Logistics

• Course website and Piazza
  • Website: http://chienjuho.com/courses/cse417t/
  • Piazza: http://piazza.com/wustl/spring2020/cse417t
  • Make sure you follow both regularly

• Office hours
  • Will be announced later this week
  • Will start next week
Logistics

• Homework 1
  • Will be announced tomorrow or before lecture on Thursday
  • Expected due: Feb 19 (Friday)
  • Mixture of math questions and programming questions (implement PLA)
    • Programming language: Python
    • We won’t teach you how to program python
    • Basic sessions? (e.g., for environment setup, etc)

• Exam and Grades
  • Two exams (one in the middle of semester, one in the last day)
  • What to expect for the final grades
Recap
Given by the learning problem

**UNKNOWN TARGET FUNCTION**

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

(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 model

(example: H: Perceptron A: PLA)

**LEARNING ALGORITHM**

\( A \)

**FINAL HYPOTHESIS**

\( g \approx f \)

(learned credit approval formula)

Goal of learning
Goal of Learning: **Generalization**

• Given training data, find $g \approx f$ on the unseen testing data.

• This goal is generally impossible without assumptions.

**Key assumption of ML**

*Training* data points and *testing* data points are i.i.d. drawn from the same (unknown) distribution
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
\[ P(x) \]

FINAL HYPOTHESIS
\[ g \]

\[ x_1, x_2, \ldots, x_N \]

\[ g(x) \approx f(x) \]
A Thought Experiment about Probability

Hoeffding’s Inequality

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

for any \(\epsilon > 0\)

Law of large numbers

- When \(N \to \infty\), \(\nu \to \mu\)

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

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

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

• With the above coloring

\[
\mu = \Pr_{\tilde{x} \sim P(\tilde{x})} [h(\tilde{x}) \neq f(\tilde{x})] \quad \overset{\text{def}}{=} \quad E_{out}(h) \quad \text{[Out-of-sample error of } h]}
\]

\[
\nu = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[h(\tilde{x}_n) \neq f(\tilde{x}_n)] \quad \overset{\text{def}}{=} \quad E_{in}(h) \quad \text{[in-sample error of } h]\]
Connection to Learning

- $E_{out}(h)$: What we really want to know but unknown to us
- $E_{in}(h)$: What we can calculate from dataset

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

**Hoeffding’s Inequality**

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

- This is verification, not learning!
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 $H = \{h_1, \ldots, h_M\}$
  - Will discuss the infinite case in the next few lectures.

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

- No!
Today’s Lecture

The notes are not intended to be comprehensive.
Let me know if you spot errors.
JELLY BEANS CAUSE ACNE!

SCIENTISTS! INVESTIGATE!

BUT WE'RE PLAYING MINECRAFT!

...FINE.

WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE (P > 0.05).

THAT SETTLES THAT.

I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.

SCIENTISTS!

BUT MINECRAFT!
From xkcd, by Randall Munroe: http://xkcd.com/882
Another Analogy

• 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
One More Analogy

• Three fair coins, numbered by 1, 2, 3. Flip each 10 times

• Question: (choosing from $>5$, $=5$, or $<5$)

  Ans: = 5
  • For coin 1, what’s the expected number of heads among 10 flips?

  Ans: = 5
  • Randomly choose a coin, what’s the expected number of heads for this coin?

  Ans: < 5
  • After observing the realized flips, choosing the coin with the smallest number of heads, what is the expected number of heads for the coin?

  Ans: = 5
  • Without observing the flips, choose the coin anyway you like, what is the expected number of heads of the 10 flips for this coin?

• You will simulate this process (with 1,000 coins) in HW1.
One More Analogy

• Connects to learning
  • Coin -> Hypothesis
  • Coin flips -> Performance of hypothesis in training data $D$

• Choosing the hypothesis “before” or “after” looking at the data (knowing the realization of the data drawing) makes a very big difference!
What Can We Do?
Connection to “Real” Learning

• Given a finite hypothesis set \( H = \{ h_1, \ldots, h_M \} \)
• Apply some learning algorithm on \( D \), output a \( g \in H \)

• Question: What can we say about \( E_{out}(g) \) from \( E_{in}(g) \)?
Derivations

• Define "bad event of $h$" $B(h)$ as $|E_{out}(h) - E_{in}(h)| > \epsilon$
  • Informally, you can interpret “bad event of $h$” as the event that we draw a “unrepresentative dataset $D$” that makes the in-sample errors of $h$ to be far away from out-of-sample error of $h$

For each fixed $h \in H$, we have $\Pr[B(h)] \leq 2e^{-2\epsilon^2 N}$

• Recall $g$ is selected from $H$ (it could be any $h \in H$)
• What can we say about $\Pr[B(g)]$?
Derivations

• Define "bad event of $h$" $B(h)$ as $|E_{out}(h) - E_{in}(h)| > \epsilon$
  
  • Informally, you can interpret “bad event of $h$” as the event that we draw a “unrepresentative dataset $D$” that makes the in-sample errors of $h$ to be far away from out-of-sample error of $h$

For each fixed $h \in H$, we have $\Pr[B(h)] \leq 2e^{-2\epsilon^2N}$

• Recall $g$ is selected from $H$ (it could be any $h \in H$)
• What can we say about $\Pr[B(g)]$?

\[
\Pr[B(g)] \leq \Pr[B(h_1) \text{ or } B(h_2) \text{ or } \ldots \text{ or } B(h_M)] \\
\leq \Pr[B(h_1)] + \Pr[B(h_2)] + \ldots + \Pr[B(h_M)] \\
\leq M \, 2e^{-2\epsilon^2N}
\]
Connection to “Real” Learning

• Given a finite hypothesis set $H = \{h_1, \ldots, h_M\}$
• Apply some learning algorithm on $D$, output a $g \in H$

• Question: What can we say about $E_{out}(g)$ from $E_{in}(g)$?

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

• $M$ can be considered as a proxy of the “complexity” of the hypothesis set
  • Will talk about what happens when $M \to \infty$ in the next few lectures
Interpreting $\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2M e^{-2\epsilon^2 N}$

• Playing around with the math
  • Define $\delta = \Pr[|E_{out}(g) - E_{in}(g)| > \epsilon]$
  • We have $\delta \leq 2M e^{-2\epsilon^2 N} \Rightarrow \epsilon \leq \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

• This means, with probability at least $1 - \delta$
  • $E_{out}(g) \leq E_{in}(g) + \epsilon \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$
More Discussion

- With probability at least \( 1 - \delta \)
  \[
  E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}
  \]

Consider \( M \) as a proxy measure on the “complexity” of \( H \)

- Our ultimate goal is to have a small \( E_{out}(g) \)
  - There is a tradeoff of choosing \( M \) (what “learning model” to use)
    - Increase \( M \) -> Smaller \( E_{in}(g) \) (more hypothesis to “fit” the training data)
    - Increase \( M \) -> Larger \( \epsilon \)
  - It also depends on \( N \), the number of data points you have
    - A small number of data points => use simple models (e.g., linear models)
    - Complex models (e.g., deep learning) work when you have a lot of data
Revisit the Learning Problem
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
\[ A \]

UNKNOWNS INPUT DISTRIBUTION
\[ P(x) \]

\[ x_1, x_2, \ldots, x_N \]

HYPOTHESIS SET
\[ \mathcal{H} \]

FINAL HYPOTHESIS
\[ g \]

\[ g(x) \approx f(x) \]
Goal: \( g \approx f \)

- A general approach:
  - Define an error function \( E(h, f) \) that quantify how far away \( g \) is to \( f \)
  - Choose the one with the smallest error (empirical risk minimization)
  - For example: \( g = \arg\min_{h \in H} E(h, f) \)

- \( E \) is usually defined in terms of a pointwise error function \( e(h(\tilde{x}), f(\tilde{x})) \)
  - Binary error (classification): \( e(h(\tilde{x}), f(\tilde{x})) = \mathbb{1}[h(\tilde{x}_n) \neq f(\tilde{x}_n)] \) (What we have discussed so far)
  - Squared error (regression): \( e(h(\tilde{x}), f(\tilde{x})) = (f(\tilde{x}) - h(\tilde{x}))^2 \)

- In-sample and out-of-sample errors
  - \( E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} e(h(\tilde{x}_n), f(\tilde{x}_n)) \)
  - \( E_{out}(h) = \mathbb{E}_{\tilde{x}}[e(h(\tilde{x}), f(\tilde{x}))] \)

The discussion on the Hoeffding’s inequality applies for general (bounded) error functions.
How to choose the error function?

• Consideration 1: Properties of application problems

Example: Fingerprint recognition
  • Input: fingerprints
  • Outputs: whether the person is authorized

\[
\begin{array}{c|c|c}
 h(\vec{x}) & f(\vec{x}) & \\
 +1 & +1 & \text{No error} \\
 -1 & -1 & \text{No error} \\
 +1 & -1 & \text{False positive} \\
 -1 & +1 & \text{False negative} \\
\end{array}
\]

• Errors assigned to false negative/positive differ depending on applications
  • Supermarket coupons vs FBI
  • False positive is a big issue for FBI but probably fine for supermarket coupons
How to choose the error function?

• Consideration 1: Properties of application problems

• Consideration 2: Computation
  • ML Algorithm is essentially doing optimization (finding $g$ with smallest error)

$$ g = \underset{h \in \mathcal{H}}{\text{argmin}} \ E(h, f) $$

• Choosing the error that is “easier” to optimize
How to choose the error function?

• Consideration 1: Properties of application problems

• Consideration 2: Computation

• Specifying the error function is part of setting up the learning problem
  • It impacts what you eventually learn
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
\[ A \]

HYPOTHESIS SET
\[ \mathcal{H} \]

UNKNOWN INPUT DISTRIBUTION
\[ P(x) \]

\[ x_1, x_2, \ldots, x_N \]

\[ x \]

FINAL HYPOTHESIS
\[ g \]

\[ g(x) \approx f(x) \]
Noisy Target

• What if there doesn’t exist $f$ such that $y = f(\vec{x})$?
  • $f$ is stochastic instead of deterministic

• Common approach
  • Instead of a target function, define a target distribution
  • Instead of $y = f(\vec{x})$, $y$ is drawn from a conditional distribution $P(y|\vec{x})$
  • $y = f(\vec{x}) + \epsilon$ where $\epsilon$ is zero-mean noise

The discussion on the Hoeffding’s inequality applies for noisy targets.
UNKNOWN TARGET DISTRIBUTION
(target function $f$ plus noise)

$P(y \mid x)$

$y_n \sim P(y \mid x_n)$

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

$x_1, x_2, \ldots, x_N \rightarrow x$

ERROR MEASURE

LEARNING ALGORITHM $\mathcal{A}$

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

FINAL HYPOTHESIS $g$

HYPOTHESIS SET $\mathcal{H}$