Lecture 1: Intro, ERM Framework

CS1420: Machine Learning

Stephen Bach
Spring 2020

Meet the Instructor

- Stephen Bach (Steve)
- He / him / his
- Email: stephen_bach@brown.edu
- Office: CIT 335
- Office hours by appointment. Just send a quick email! :)

Meet the TAs

Head TAs:
Angie Kim <jkim162>
Jessica Dai <jdai6>
Dylan Sam <dsam>

TAs:
Andrew Canino <acanino>
Andrew Wei <awei6>
Fabrice Guyot-Sionnest <fguyotsi>
Jacob Migneault <jmignea1>
Kelvin Yang <kyang35>
Shibei Guo <sguo16>
Snigdha Sinha <ssinha5>
Tyler Jiang <tjiang12>
Zsozso Biegl <zbiegl>
Andrew Peterson <apeter10>
Daniel Ben-Isvy <dbenisvy>
Irene Rhee <irhee>
Ken Noh <knoh1>
Rudra Srivastava <rsrivas2>
Seneca Meeks <smeeks>
Siyao Wang <swang181>
Yiming Zhang <yzhan281>

Ethics TAs:
Karen Tu <ktu2>
Kelvin Yang <kyang35>

Course Communications

- Our course website: http://cs.brown.edu/courses/csci1420/index.html

- We will be doing questions and announcements via Piazza: https://piazza.com/brown/spring2020/csci1420

Action Item: Visit the Piazza site to sign up

Waitlist

- Unfortunately, we are constrained by the capacity of the classroom
- See the Waitlist FAQ for all the info

Required Textbook

- Using http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/.

  (PDF is downloadable!)

- Chapters 1, 2.0, 2.1, 2.2 today.

- Parts of chapter 9 for next class (see schedule on course website)

- 1-page summary of notation on page 28!

Action Item: Get the textbook and start reading

Grading Breakdown

- 12 homeworks, written and programming (60%)
  - 4 late days, up to 3 on any assignment
- Midterm (15%)
  - March 19th (in-class)
- Final Exam (20%)
  - May 8th
- TopHat (5%)
  - Can miss 15 votes. After those 15, each missed vote reduces grade. Number of votes per class will vary! Instructions here.

Action Item: Set Up TopHat before next class

Collaboration and Other Policies

- Everyone is required to be familiar with the full missive: http://cs.brown.edu/courses/csci1420/docs/cs1420-missive.pdf
- The missive incorporates the full collaboration policy: http://cs.brown.edu/courses/csci1420/docs/cs1420collaborationpolicy.pdf
  - Students must turn in their own homeworks
  - Students may discuss the assignments
  - Students may not look at any other's work in progress
- To submit homeworks, you must agree to policies and set up GradeScope with this form

Action Item: Read missive, collaboration policy, and complete form

Homework 1

- Due Jan. 30 (1 week!)
- Exercises related to prerequisite topics
- Also some Python environment setup and programming
- Good test of appropriate preparation for this course

Action Item: Homework 1 due Jan. 30

Action Items

1. **Visit the Piazza site to sign up**
2. **Get the textbook and start reading**
3. **Set up TopHat before next class**
4. **Read missive, collaboration policy, and complete GradeScope form**
5. **Homework 1 due Jan. 30**

Course Philosophy vs. Other Courses in Brown CS

- (my biased take)
- This course’s approach: blend of practical and theoretical machine learning
- Some things are necessarily left out because of this choice
- Other classes at Brown that are more applied: Artificial Intelligence, Data Science, Computer Vision, Computational Linguistics
- Other classes at Brown that are more theory-based: Advanced Probabilistic Methods in Computer Science

What is Machine Learning?

Machine Learning as Program Generation

- Want to create programs based on data without explicitly programming them

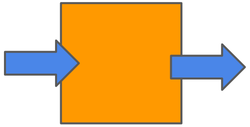
- **Representation**: Define a space of possible programs
    - *Tradeoffs: Expressiveness, simplicity, convenience…*

- **Loss function**: Decide how to score a program
    - *The data usually comes into play here*

- **Optimizer**: Search the space of programs for one with a high score
    - *The best scoring program is the one returned*

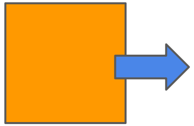
Three Kinds of Programs

Function



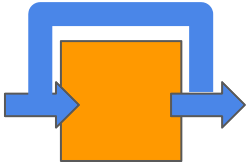
- Maps input to output
- Query/response, transformations

Generator



- Takes no input, produces output
- Content or random number generator

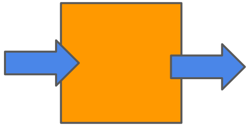
Interactive



- Takes some input, produces some output, expects more input
- Operating systems, games, UIs

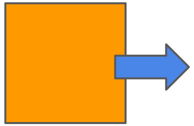
Three Kinds of Machine Learning

Supervised



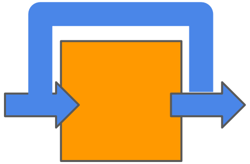
- Given input/output examples, finds mapping
- Predictive: What will happen? What’s missing?

Unsupervised



- Given data, finds a representation
- Descriptive: What happened?

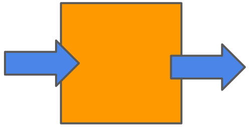
Reinforcement



- Translate state to action to maximize reward
- Prescriptive: What should we do?

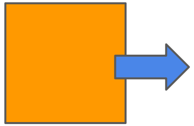
This Course

Supervised



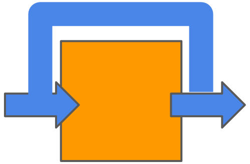
- Main focus

Unsupervised



- Some in April

Reinforcement



- Not in this course
- See CSCI 1410, CSCI 1470, etc.

Organizing Principle

ML algorithm = representation + loss function + optimizer

Note:

- Optimizer might not be perfect (computationally intractable).
- Loss/error function is with respect to data.

Example of Supervised Learning:
Does this animal have cute babies?

Problem: Is this baby cute? Want to go viral!

cute

ADORABLY

31 Pictures Of Baby Animals To Remind You The World Is Wonderful

We have a planet full of baby animals, so you should never be TOO sad.






Representation

1. Domain Set: How do we represent the objects we want to label?
2. Label Set: How do we represent the labels we want to predict?
3. Training Data: What labeled objects do we have access to?
4. Learner's Output: How do we represent what is learned?

Example Domain Set: Animals

	Num. Eyes	Num. Legs	Num. Fins
 Tiger	2	4	0
 Spider	8	8	0
 Shark	2	0	2
 Snake	2	0	0

Example Domain Set: Animals

Tiger

$$\mathbf{x}_1 = (2, 4, 0)$$

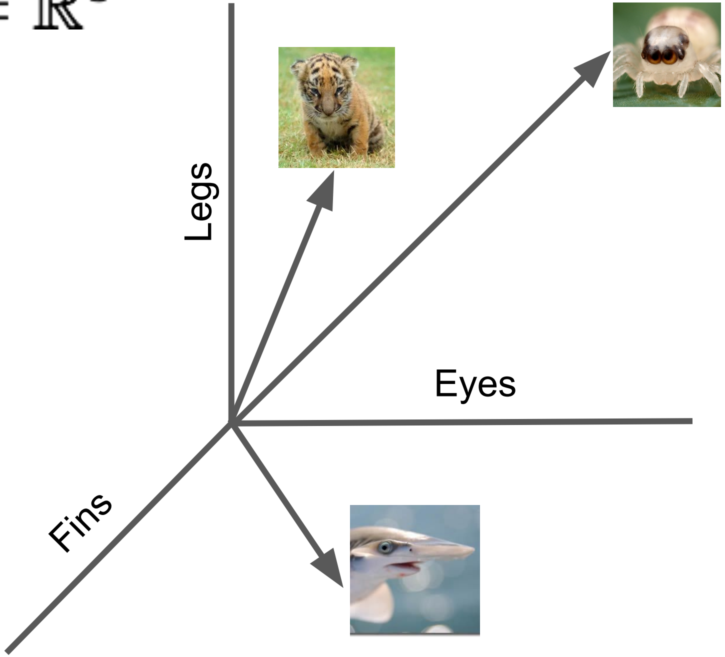
Spider

$$\mathbf{x}_2 = (8, 8, 0)$$

Shark

$$\mathbf{x}_3 = (2, 0, 2)$$

$$\mathcal{X} = \mathbb{R}^3$$



Representation

1. Domain Set: How do we represent the objects we want to label?
2. Label Set: How do we represent the labels we want to predict?
3. Training Data: What labeled objects do we have access to?
4. Learner’s Output: How do we represent what is learned?

Example Label Set: “Cute” or “Not Cute”

$$\mathcal{Y} = \{1, -1\}$$

1

-1

Representation

1. Domain Set: How do we represent the objects we want to label?
2. Label Set: How do we represent the labels we want to predict?
3. Training Data: What labeled objects do we have access to?
4. Learner's Output: How do we represent what is learned?

Training Data

Examples paired with labels determine what our learned program does

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$$

$$Z = \mathcal{X} \times \mathcal{Y}$$

Cute or not Cute?

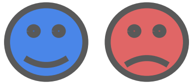
Cute or not Cute?

Cute or not Cute?

Cute or not Cute?

Example Training Data: Animals

	Num. Eyes	Num. Legs	Num. Fins	Cute?
 Tiger	2	4	0	
 Spider	8	8	0	
 Shark	2	0	2	
 Snake	2	0	0	



Representation

1. Domain Set: How do we represent the objects we want to label?
2. Label Set: How do we represent the labels we want to predict?
3. Training Data: What labeled objects do we have access to?
4. Learner's Output: How do we represent what is learned?

Example Learner's Output: Rules

$h : \mathcal{X} \to \mathcal{Y}$ $\mathcal{H} = \{h_1, h_2, \dots\}$

Ideas?

h_1 =

h_2 =

h_3 =

Representation

1. Domain Set: How do we represent the objects we want to label?
2. Label Set: How do we represent the labels we want to predict?
3. Training Data: What labeled objects do we have access to?
4. Learner’s Output: How do we represent what is learned?

ML algorithm = representation
+ loss function + optimizer

Example Loss Function: 0-1 Loss

- Empirical risk:

$$L_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \ell_{0-1}(h, \mathbf{z}_i)$$

- $L_S(h)$ is average of loss computed over each example:

$$\ell_{0-1}(h, (\mathbf{x}, y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } \ h(\mathbf{x}) = y \\ 1 & \text{if } \ h(\mathbf{x}) \neq y \end{cases}$$

ML algorithm = representation
+ loss function + optimizer

Optimization in this course:

empirical risk minimization

$$h_S \stackrel{\text{def}}{=} \mathrm{ERM}_{\mathcal{H}}(S) \in \operatorname*{arg\,min}_{h \in \mathcal{H}} L_S(h)$$

Example Optimizer: Brute Force Search

Which output has the lowest loss (empirical risk)?

$$\underset{h \in \mathcal{H}}{\arg\min}\, L_S(h) = ?$$

Just like in this course, there *will* be a test...

- We assume S was *sampled* from some distribution $\mathcal{D}$, i.e. $S \sim \mathcal{D}^m$
- After learning, we will get more samples from $\mathcal{D}$, and want to do well on them
  - Sometimes called the “test data”
- We do not know what the test data will be, but we want low *expected* loss:

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{\mathbf{z} \sim \mathcal{D}} [\ell(h, \mathbf{z})]$$

The Fundamental Challenge in Machine Learning

$$L_S(h) \neq L_{\mathcal{D}}(h)$$

- The training data is a finite sample of the test data
- Even if we find an empirical risk minimizer, will it do well at test time?

Potential Pitfall: Overfitting

What will happen at test time if…?

- What if we include this hypothesis in $\mathcal{H}$?
  - To classify an example, look in the training data. If there's an identical example, return its label. Otherwise, return -1.

  - Formally:

$$h(\mathbf{x}) = \begin{cases} y_i & \text{if } \exists i \in [m] \text{ s.t. } \mathbf{x}_i = \mathbf{x} \\ -1 & \text{otherwise} \end{cases}$$

- $L_S(h)$ is 0, but $L_{\mathcal{D}}(h)$ can be arbitrarily large (depends on unknown $\mathcal{D}$)

Conclusions

- The hypothesis class $\mathcal{H}$ cannot include all possible hypotheses, or learning is doomed to failure!

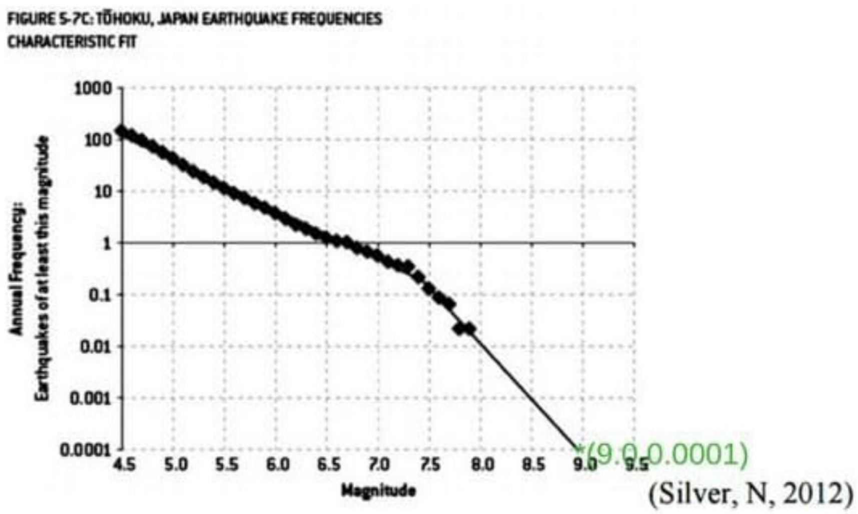
- We must choose a subset of all possible hypotheses to use as $\mathcal{H}$, capturing our prior knowledge about the domain

- Finding a hypothesis that does “too well” on the training data, but poorly on the test data is called *overfitting*

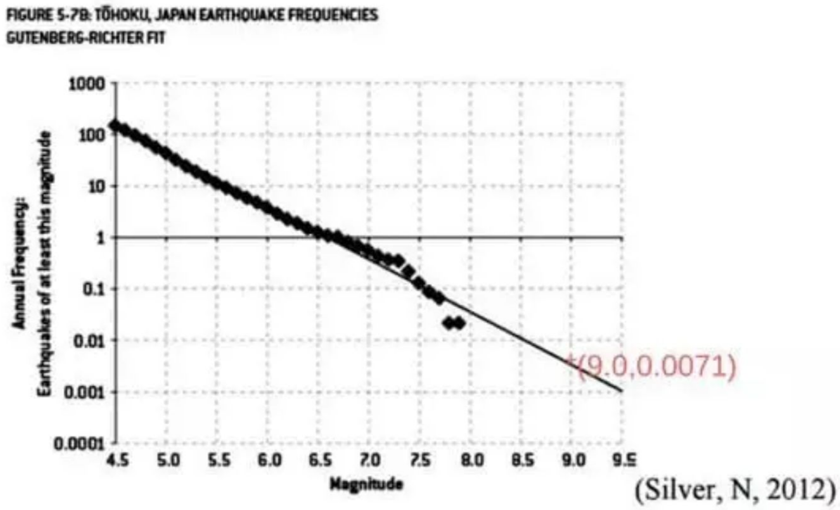
Example: Fukushima Nuclear Disaster

- Fukushima nuclear power plant was hit with a 9.1 earthquake in March 2011
- Most severe nuclear accident since Chernobyl
- The plant was designed to withstand an earthquake of up to 8.6
- What does this have to do with overfitting?

Fukushima Nuclear Disaster



Model learned by Fukushima's data scientists



Alternative model using a Gutenberg-Richter model (linear regression)

The Most Important Things

- We focus on supervised machine learning, with some unsupervised in April
- We use ***empirical risk minimization (ERM)***
  - ERM = pick a hypothesis that minimizes the loss (i.e. empirical risk) on a set of training data
- Naively applying ERM can lead to the pitfall of ***overfitting***
  - Overfitting = picking a hypothesis that is great on training data but very bad on new test data
- Textbook: chapter 1, sections 2.0, 2.1, 2.2

Next Class

- What is a practically useful class of hypotheses?
- How to select an ERM hypothesis from that class computationally efficiently?
- Textbook: sections 9.0, 9.1.0, 9.1.2