- **Define** \( \delta = \Pr[|\mu - \nu| > \epsilon] \)
  - Probability of the bad event
  - Probability of the bad event is bounded by \( 2e^{-2\epsilon^2 N} \)

- **A tradeoff between** \( \delta, \epsilon, N \)
  - Fix \( \epsilon, \delta = O(e^{-N}) \)
  - Fix \( N, \delta = O(e^{-\epsilon^2}) \)
  - Fix \( \delta, \epsilon = O(\sqrt{1/N}) \)

- **For example,** \( N=1000 \)
  - \( \mu - 0.05 \leq \nu \leq \mu + 0.05 \) with 99% chance
  - \( \mu - 0.10 \leq \nu \leq \mu - 0.10 \) with 99.9999996% chance
Interpretations

\[ \Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \]

- Define \( \delta = \Pr[|\mu - \nu| > \epsilon] \)
  - Probability of the bad event

- For example, \( N=1000 \)
  - \( \mu - 0.05 \leq \nu \leq \mu + 0.05 \) with 99% chance
  - \( \mu - 0.10 \leq \nu \leq \mu - 0.10 \) with 99.9999996% chance

- \( \nu \) is approximately close to \( \mu \) with high probability
- \( \nu \) as an estimate of \( \mu \) is probably approximately correct (P.A.C.)

PAC learning is proposed by Leslie Valiant, who wins the Turing award in 2010.
Connection to Learning

• Let each marble represent a point $\hat{x}$, drawn from unknown $P(\hat{x})$
  • Dataset $D = \{(\hat{x}_1, y_1), \ldots, (\hat{x}_N, y_N)\}$
  • Recall that $y_n = f(\hat{x}_n)$ (will discuss noisy target function $f$ later in the semester)

• “Fix” a hypothesis $h$
  • For each marble $\hat{x}$, color it as below
    • If $h(\hat{x}) = f(\hat{x})$, color it as **green marble** [$h$ is **correct** on $\hat{x}$]
    • If $h(\hat{x}) \neq f(\hat{x})$, color it as **red marble** [$h$ is **wrong** on $\hat{x}$]
Connection to Learning

• Let each marble represent a point \( \vec{x} \), drawn from unknown \( P(\vec{x}) \)
  - Dataset \( D = \{ (\vec{x}_1, y_1), ..., (\vec{x}_N, y_N) \} \)
  - Recall that \( y_n = f(\vec{x}_n) \) (will discuss noisy target function \( f \) later in the semester)

• “Fix” a hypothesis \( h \)
  - For each marble \( \vec{x} \), color it as below
    • If \( h(\vec{x}) = f(\vec{x}) \), color it as green marble [\( h \) is correct on \( \vec{x} \)]
    • If \( h(\vec{x}) \neq f(\vec{x}) \), color it as red marble [\( h \) is wrong on \( \vec{x} \)]

• With the above coloring

\[
\mu = \Pr_{\vec{x} \sim P(\vec{x})} \left[ h(\vec{x}) \neq f(\vec{x}) \right] \quad \text{def} \quad E_{out}(h) \quad \text{[Out-of-sample error of } h\text{]}
\]
\[
\nu = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[h(\vec{x}_n) \neq f(\vec{x}_n)] \quad \text{def} \quad E_{in}(h) \quad \text{[in-sample error of } h\text{]}
\]
Connection to Learning

- Look at the error again
  - $E_{out}(h)$: What we really care about but unknown to us
  - $E_{in}(h)$: What we can calculate from dataset $D$

- Fixed a $h$, What can we say about $E_{out}(h)$ from $E_{in}(h)$?

  **Hoeffding’s Inequality**

  $$\Pr[|E_{out}(h) - E_{in}(h)| > \epsilon] \leq 2e^{-2\epsilon^2N} \quad \text{for any } \epsilon > 0$$

- Are we done?
  - Not really, this is verification, not learning

\[
\Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N}
\]
Verification vs. Learning

• Verification
  • I have a hypothesis \( h \).
  • I know \( E_{in}(h) \), i.e., how well \( h \) performs in my dataset.
  • I can infer what \( E_{out}(h) \) (how well \( h \) will perform for unseen data) might be.

• Learning
  • Given a dataset \( D \) and hypothesis set \( H \).
  • Apply some learning algorithm, that outputs a \( g \in H \).
  • Know \( E_{in}(g) \).
  • Want to infer \( E_{out}(g) \).
Connection to “Real” Learning

• Given a finite hypothesis set \( H = \{h_1, \ldots, h_M\} \)

• Apply some learning algorithm on \( D \), output a \( g \in H \)
  • For example, choosing the hypothesis that minimizes in-sample error
    • \( g = \arg\min_{h \in H} E_{in}(h) \)

• Can we apply Hoeffding’s inequality and claim
  \[
  \Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0
  \]

• No!
Consider this example

• If you toss a fair coin 10 times, the prob that you get heads 10 times is

\[ 2^{-10} = \frac{1}{1024} \]

• If you toss 1000 fair coins 10 times each, the probability that at least one coin comes up heads 10 times is

\[ 1 - \left(\frac{1023}{1024}\right)^{1000} \approx 62.36\% \]

• If each hypothesis is doing random guessing (i.e., tossing a fair coin), if we have 1000 hypothesis with 10 data points, more than 60% chance there will be at least one hypothesis with zero in-sample error

  • But that hypothesis is still random guessing and has 50% out-of-sample error