Algorithmic Aspects of Machine Learning

COMPSCI 590.7 Fall 2015
Rong Ge
Basic Info

- Contact me:
  LSRC D226
  Email: rongge@cs.duke.edu
- Email List:
  You are already on the list if you registered. Send me an email otherwise.
Basic Info

- **Expect to see (and do) Proofs.**
- **Homework:**
  ~3 problem sets, due 2 weeks after posted in class.
  Discussions allowed but *must acknowledge*.
- **Final Project:**
  Form groups of size 2-3, ~1 month.
  Can be original research or literature survey.
Lecture 1: Machine Learning Basics
Machine Learning Basics

- What do we want “machines” to learn?
- How do we know machines have learned?
- What guarantees can we hope to get?
- What tools do we have?
What do we want machines to learn?
Example: Dogs vs Cats

https://www.kaggle.com/c/dogs-vs-cats

Input: Many images with labels (cat or dog)
Goal: Given new image, decide cat or dog.
Example: Netflix

Input: Movie ratings from users
Goal: Recommend new movies for users
Example: Community Finding

Input: List of friends
Goal: Find “communities”
Supervised vs Unsupervised

- Input: (data, label)
- Output: Function $f$
- Hope: $f(data) = label$

- Input: data
- Output: “structure”
- Hope: explain and predict

Supervised

Unsupervised
Example of Structure

- Movies have different genres.
- Users like different genres.
- Learning ~ Find the genres of movies and users
- To recommend
  - determine what genres the user likes
  - recommend a good movie in that genre.
Example of Structure: Probabilistic Models

- People are in different communities
- Same community ⇒ Higher probability \( p \) to know each other
- Different community ⇒ Lower probability \( q \) to know each other
- Learning ~ Find out which groups of people are in the same community
Probablistic Models

Given: Data

Assumption: Data is generated from a class of distributions ("model")

Learning ~ Finding parameters of the model
Connection

MNIST: Given images recognize digits.

Unsupervised: Given images, cluster similar images

Unsupervised Learning $\Rightarrow$ features for Supervised Learning
How do we know machines have learned?

(General approach: Give an exam)
Human Learning, Machine Learning

- Take a course
- Understand the course material
- Take an exam

Goal of Exam: If the students understood the material, do well in exam.

- Get training examples
- Learn a hypothesis - maps input to labels
- Test on new examples

Goal of Test: If the hypothesis is good, do well in test samples.
What is a good hypothesis?

Ways to fail an exam

- Do not understand examples given in class (Make many mistakes on training samples)
- Only memorize the basic examples (Come up with a complicated hypothesis)

Good hypothesis = Simple + Do well on training
PAC Learning

[Valiant 1984] For a concept class $C$, for any distribution $D$ on data points, given samples $(x,y): x \sim D, y = c(x)$ for fixed $c$ in $C$ an algorithm PAC learns the concept class if with probability $1-\delta$ the algorithm outputs a function $f$ such that

$$\Pr_{x \sim D}[f(x) = c(x)] \geq 1 - \epsilon$$
Example

- Concept Class: Boxes
- Get samples and labels
- f may not be a box
- Tested on new samples

Generalization
If f is “simple”, doing well on the current samples guarantees good performance on future samples.
Exams for Unsupervised Learning

Make (probabilistic) predictions

What other movies do this user like?

What other people do this user know?

Generalization

If the model is “simple”, explaining the current samples guarantees good prediction on future samples.
Alternative Guarantees

- Parameter Estimation:
  - Estimate parameters of the model
    - (what movies are comedies, who are in same community…)
  - Assumes the model is “correct”.
  - Easier to work with, often see in this course

- Maximum Likelihood Estimation:
  - Find the parameters that are best at explaining data
  - Only hope to do as well as the “best model”
  - Still restrict to the model (like restricting $f \sim \text{box}$)

All models are wrong, but some are useful.
Learning Parameters vs Making Predictions

Parameter Estimation
- Has a well-defined model.
- Easy to explain/interpret.
- Concise representation.

Making Prediction
- More robust to model mismatch.
- More efficient algorithms.
- Often what we really want.

Often
Not Necessary
What guarantees can we hope to get?

(and why do we care about guarantees?)
Why do we want guarantees?

- Understand why learning works (or not work)
  - Does the algorithm work with high dimension/high noise/…?
- Make sound conclusions
  - “I tried to find a good set of parameters but failed”
  - vs. “If there is a good set of parameters I’ll find it”
- Design better algorithms
  - Can I tweak the algorithm if data is not ideal?
  - Get new ideas by different way of thinking.
What we are not likely to do

● PAC learning for many problems
  ○ intersection of halfplanes
  ○ 2 layer “neural network”

● Learn many functions even with specific distribution
  ○ Learning parities with noise...

● Maximum Likelihood for many problems
  ○ mixture of Gaussians,
  ○ topic models
The Hope

Natural Instance

Reasonable Model
Reasonable Assumptions

Machine Learning Problem (worst case, for every...)
Example: Netflix

Input: Movie ratings from users
Goal: Recommend new movies for users
Natural Instances

- Models are not perfect, but they can work
  - Breaking movies into genres is not perfect, but allows reasonable commendation

- Natural instances are often easier
  - People usually have a preference over different genres, ...

- “What is natural” is the hard problem
  - We will see examples later in the course.
What we will see

Matrix Factorization

Social Network

Independent Component Analysis

Topic Models

Tensor

Tensor Methods
What tools do we have?
Tools

● Geometry
  ○ Nonnegative constraints
  ○ Finding extreme points

● Linear Algebra
  ○ Spectral methods
  ○ Tensor decomposition

● Optimization
  ○ use convex programs (LP, SDP) in learning
  ○ optimize a nonconvex function

● Hope: Will have more tools!
Schedule

- Nonnegative Matrix Factorization and Topic Models (~4 lectures)
- Spectral Clustering (~2 lectures)
- Tensor Decompositions (~5 lectures)
- (Non-convex) Optimization (~5 lectures)
- Matrix Completion (~4 lectures)