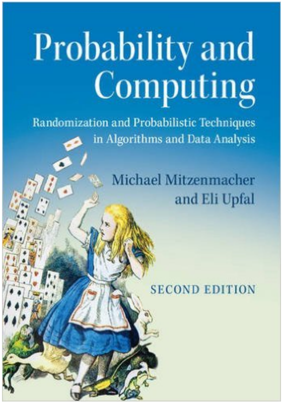
CS 1550/2540: Probabilistic Methods in Computer Science

Introduction + Chapter 1



Overview

- CSCI 1550/2540 is a theory/math course - analysis, theorems, proofs. No implementations.
- The course covers modern mathematics at the interface of probability theory and computation
- Formulates, and explains many of the great successes of computing, such as machine learning, cryptography, modern finance, computational biology, etc.
- The course focuses on tools, not applications.

Why Probabilistic Methods?

Almost any advance computing application today has some randomization/statistical/machine learning components:

- Randomized algorithms - random steps help!
  - Efficiency: Hashing, Quicksort, ...
  - Security and Privacy: Open key cryptography, one way function,...
  - Monte Carlo methods: scientific computing, finance, weather forecast,...
- Probabilistic analysis of algorithms - Theoretically "hard to solve" problems are often not that hard in practice.
  - Average case analysis
  - Almost always analysis
- Modeling data
  - Statistical machine learning
  - Data mining
  - Recommendation systems

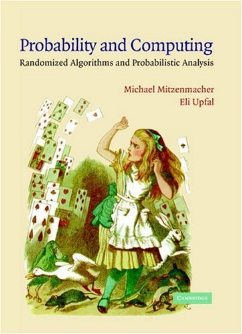
Course Details - Main Topics

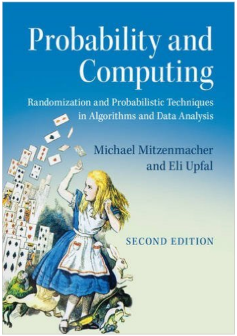
1. Basic randomized algorithms and probabilistic analysis of algorithms (and review of relevant probability theory concepts)
2. Large deviation bounds: Chernoff and Hoeffding bounds
3. The probabilistic method
4. Martingale (in discrete space)
5. Theory of statistical learning, PAC learning, VC-dimension
6. Monte Carlo methods
7. Convergence of Monte Carlo Markov Chains
8. ...

This course emphasize rigorous mathematical approach, mathematical proofs, and analysis.

Course Details

- Pre-requisite: Discrete probability theory (first three chapters in the course textbook); mathematical maturity; interest in theory of algorithms
- Course textbook:





- Follow: Ed-discussion, Canvas, and the course website: https://cs.brown.edu/courses/csci1550/

Course Work and Grading:

For 1550 credit: 80% homework assignments, 20% projects reviews.

For 1550 credit as a capstone course: 60% homework assignments, 30% project, 10% projects reviews.

For 2540 credit: 40% homework assignments, 50% project, 10% projects reviews.

Homework Assignments:

- 4-5 assignments (problem sets) - graded for mathematical corrections and mathematical style.
- All but the last problem set can be submitted by groups of 1-3 students. Each group submits one write-up and all members of the group will receive the same grade on that assignment.
- **Deadlines:**
  - Assignments submitted by their deadline will be graded and returned with corrections.
  - Assignments submitted after their deadline will be graded but not retuned with corrections.
  - No assignment will be accepted after the last class.

Homework Assignments (cont.):

Assignments collaboration policy:

- You may not discuss the problems with students outside your group.
- The last problem set must be done individually with no discussion and/or collaboration with other students.
- In preparing the assignments you are allowed to use the textbook, the course's slides, the discussions on the course's Ed-discussion and with the TA's. **Any other source must be disclosed in the submission.**

Some of the homework assignments are challenging! You're not expected to solve all problems, but you're expected to try!

Project:

- Submit a 6-8 page research project (with possible appendices) on any application of probabilistic method in CS.
- 15-minute presentation on your project in class.
- A project can be done by 1 or 2 students. Two students who submit a joint project will receive the same grade on their project.
- Students who choose to prepare projects will interact with the instructor and the TA's on preparing their project throughout the course.
- **Project reviews:** Depending on the total number of projects, you will be asked to submit short reviews for all, or some of the projects.
- If you plan to submit a project: (1) let the HTA know by Feb. 15; (2) Submit a title and one paragraph plan by Feb. 29.

We treat you as adults...

- You don't need to attend class - but you cannot ask the instructor/TA's to repeat information given in class.
- HW-0, not graded. Do your own evaluation! DON'T take this course if you don't enjoy HW-0 type exercises
- Don't postpone the HW assignments to the last day - you cannot do it in one evening!
- Make good use of course's resources:
  - Course's slides
  - book
  - TA's hours
  - Ed-Discussion

Questions?

Verifying Matrix Multiplication

Given three $n \times n$ matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in a Boolean field, we want to verify

$$\mathbf{AB} \;=\; \mathbf{C}.$$

Standard method: Matrix multiplication - takes $\Theta(n^3)$ ($\Theta(n^{2.37})$) operations (multiplications).

Verifying Matrix Multiplication

Randomized algorithm (takes $\Theta(n^2)$ multiplications):

1. Chooses a random vector $\bar{r} = (r_1, r_2, \ldots, r_n) \in \{0,1\}^n$.
2. Compute $\mathbf{B}\bar{r}$;
3. Compute $\mathbf{A}(\mathbf{B}\bar{r})$;
4. Computes $\mathbf{C}\bar{r}$;
5. If $\mathbf{A}(\mathbf{B}\bar{r}) \neq \mathbf{C}\bar{r}$ return $\mathbf{AB} \neq \mathbf{C}$, else return $\mathbf{AB} = \mathbf{C}$.

Theorem

If $\mathbf{AB} \neq \mathbf{C}$, and $\bar{r}$ is chosen uniformly at random from $\{0,1\}^n$, then

$$\Pr(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \leq \frac{1}{2}.$$

If $\mathbf{AB} = \mathbf{C}$ the algorithm is always correct. If $\mathbf{AB} \neq \mathbf{C}$, the algorithm gives the correct answer with probability $\geq 1/2$

Probability Space

Probability is always with respect to a probability space!

What is the probability that the sun will rise tomorrow?
Pierre-Simon de Laplace - 1749–1827.

Definition

A probability space has three components:

1. A sample space Ω, which is the set of all possible outcomes of the random process modeled by the probability space;
2. A family of sets $\mathcal{F}$ representing the allowable events, where each set in $\mathcal{F}$ is a subset of the sample space Ω;
3. A probability function $\Pr : \mathcal{F} \to [0, 1]$ defining a measure.

In a discrete probability an element of Ω is a simple event, and $\mathcal{F} = 2^{\Omega}$.

Probability Function

Definition

A probability function is any function $\Pr : \mathcal{F} \to \mathbf{R}$ that satisfies the following conditions:

1. For any event E, $0 \leq \Pr(E) \leq 1$;
2. $\Pr(\Omega) = 1$;
3. For any finite or countably infinite sequence of pairwise mutually disjoint events $E_1, E_2, E_3, \ldots$

$$\Pr\left(\bigcup_{i \geq 1} E_i\right) = \sum_{i \geq 1} \Pr(E_i).$$

In discrete sample space, the probability of an event is the sum of the probabilities of its simple events.

Lemma

Choosing $\bar{r} = (r_1, r_2, \ldots, r_n) \in \{0,1\}^n$ *uniformly at random is equivalent to choosing each* r_i *independently and uniformly from* $\{0,1\}$*.*

Proof.

If each r_i is chosen independently and uniformly at random, each of the 2^n possible vectors $\bar{r}$ is chosen with probability 2^{-n}, giving the lemma. $\square$

Proof:

Assume $\mathbf{D} = \mathbf{AB} - \mathbf{C} \neq 0$.
$\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}$ implies that $\mathbf{D}\bar{r} = 0$.
Since $\mathbf{D} \neq 0$ it has some non-zero entry; assume d_{11}.
For $\mathbf{D}\bar{r} = 0$, it must be the case that

$$\sum_{j=1}^{n} d_{1j} r_j = 0,$$

or equivalently

$$r_1 = -\frac{\sum_{j=2}^{n} d_{1j} r_j}{d_{11}}. \tag{1}$$

Here we use $d_{11} \neq 0$.

Principle of Deferred Decision

Assume that we fixed $r_2, \ldots, r_n$.
The RHS is already determined, the only variable is r_1.

$$r_1 \quad = \quad -\frac{\sum_{j=2}^{n} d_{1j} r_j}{d_{11}}. \tag{2}$$

Probability that $r_1 =$ RHS is no more than $1/2$. ($= 1/2$ in the Boolean field.)

Theorem

If $\mathbf{AB} \neq \mathbf{C}$, and $\bar{r}$ is chosen uniformly at random from $\{0,1\}^n$, then

$$\Pr(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \leq \frac{1}{2}.$$

$\leq 1/2$ because it must hold on all non-zero rows of D.

More formally, summing over all collections of values $(x_2, x_3, x_4, \ldots, x_n) \in \{0,1\}^{n-1}$, we have

$$
\begin{aligned}
& \Pr(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \\
= & \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} \Pr\left(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r} \mid (r_2,\ldots,r_n) = (x_2,\ldots,x_n)\right) \\
& \qquad\qquad \cdot \Pr\left((r_2,\ldots,r_n) = (x_2,\ldots,x_n)\right) \\
= & \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} \Pr\left((\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \cap ((r_2,\ldots,r_n) = (x_2,\ldots,x_n))\right) \\
\leq & \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} \Pr\left(\left(r_1 = -\frac{\sum_{j=2}^{n} d_{1j} r_j}{d_{11}}\right) \cap ((r_2,\ldots,r_n) = (x_2,\ldots,x_n))\right) \\
= & \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} \Pr\left(r_1 = -\frac{\sum_{j=2}^{n} d_{1j} r_j}{d_{11}}\right) \cdot \Pr\left((r_2,\ldots,r_n) = (x_2,\ldots,x_n)\right) \\
\leq & \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} \frac{1}{2} \Pr((r_2,\ldots,r_n) = (x_2,\ldots,x_n)) \\
= & \frac{1}{2}.
\end{aligned}
$$

Smaller Error Probability

The test has a one side error, repeated tests are independent.

- Run the test k times.
- Accept $AB = C$ if it passed all k tests.

Theorem

The probability of making a mistake is $\leq (1/2)^k$.

Independent Events

Definition

Two events E and F are independent if and only if

$$\Pr(E \cap F) \;=\; \Pr(E) \cdot \Pr(F).$$

More generally, events $E_1, E_2, \ldots E_k$ are mutually independent if and only if for any subset $I \subseteq [1, k]$,

$$\Pr\left(\bigcap_{i \in I} E_i\right) \;=\; \prod_{i \in I} \Pr(E_i).$$

If E and F are independent then the probability of E does not depend on F.

$$\Pr(E \mid F) = \frac{\Pr(E \cap F)}{Pr(F)} = \frac{\Pr(E) \cdot \Pr(F)}{Pr(F)} = Pr(E)$$

Min-Cut

A graph $G = (V, E)$, V-set of vertices, E set of edges.

A Min-Cut set - A minimum set of edges that disconnects the graph.

Min-Cut

A graph $G = (V, E)$, V-set of vertices, E set of edges.

A Min-Cut set - A minimum set of edges that disconnects the graph.

Fundamental computation problem in transportation, network reliability, arbitrage, ...

Has a deterministic polynomial time solution.

Any ideas?

Min-Cut

A graph $G = (V, E)$, V-set of vertices, E set of edges.

A Min-Cut set - A minimum set of edges that disconnects the graph.

Fundamental computation problem in transportation, network reliability, arbitrage, ...

Has a deterministic polynomial time solution.

Algorithm: Run a max flow algorithm ($O(|V|^2)$ or $O(|V| \cdot |E|)$ complexity) between all pairs of nodes ($\binom{|V|}{2}$).

Instead, we present and analyze a simple randomized algorithm with better run-time.

Min-Cut Algorithm

Input: An n-node graph G.
Output: A minimal set of edges that disconnects the graph.

1. **Repeat $n-2$ times:**
   1. Pick an edge uniformly at random.
   2. Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.
2. Output the set of edges connecting the two remaining vertices.

How good is this algorithm?

Does it always give a correct result?

Min-Cut Algorithm

Input: An n-node graph G.

Output: A minimal set of edges that disconnects the graph.

1. **Repeat $n-2$ times:**
   1. Pick an edge uniformly at random.
   2. Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.
2. Output the set of edges connecting the two remaining vertices.

Theorem

1. *The algorithm outputs a min-cut edge-set with probability* $\geq \frac{2}{n(n-1)}$.
2. *The smallest set output in* $O(n^2 \log n)$ *iterations of the algorithm gives a correct answer with probability* $1 - 1/n^2$.

Min-Cut Algorithm

Input: An n-node graph G.

Output: A minimal set of edges that disconnects the graph.

1. **Repeat $n-2$ times:**
   1. Pick an edge uniformly at random.
   2. Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.
2. Output the set of edges connecting the two remaining vertices.

Theorem

The algorithm outputs a min-cut edge-set with probability $\geq \frac{2}{n(n-1)}$.

Analysis of the Algorithm

Assume that the graph has a min-cut set of k edges.
We compute the probability of finding one such set C.
$Pr(\text{Alg. returns any minimal cut set}) \geq Pr(\text{Alg. returns } C)$

Two parts proof:

- **Deterministic analysis part:**

Lemma

If no edge of C was contracted, the algorithms outputs C.

- **Probabilistic analysis part:**

Lemma

$Pr(\textit{no edge of } C \textit{ is contracted}) \geq \frac{2}{n(n-1)}$.

Deterministic part:

Lemma (Correctness of a step)

If no edge of C was contracted, no edge of C was eliminated.

Proof.

Let X and Y be the two set of vertices cut by C.
If the contracting edge connects two vertices in X (res. Y), then all its parallel edges also connect vertices in X (res. Y). □

Corollary (Correctness of a run)

If the algorithm terminates before contracting any edge of C, the algorithm gives a correct answer.

Lemma (One side error)

Vertex contraction does not reduce the size of the min-cut set. Every cut set in the new graph is a cut set in the original graph.

Probabilistic Analysis:

Lemma

$Pr(\textit{no edge of } C \textit{ is contracted}) \geq \frac{2}{n(n-1)}$.

What's the probability space? It's a product of spaces corresponding to rounds of the loop.

The probability space at step i depends on the probability space at step $i-1$.

Conditional Probabilities

Definition

The conditional probability that event E_1 occurs given that event E_2 occurs is

$$\Pr(E_1 \mid E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)}.$$

The conditional probability is only well-defined if $\Pr(E_2) > 0$.

By conditioning on E_2 we restrict the sample space to the set E_2. Thus we are interested in $Pr(E_1 \cap E_2)$ "normalized" by $Pr(E_2)$.
A condition E_2 defines a new sample space, with a new probability function $P(\cdot \mid E_2)$

Let $E_i =$ "the edge contracted in iteration i is not in C."
Let $F_i = \cap_{j=1}^{i} E_j =$ "no edge of C was contracted in the first i iterations".

Since the minimum cut-set has k edges, all vertices have degree $\geq k$, and the graph has $\geq nk/2$ edges.

There are at least $nk/2$ edges in the graph, k edges are in C.
Thus, $Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}$.

Conditioning on E_1, after the first vertex contraction we are left with an $n-1$ node graph, with minimum cut set, and minimum degree $\geq k$. The new graph has at least $k(n-1)/2$ edges, thus $Pr(E_2 \mid F_1) \geq 1 - \frac{k}{k(n-1)/2} \geq 1 - \frac{2}{n-1}$.

Similarly, $Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$.

How do we combine $Pr(E_i \mid F_{i-1})$'s to compute F_{n-1}?

Useful identities:

$$Pr(A \mid B) \;=\; \frac{Pr(A \cap B)}{Pr(B)}$$

$$Pr(A \cap B) = Pr(A \mid B)Pr(B)$$

$$Pr(A \cap B \cap C) = Pr(A \mid B \cap C)Pr(B \cap C)$$

$$= Pr(A \mid B \cap C)Pr(B \mid C)Pr(C)$$

Let $E_1, ..., E_n$ be a sequence of events. Let $F_i = \bigcap_{j=1}^{i} E_i$

$$Pr(F_n) = Pr(E_n \mid F_{n-1})Pr(F_{n-1}) =$$

$$Pr(E_n \mid F_{n-1})Pr(E_{n-1} \mid F_{n-2})....P(E_2 \mid F_1)Pr(F_1)$$

We need to compute

$$Pr(F_{n-2}) = Pr(\cap_{j=1}^{n-2} E_j)$$

We have

$$Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}$$

and

$$Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}.$$

$$Pr(F_{n-2}) = Pr(E_{n-2} \cap F_{n-3}) = Pr(E_{n-2} \mid F_{n-3})Pr(F_{n-3}) =$$

$$Pr(E_{n-2} \mid F_{n-3})Pr(E_{n-3} \mid F_{n-4}) \ldots . Pr(E_2 \mid F_1)Pr(F_1) =$$

$$Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j \mid F_{j-1})$$

The probability that the algorithm computes the minimum cut-set is

$$Pr(F_{n-2}) = Pr(\cap_{j=1}^{n-2} E_j) = Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j \mid F_{j-1})$$

$$\geq \Pi_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \Pi_{i=1}^{n-2} \left(\frac{n-i-1}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \cdots$$

$$\frac{2}{n(n-1)}.$$

Theorem

The algorithm outputs a min-cut edge-set with probability $\geq \frac{2}{n(n-1)}$.

Theorem

Assume that we run the randomized min-cut algorithm $n(n-1)\log n$ *times and output the minimum size cut-set found in all the iterations. The probability that the output is not a min-cut set is bounded by* $\frac{1}{n^2}$.

Proof.

The algorithm has a one side error: the output is never smaller than the min-cut value.

The probability that C is not the output of any of the $n(n-1)\log n$ runs is (using independence)

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\log n} \leq e^{-2\log n} = \frac{1}{n^2}.$$

$\square$

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\log n} \leq e^{-2\log n} = \frac{1}{n^2}.$$

The Taylor series expansion of e^{-x} gives

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \ldots\ldots$$

Thus, for $x < 1$,

$$1 - x \leq e^{-x}.$$

Theorem

1. *The algorithm outputs a min-cut edge set with probability* $\geq \frac{2}{n(n-1)}$.
2. *The smallet output in* $O(n^2 \log n)$ *iterations of the algorithm gives a correct answer with probability* $1 - 1/n^2$.