

A Modern Approach to Teaching an Introduction to Optimization

A Modern Approach to Teaching an Introduction to Optimization

Warren B. Powell

Professor Emeritus, Princeton University
Chief Innovation Officer, Optimal Dynamics
USA
wbpowell328@gmail.com

now

the essence of knowledge

Boston — Delft



Documents

Document Sharing Hub

Published, sold and distributed by:

now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:

now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

Warren B. Powell. *A Modern Approach to Teaching an Introduction to Optimization*, pp. 1–119, 2024.

ISBN: 978-1-63828-321-8
© 2024 Warren B. Powell

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Preface

“Optimization” is widely taught in departments such as operations research, industrial engineering, and sometimes applied math, as focusing on complex, multidimensional (and often very high-dimensional) problems that can be formulated as linear, nonlinear or integer programs. Introductory courses are often centered on linear programming, the simplex algorithm and duality theory.

“Optimization” should be the study of making good decisions, and should start with the simplest (but nontrivial) decisions that are familiar to every student. Linear programs solve a very tiny fraction of decision problems, even in areas such as business where linear programming is often taught. I will note that almost no-one in business without formal training in linear programming has even heard the term. More distressingly, only a fraction of undergraduates or masters students who take linear programming ever solve a linear program (even with a package). And no-one outside a tiny core of specialists has ever programmed the simplex algorithm.

This book outlines a new way of teaching optimization that starts with some basic machine learning problems that are very familiar today given the attention that “AI” has attracted. Few people recognize that machine learning is actually solving a stochastic optimization problem by assuming we are given a training dataset. I leverage this idea to introduce students to some simple sequential decision problems

that involve decisions that everyone faces. We make these decisions using methods (policies) that range from simple, parameterized rules or functions that can be optimized exactly as we solve our machine learning problem.

I then present a series of topics that progress from simple problems that are familiar to everyone, eventually reaching topics such as linear, integer, and nonlinear programming. However, I minimize the attention given to the design of algorithms given that there are widely available packages (it will be the rare beginning student who ever transitions to an algorithmic designer). Instead, I focus on modeling and evaluating the solution which is typically done in the presence of uncertainty.

This book is aimed at faculty who are already teaching an introductory optimization course, or who have a background in optimization and are designing an optimization course. It is also useful to anyone with conventional training in optimization, since it will show you how to think about optimization differently. The presentation consists of a set of topics to guide the design of lectures, leaving considerable flexibility in terms of how much emphasis is placed on individual topics.

The presentation starts with sequential problems, since these are the simplest nontrivial decision problems that are most familiar to students. It then transitions to classical static optimization problems (linear, integer and nonlinear programming) but in each case we show how static optimization models can often be viewed as methods for making decisions in a sequential setting.

Contents

1	Introduction	1
2	Audience	4
2.1	Academic Departments	4
2.2	Students	6
2.3	Faculty	7
2.4	Students/Professionals with Prior Optimization Training	7
3	Course Outline	8
4	Readings	16
5	Lectures	18
	Topic 0: Applications	18
	Topic 1: Machine Learning	21
	1.1 Linear Models	21
	1.2 Nonlinear Models	22
	1.3 Neural Networks (Optional)	25
	1.4 Teaching Notes for Machine Learning	26
	Topic 2: Sequential Decision Problems I	27
	2.1 Introduction to Sequential Decision Problems	27

2.2	Asset Selling	31
2.3	Inventory Planning	32
2.4	Teaching Notes	34
Topic 3: Adaptive Optimization – The Newsvendor Problem		35
Topic 4: Optimal Learning – Finding the Best Treatment		38
Topic 5: Shortest Path Problems		42
5.1	Static Shortest Paths	42
5.2	Dynamic Shortest Paths	43
Topic 6: General Concepts		47
6.1	Modeling Static Optimization Problems	47
6.2	Modeling Sequential Decision Problems	50
6.3	Designing Policies	55
6.4	Evaluating Policies	58
Topic 7: Linear Programming		61
7.1	As a Static Problem – The Simplex Algorithm I	62
7.2	The Simplex Algorithm II – with the Matrix Linear Algebra	68
7.3	As a Policy for a Dynamic Problem	79
Topic 8: Dynamic Inventory Problems – Energy Storage		85
Topic 9: Integer Programming		90
9.1	Static Facility Location	90
9.2	Types of Integer Programs	93
Topic 10: Dynamic Facility Location		96
10.1	Notation	96
10.2	Single-period Model with Uncertain Demands	98
10.3	Evaluating the Policy for a Multiperiod Problem	102
10.4	Alternative Facility Location Policies	105
Topic 11: Nonlinear Programming		110
11.1	Static Portfolio Optimization	110
11.2	Dynamic Portfolio Optimization	113

Index

116

1

Introduction

Making decisions is a universal human activity, something we all have done since we are born. Making good decisions is how we perform better, whether we are running a business, managing a health or energy system, designing and controlling a supply chain, inventing new products and materials, or creating new drugs. Optimization, I claim, is the science of making the best decisions that we can.

The academic community has largely limited the scope of optimization to pet methods for different communities. Industrial engineering and operations research equate optimization with linear programming (as a starter), progressing into even more specialized fields such as non-linear and integer programming. Dynamic programming is considered a very advanced topic, rarely taught at the undergraduate level. Faculty in engineering (mechanical, electrical, chemical) and economics focus on the field of optimal control, a close cousin of dynamic programming, typically using fairly advanced mathematics. Computer scientists will either study combinatorics (a close cousin of integer programming) or, more recently, reinforcement learning (a cousin of dynamic programming). All of these approaches to optimization are typically taught at a fairly advanced level.

I am going to suggest a new way to teach introductory optimization to undergraduates and masters that is far more useful and addresses the modeling and algorithmic challenges that arise in decision problems that occur in practice. To do this, I am going to take on two of the titans of optimization: George Dantzig and Richard Bellman. In this course, we will still teach linear programming, but it is demoted to a method that is used only for a small number of problems, and while I will illustrate the simplex algorithm using a network problem (where the steps of the simplex algorithm can be described visually), I do not drag students through the simplex algorithm for general linear programs.

By contrast, we will pay considerably more attention to sequential decision problems (also known as dynamic programs), but we will largely ignore Bellman's equation (or Hamilton-Jacobi equations). Also, we note that the origins of linear programming were in the context of sequential decision problems, and we will make the distinction between linear programming as a static problem, and as a policy for sequential decision problems. We do the same for integer and nonlinear programming.

My approach will be to start with the simplest nontrivial problems, and progress to more complex settings. We start with some basic machine learning problems (with linear and nonlinear models) that lay the foundation for modeling and solving sequential decision problems (SDPs). SDPs, which usually involve making relatively simple decisions over time, are problems that are pervasive in business, economics, engineering, the sciences, and even everyday life. SDPs include problems where the decision may be binary (which web page design to use, when to stop and sell an asset), discrete (choosing the best catalyst, drug, product, path, supplier, employee, . . .), and scalar continuous (finding the best price, dosage, temperature, budget). We make decisions using the three classes of policies that are widely used in practice (albeit in an ad hoc way), but show students how to properly model and tune these policies. And we do not use Bellman's equation other than a brief illustration in a shortest path problem. Bellman requires a level of sophistication for both modeling and algorithmic work that is not suitable for an introductory course, and it is only useful for a very narrow subset of problems.

In the process of using sequential decision problems, we are going to have to introduce the dimension of modeling sequential information processes. While this potentially opens a Pandora's box into the complex arena of stochastic modeling, we avoid this by exclusively working with random samples, just as is done with machine learning where a training dataset is a sample of the random observations.

The course is aimed at undergraduates (or masters) using a minimal amount of calculus or linear algebra. We use very simple Monte Carlo simulation, but do not require a course in probability or statistics. There are many opportunities for a faculty member to adjust the examples and scope of the material to their own students. However, we urge faculty to resist the typical style of teaching this material which emphasizes the methods that have a strong theoretical foundation. This material can be taught at a very advanced level, but this course focuses on teaching students how to *think* about making decisions.

2

Audience

This book was originally written to help schools teaching introductory optimization courses adjust their pedagogical approach to a modern style that is much more useful and relevant to students. For this purpose, I discuss below the audience from three perspectives: what departments can use this approach, the students that I am targeting, and the faculty who might be interested in teaching this.

A fourth audience is people who have already taken a traditional course in (typically deterministic) optimization. Simply reading these notes will provide a different perspective that draws on the skills you have already learned.

2.1 Academic Departments

I think the style of this course can be used in any department that involves making quantifiable decisions. This includes engineering (all departments), the physical sciences (laboratories are full of sequential decision problems), the social sciences, economics, business schools, politics, and psychology.

“Optimization” (linear, nonlinear and integer programming) is traditionally taught in operations research, industrial engineering and

applied math. In engineering there is much more emphasis on control theory (a form of sequential decision problem), while economics (along with advanced courses in OR and IE) will emphasize dynamic programming. Computer science for the past decade has taught reinforcement learning, a field that addresses “Markov decision processes” which is just a different form of control problem. These courses all tend to be taught with a moderate to high level of mathematical sophistication, ignoring the reality that virtually everyone needs to solve sequential decision problems.

I will note that none of these classical courses on optimization traditionally deal with what I call “optimal learning” problems, which are problems where the decision controls what information to collect to improve your understanding of a process so you can make better decisions in the future. Optimal learning problems arise in laboratory experimentation (either physical experiments or computer simulations) and a host of field settings (what is the best product to recommend, what is the best price to charge, what is the best medical treatment, what process to use to make a material, . . .). I taught a course called Optimal Learning for 10 years at Princeton at the undergraduate level. The course was quite popular, and deals with problems that are much more familiar to our students than linear programs.

The course I am proposing covers both static problems (such as linear programs) as well as sequential decision problems (which includes dynamic programming, optimal control and reinforcement learning). Topics like linear programming can be ignored (for fields where these problems simply do not arise), covered very briefly (a single lecture), or extensively (some courses will spend 4–6 weeks just on linear programming).

The real novelty of our teaching style is how we approach sequential decision problems, which arise in virtually every problem domain, and yet are often ignored in introductory courses in optimization. Our approach to teaching this topic emphasizes practical solution methods that reflect what is used in practice but placed in a framework that formalizes the evaluation and tuning of policies.

Our point of departure from traditional approaches for teaching sequential decision problems is that we largely ignore methods based

on solving Bellman's equation (known as Hamilton-Jacobi equations in the optimal control field). Methods based on HJB equations are mathematically elegant, but only apply to a very narrow set of problems, which is a reason why people routinely make decisions over time who have never even heard of HJB equations.

[Side note: I spent 20 years of my career, and wrote a popular 500-page book on approximate dynamic programming (also known as reinforcement learning) which is a field that focuses on methods for solving HJB equations approximately. My conclusion that this approach has limited practical value is based on many years of research.]

2.2 Students

The course is aimed at undergraduates or masters students with no prior training in optimization. The following skills will be useful:

- Students will need some calculus, but only at the level of understanding a derivative and gradient (which, to be honest, can be presented very quickly). When we do use derivatives/gradients, these can often be estimated numerically instead of using the analytical formulas stressed in introductory calculus courses.
- We use a very modest amount of linear algebra – much less than traditional courses in optimization. For example, there are perhaps two places where we use the concept of an inverse of a matrix. Students without any prior training in linear algebra could be taught the basic idea of a vector and matrix in a short tutorial session.
- We will occasionally use some very basic concepts from statistics, but we will do this in a way that does not require a prior course in statistics. For example, it is very easy to introduce a student to a mean and variance. In this course there is no need for familiarity with different probability distributions.

2.3 Faculty

I am assuming that the faculty teaching this course are already trained in the core fields of linear, nonlinear and integer programming if you wish to cover this material, but there are entire fields (such as computer science) where students are typically not introduced to linear programming. These topics are covered, but not nearly in the depth that the faculty member might remember from their own training. Times have changed, and there is no longer a need to teach, for example, algorithms for solving linear programs (packages will be used).

2.4 Students/Professionals with Prior Optimization Training

Reading (even skimming) these notes by someone who has already had a course in (deterministic) optimization should change how they think the solution of the optimization model should be evaluated. This means appreciating that in many (most? almost all?) settings, deterministic optimization models are actually policies (methods for making decisions) that need to be evaluated over time, under uncertainty. This then opens a door to improving the solution of their deterministic model in terms of real-world performance.

3

Course Outline

This section provides a sketch of the course. The material is divided into 11 topics, which are described in much more depth in Section 5. Here, I summarize each of the topics, focusing on the development of key concepts.

The course will transition from the simplest (but nontrivial) decision problems to more complex settings. We will start with basic machine learning problems partly because they are very familiar today, but also because they are well motivated and easy to understand. The machine learning problems will also lay the foundation for how we handle random observations in sequential decision problems which are of fundamental importance (since so many real problems are, in fact, sequential in nature).

The lectures are organized to follow a natural progression from simpler decisions (binary, discrete, continuous scalar) to more complex ones (continuous vectors, integer variables, nonlinear functions). Our emphasis is on formulating optimization models, including the critical (but historically overlooked) problem of understanding how to handle decision problems when they are made sequentially over time. We present algorithms when one or both of the following applies:

- An understanding of the algorithm can help students appreciate the behavior of the solution, even if they never implement an algorithm.
- Students may need to program the algorithm if they are to solve the problem (we limit these to relatively simpler algorithms).

This book presents the course as a series of “topics” that can be adapted to the interests of the faculty member, and the background and interests of the students. Most topics can be presented in 1 to 3 lectures, but some topics can be extended to as many as 6–8 lectures depending on the interests of the faculty member and the skills and background of the students.

Below I will list each topic and describe the key points being covered, emphasizing the transition from simpler to more complex problems. This is a sharp departure from introductory courses in “optimization” that turn out to be courses in “linear programming with extensions.”

- Topic 1 – Machine learning – We start with fitting a linear model to introduce the idea of solving a convex optimization problem exactly. Students should learn the minimum amount of data required to fit a linear model (for example, $n \geq p$) in addition to other conditions on the data. We then transition to nonlinear models and introduce gradient search and the issue of multiple optima (a major topic with neural networks that are so prominent today). Also note that with nonlinear estimation, we no longer require $n \geq p$, implying that we can fit, say, a neural network with 100 million parameters with a single datapoint.

A particularly important piece of pedagogy in Topic 1 is the idea (often overlooked) that estimation problems are, in fact, stochastic optimization problems, where random variables are replaced with sampled observations. This is going to set the style for handling uncertainty which will run throughout our handling of sequential decision problems. We note that this style allows students to do “stochastic optimization” without any training in stochastic optimization, probability, or even a course in statistics.

- Topic 2 – Sequential decision problems – We next introduce the concept of a sequential decision problem, and then use some important and visible problems to illustrate how to model these. Most important is the concept of a *policy* which is a method (that is, a function) for making decisions, that is controlled by tunable parameters. This closely parallels fitting a model to data (as we do in Topic 1). We start with a basic example for selling an asset that uses historical data, and then transition to an inventory problem where we need to randomly generate observations. This is done using a very basic introduction to Monte Carlo sampling.

The policies we introduce are both forms of policy function approximations (PFAs) which are widely used by individuals as well as corporations. PFAs help us create a natural bridge to estimating functions in machine learning, but students also learn how to set up an objective function for sequential decision problems. This skill will stay with us as we progress to more complex decision problems.

- Topic 3 – Adaptive optimization – In this topic I use the newsvendor problem to introduce the idea of using sampled information to compute a gradient. This is widely known as a stochastic gradient in the literature, but the gradient is based on a sample, which means we are taking the derivative of a deterministic function. This problem will require generating random variables dynamically rather than creating a sample in advance as we did in Topic 2. It is important to recognize that while the newsvendor problem is perhaps the most widely studied stochastic optimization problem, the algorithm is quite simple, and outside of generating random samples, all of the steps use deterministic methods.
- Topic 4 – Optimal learning – This is a topic where the optimization problem is making decisions of what to observe, such as how a patient responds to a drug, how many clicks a website attracts, and the market demand for a product at a particular (discretized) price (applications of this model are endless, and familiar to everyone). We use a policy called interval estimation (a form

of upper confidence bounding) that is very popular with tech companies (e.g. Google and Facebook) for maximizing ad-clicks. Optimal learning will also play a role any time we need to do parameter tuning, which will turn out to be a common type of optimization problem.

Upper confidence bounding policies represent a form of “cost function approximation” (CFA) where the policy for making decisions has an imbedded optimization problem. For UCB policies, this optimization problem involves a simple sort, but it introduces the idea of a policy that involves solving an optimization problem to make a decision. This means we have an optimization problem which requires sorting a set of estimates within a larger optimization problem of tuning the parameter in the UCB policy.

- Topic 5 – Shortest path problems – Here we introduce our first nontrivial static, deterministic optimization problem which is also a very special form of linear program (but that comes later). This is the only time we use Bellman’s equation in the course, although there are problems (in Topic 10) where we could draw on Bellman again.

After presenting the model and algorithm for a deterministic, static shortest path problem, we then transition to show how this can be used in a dynamic setting, as would happen if we are modeling a path through a dynamic network. We show how to model this problem, and then show how to create a classic deterministic dynamic lookahead approximation (deterministic DLA). We show how to evaluate the shortest path problem as a policy, and how to parameterize it so that it works better in a stochastic, dynamic environment.

We are going to copy this setting as we move into more general optimization problems. We will start by presenting a basic, static optimization problem (this could be a linear, nonlinear or integer program), and then show how it is often used as a policy in a dynamic setting. In my experience, the vast majority of “optimization problems” are actually policies used in a sequential problem

setting, something that is typically overlooked in standard texts on optimization.

- Topic 6 – General concepts – We pause at this point to discuss two important dimensions of sequential decision problems:
 - Classes of policies – So far, we have illustrated four ways of making decisions, each of which come from the four classes of policies. These four classes cover every possible method that we might use to make decisions, including any method people are already using.
 - Evaluating policies – The biggest difference between people who make decisions in an ad hoc way versus someone with formal training is their understanding of the concept of a policy, and how to evaluate it. In this topic (typically a single lecture) we start by reviewing how we have evaluated policies in topics 2 – 5. We then list different ways of evaluating policies such as cumulative reward for online learning, and final reward for learning in a lab. We also differentiate between expected performance versus risk. While the academic literature deals with risk with a considerable amount of mathematical sophistication, we are going to show students how to model risk in a way that can be easily computed in a spreadsheet.
- Topic 7 – Linear programming – Here is where we introduce linear programming. This can be done in a single lecture (which I recommend for an introductory optimization course) or expanded given the time available, interests of the students, and the interests of the faculty member teaching the course.

Our preferred style for an introductory course is to teach the idea of a linear program and then transition to the understanding that “algorithms exist” for solving it. An in-depth presentation of the simplex algorithm is simply not appropriate at this stage, since no-one is ever going to implement their own simplex algorithm. Modern implementations of the simplex algorithm use a

variety of sophisticated strategies to improve performance; these strategies are never discussed in textbook treatments of the simplex algorithm, so it is not clear what a student is learning from these streamlined presentations. In addition, production implementations might combine strategies such as dual simplex or even interior point methods. This material is simply not appropriate for an introductory course.

This said, we illustrate the simplex algorithm graphically using a network problem which helps students understand, in a highly visual way, the concept of a basis, pivoting, and most important, dual variables. This can be done without any linear algebra, but we do have a section where we present the simplex algorithm (for a network problem) both graphically, and then algebraically. We leave it to the instructor to decide which presentation best suits their students.

We begin by presenting linear programming as the solution to a static problem, but we then transition to using linear programming as a policy for sequential decision problems. We suspect that most linear programming applications arise in the context of sequential decision problems (and this is certainly true of the original motivating applications used by George Dantzig). I think the recognition that linear programs are often used as policies for sequential decision problems is one of the great failures of the math programming community. Topic 8 illustrates how a deterministic linear program might be used in a sequential inventory problem.

- Topic 8 – Dynamic inventory problem – Here we are going to copy what we did for our dynamic shortest path problem but use the context of an energy storage problem in a highly dynamic setting with rolling forecasts (a topic that has been completely overlooked in the operations research literature). This requires solving a series of simple linear programs, even though the decision at a point in time is a scalar (we get the LP because we are optimizing over a planning horizon, which means our decision variable is now a vector). The lookahead LP will be parameterized to help mitigate the errors in the rolling forecasts, and we will show

that this produces a much better result than using typical point forecasts. The challenge, as always, will be the tuning, a problem we first saw in the machine learning problem in Topic 1. We will suggest a strategy that is fairly easy to implement.

- Topic 9 – Integer programming – Here we introduce the idea of integer variables in the context of a facility location problem. As with our shortest path problem, we will start with a simple, static facility location problem. Then, Topic 10 shows how the static model can be used as a policy in a fully sequential problem.

Optimization books tend to become drawn into the fairly sophisticated algorithms required to solve integer programs. However, since year 2000, commercial packages have conquered wide classes of even very large integer programs, although some care has to be used since there is a wide range of integer programming problems, and some still require specialized algorithms. The best packages (such as Gurobi and Cplex) can be dramatically better than free software that students can download over the internet. As with algorithms for linear programs, teaching algorithms for integer programs is pointless for an introductory course – these algorithms are quite sophisticated and no-one today would implement their own. However, it is important for students to be able to recognize which types of integer programming problems are likely to be solvable with a general purpose package.

- Topic 10 – Dynamic facility location – As we did with linear programming, we start by presenting a static integer programming problem using facility location, and then extend it here to a dynamic setting. We start by making the case that any facility location problem would have to be implemented in a stochastic environment. We separate the decision of where to locate facilities, which is made using forecasted demands, and the “real world” decisions of how to meet demands which are revealed after we make the decision to locate facilities.

We then recognize that decisions to locate facilities are themselves decisions that are made over time, in the presence of the

uncertainties about demands. We illustrate a few strategies for solving the dynamic facility location problem.

- Topic 11 – Nonlinear programming – We have already seen nonlinear programming in topic 1 when we fitted a nonlinear model, but here we are going to address this rich topic in more depth. As with linear and integer programming, we are going to present nonlinear programming in two stages: first as a static problem, and then as a policy in a fully sequential problem. Nonlinear programming is a rich topic that can be introduced in a single lecture but can span an entire course. It is up to the professor to decide how much time to spend on this topic given the interests of the students.

We are going to introduce students to a quadratic programming problem that arises when optimizing investments over a portfolio. We will first introduce this problem in its classical formulation as a static problem, and then transition to solving it sequentially over time, treating it as a policy in a fully sequential decision problem (based on actual practice on Wall St.). This will be a sophisticated (but very real) extension of the asset selling problem we introduced in Topic 2.

4

Readings

Many of the topics are organized around sequential decision problems presented in:

Warren Powell, *Sequential Decision Analytics and Modeling*, Now Publishers, 2022 (available for free download from <https://tinyurl.com/sdamodeling>). Below I refer to this as “SDAM.”

Readings from SDAM are indicated at the beginning of each topic (or subtopic).

Occasionally I refer to material in my graduate-level book:

Warren Powell, *Reinforcement Learning and Stochastic Optimization*, Wiley, 2022 (see <https://tinyurl.com/RLandSO/> for an overview). Below I refer to this as “RLSO.”

RLSO is not appropriate for an introductory course such as this, but I recommend that the instructor have a copy of the book.

There are blocks of material on mature topics like linear, integer, and nonlinear programming. I assume that any professor teaching a course in optimization will already have a favorite book they like to use for these topics. We encourage, for an introductory course like this, putting more emphasis on describing what these problems are and how they are used, with less emphasis on algorithms, especially when these are widely available in packages.

5

Lectures

In this section I sketch out a sequence of topics that steadily transition from relatively simple decision problems to more complex ones. There is considerable flexibility in terms of how much time is spent on each topic. For example, linear programming can easily be taught in a single lecture (basically defining what a linear program is), but it can also fill half a course. There are also topics on integer programming and nonlinear programming which can also be taught in a single lecture, but there are entire (graduate level) courses dedicated to each of these topics. In an introductory course, I think students should be introduced to these topics, but in a modest way.

Topic 0: Applications

It always helps to start an introductory course such as this with a series of applications. This will be very dependent on the department where the course is being taught. Below I give some illustrative applications that I used when I was teaching this material.

- Machine learning problems – These require optimizing parameters to make a model fit a training dataset.

- Management of physical resources – Physical resources might be:
 - People (hiring, firing, training, moving)
 - Equipment (trucks, drones, robots, medical equipment, ...)
 - Facilities (building, leasing, closing, resizing)
 - Product (planning inventories for retail sale, or parts used in manufacturing)
- Management of financial resources – These include
 - Planning cash reserves
 - Making investments
 - Arranging different financial instruments (loans, insurance contracts, ...)
 - Setting budgets
- Information acquisition and communication
 - Running experiments in a lab or the field
 - Running medical tests
 - Sending/sharing information about the status of a system
- Finding the best ways of making decisions
 - Choosing the best method for making decisions
 - Tuning parameters used by a method

These decision problems can come in two forms:

- Static problems – These are problems we solve once and then use the solution
- Sequential decision problems – These are decisions that are made repeatedly over time as new information is arriving.

Sequential decision problems are quite rich, and typically involve relatively simple decisions. However, the sequential nature, and in particular the flow of new information, can introduce significant complexities.

We are going to steer around these complexities and show how to model and solve problems that everyone encounters in their own personal activities, or any of a wide range of problems in business, engineering and the sciences.

The powerpoint slides I used in my first lecture can be downloaded by going to <https://tinyurl.com/RLSOcourses/>. Scroll down to the heading “Undergraduate/masters course in sequential decision analytics” and then scroll down to “Lecture 1” and download the slides. However, it is very important that these applications be chosen based on the interests of the students.

Topic 1: Machine Learning

One of the most visible (and accessible) optimization problems today arises in machine learning, where we have to find the best fit of a model to a training dataset (this is also known as supervised machine learning).

Below we describe the optimization problems and solution methods that arise when we are fitting linear models and nonlinear models. Each setting will allow us to illustrate different optimization strategies, from finding an optimal solution analytically with linear models to using a derivative-based search algorithm for nonlinear models. We will also learn some properties of optimal solutions along with the necessary conditions for optimality in each setting.

1.1 Linear Models

We are going to start by assuming we have a basic training dataset that we can write as

$$(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N).$$

We assume that we have a model of the form

$$y = f(x|\theta) = \sum_{n=1}^N \theta_f \phi_f(x^n), \quad (1.1)$$

where $\phi_f(x)$ is known as a “feature” which is some function of the input data x . Given the features (chosen manually) our optimization problem is given by

$$\min_{\theta} \sum_{n=1}^N (y^n - f(x^n|\theta))^2. \quad (1.2)$$

I would start by deriving the well-known normal equations, given by

$$\theta^* = [X^T X]^{-1} X^T Y,$$

where X is the *design matrix* given by

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_p^1 \\ x_1^2 & x_2^2 & \dots & x_p^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^n & x_2^n & \dots & x_p^n \end{bmatrix},$$

and Y is our vector of observations (also called responses or labels)

$$Y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{bmatrix}.$$

This should be a warmup using their linear algebra and setting up a quadratic optimization problem that can be solved analytically.

Bring out that we need n observations $\geq p$ (number of parameters), and that the data has to be well behaved (so that the $[X^T X]$ matrix is invertible). It is easy to illustrate this with a simple example – fitting a line through two datapoints.

In the teaching notes below, we argue that the optimization problem in (1.2) is actually a stochastic optimization problem. It looks deterministic here (and it is) because we are working with a sample of the random variables y . The sequence $(y^1, \dots, y^N)$ is actually a sample of the random observations. We are going to see later that we can turn a lot of stochastic optimization problems into problems requiring deterministic methods by using samples, so this starter problem not only introduces an important application (machine learning), it is setting the stage for how we are going to solve a wide range of sequential problems that involve random information.

1.2 Nonlinear Models

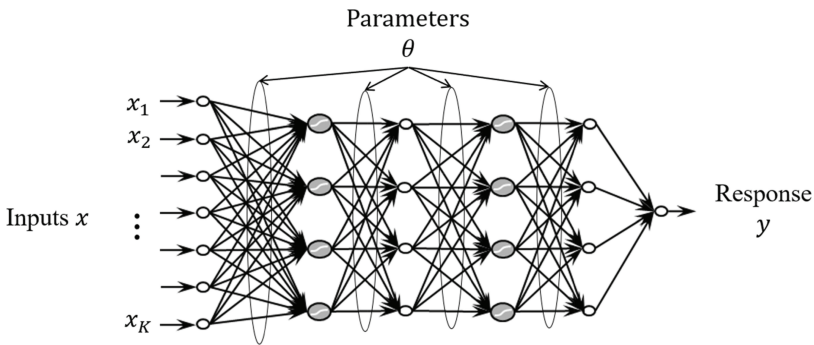
Next we transition from a linear model (where $f(x|\theta)$ is linear in the parameters) to a nonlinear model. One example is a logistic regression such as

$$f(x|\theta) = \frac{e^{\theta_0 + \theta_1 x}}{1 + e^{\theta_0 + \theta_1 x}},$$

or

$$f(x|\theta) = \begin{cases} -1 & x < \theta_1 \\ 0 & \theta_1 \leq x \leq \theta_2 \\ +1 & x > \theta_2 \end{cases}.$$

Or, our nonlinear model could be a neural network with millions (or billions) of parameters θ that looks like



We now have the optimization problem

$$\min_{\theta} g(\theta) = \sum_{n=1}^N (y^n - f(x^n|\theta))^2. \quad (1.3)$$

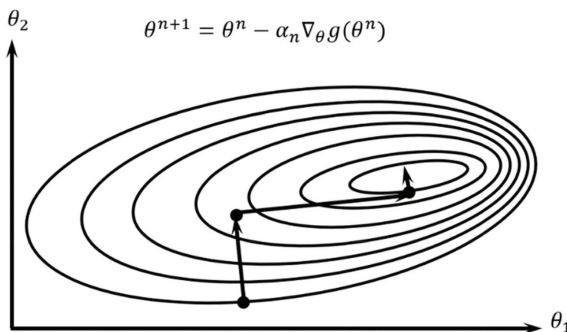
We can solve this using a gradient-based search algorithm that looks like:

$$\theta^{n+1} = \theta^n - \alpha_n \nabla_x g(\theta^n). \quad (1.4)$$

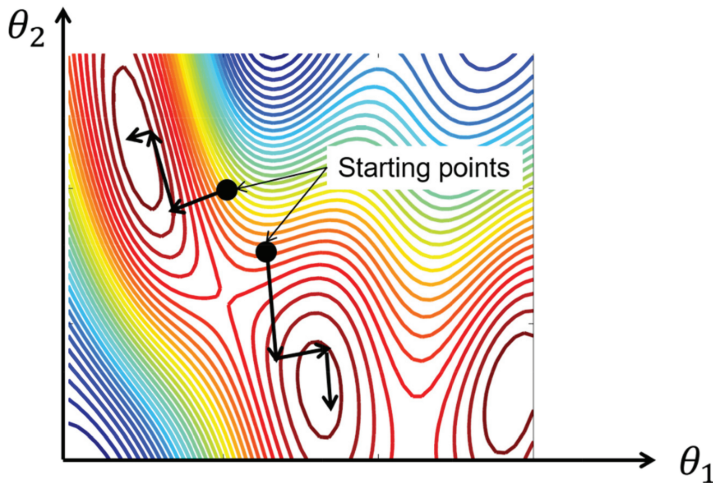
(Note that we use the negative gradient because we are minimizing). You need to talk the students through the process of finding the gradient (and possibly explaining what this is). One key step is finding the stepsize α_n which is done by solving the one-dimensional search problem:

$$\alpha_n = \operatorname{argmin}_{\alpha > 0} g(\theta^n - \alpha_n \nabla_x g(\theta^n)). \quad (1.5)$$

The figure below illustrates the search process.



One issue that often arises is that the function $g(x|\theta)$ may have local minima (illustrated below), which means your gradient algorithm may produce different optimal depending on the starting point.



Also – I would note that while we need n (the number of observations) $\geq p$ (the dimensionality of θ) to use the normal equations for our linear model (in Topic 1), we no longer require this for the nonlinear model. We can apply the gradient algorithm for any value of n , even if it is smaller (and potentially much smaller) than p . Students need to understand that just because the algorithm returns an estimate of the best value of θ , that does not mean that it is guaranteed to be a good value that will work well on future datasets.

You can start by illustrating it using a linear model, and then show how to use it for a nonlinear model. For example, fit a logistic regression to predict demand as a function of price, or the probability of winning a bid for placing an ad on Google or Facebook. Then extend to a simple neural network. Be sure to highlight the presence of multiple optima and the need to use multiple starting points. Then, show that you get an answer even when $n < p$, an issue that becomes important with deep neural networks.

This will create a basis for later discussing a well-known issue with neural networks (all students will have heard of this as “AI”) which use models where $p \gg n$. Students will also learn that there is not a unique

solution when we optimize parameters for nonlinear models. Finally, I would also make the point that while linear models require $n \geq p$, nonlinear models do not. We can fit a neural network with 100 million parameters with a single datapoint (!).

1.3 Neural Networks (Optional)

A special kind of nonlinear model is a neural network. Given the visibility of this technology, an instructor may wish to introduce students to the basic idea of a neural network, since the calculations are relatively simple. It is easy to show how to construct a basic neural network, and how to take a vector of inputs and translate it to an output (or set of outputs). Section 3.9.3 of RLSO describes how to compute the output of a neural network through a simple forward pass.

You can then jump to section 5.5 of RLSO if you wish to show how to compute the gradient of a neural network with respect to the weights on each link (these are the tunable parameters). Key to neural networks is that these derivatives are easy to compute. Given the interest in deep neural networks, you can show how these calculations can be done in parallel, which is why chips such as those by Nvidia (which are designed for massively parallel computation, originally for the graphics in video games) are so popular.

I would also point out that there are packages such as TensorFlow that do these calculations very efficiently. However the calculations are performed, the core ideas are the same as what we illustrated using the logistic regression example in section 1.2. Remember – the idea in an introductory course is not to prepare students to actually do research on this topic; it is only to introduce them to important optimization problems and how we go about solving them. For example, the issue of multiple local optimal solutions that we introduced in our section on nonlinear models above is important, since it helps to understand that the “optimal” solution we obtain is not truly optimal, and that small changes in inputs might result in finding a different local solution.

Also – remember that while we require $n \geq p$ in a linear model, we have no such requirement for nonlinear models, including neural networks. We might have a neural network with 100 million parameters,

but we will still get a number if we try to train it with just one data point. While this seems silly, it is quite common to fit neural networks where the number of observations (that is, the size of the training dataset) is quite a bit smaller than the number of parameters.

I don't personally encourage including this material, but it fits very nicely here, and might go a long way to attracting student interest.

1.4 Teaching Notes for Machine Learning

There are some key points that should be brought out in this topic:

- The linear model is a nice opportunity to remind students of some basic linear algebra when deriving the normal equations.
- Be sure to bring out the requirement that $[X^T X]$, where X is the “design matrix” of data, must be invertible, which requires that $n \geq p$. I suggest illustrating with the problem of fitting a line to a single data point, and then to two data points where x is the same for each one.
- The objective functions (equations (1.2) and (1.3)) look like deterministic optimization problems, but they are not. Fitting a function (linear or nonlinear) to data is actually a stochastic optimization problem which should be written

$$\min_{\theta} \mathbb{E}_X \mathbb{E}_{Y|X} (Y - f(X|\theta))^2. \quad (1.6)$$

Here we view X , the explanatory variables (also known as independent variables or covariates), and the response $Y = f(X|\theta)$, as random variables. We assume we are given a sample of these variables

$$(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N),$$

which we call the training dataset. However, this is just a sample of the random variables X and Y , which turns our stochastic optimization problem (1.6) into a deterministic optimization problem (1.2) or (1.3). We are going to use this technique over and over again in this course to handle virtually any form of uncertainty. The only difference will be in future applications is that we may need a way of generating our own random sample.

Topic 2: Sequential Decision Problems I

The vast majority of all decision problems are sequential decision problems. Later, we are going to motivate harder decision problems that use linear, nonlinear and integer programming, but to start we are going to use simpler problems which are still important as well as challenging.

2.1 Introduction to Sequential Decision Problems

Readings: SDAM Chapter 1 (introduction to sequential decision problems).

A sequential decision problem is any problem that consists of the sequence:

decision, information, decision, information, ...

where each decision receives a contribution or incurs a cost. Sequential decision problems cover an extremely broad class of optimization problems. Most important is that they cover problems that arise in almost any setting: business, health (all kinds), energy, economics, laboratory experiments, field experiments, ..., the list is endless, which means it is possible to illustrate these problems with applications that are suitable to any class.

Decisions are made with methods that we call “policies.” There are two broad strategies for designing policies, and each of these produces two classes, creating four classes of policies:

Strategy 1 – Policy search – This is where we identify a class of functions for making decisions, and then search for the best function that works well over time. The two classes of policies in this strategy are:

1. Policy function approximations (PFAs) – These are analytical functions that take what we know to determine what decision to make now. Examples are order-up-to policies for inventory, or buy low, sell high policies in finance.
2. Cost function approximations (CFAs) – These are simplified (usually deterministic) optimization models that have been parameterized to work well under uncertainty. We will see CFAs when

we introduce optimal learning (Topic 4). We will also see these when we introduce linear, integer and nonlinear programming in topics 7–11.

Strategy 2 – Lookahead approximations – We estimate the value of a decision by combining the immediate cost or reward of a decision plus some approximation of downstream costs and rewards from the initial decision. This strategy can be divided into two classes:

3. Value function approximations (VFAs) – Here we find the decision that optimize the immediate cost or reward plus an approximate value of the state that the decision takes us to. These are the only policies that use Bellman’s equation (Hamilton-Jacobi if you are a controls person).
4. Direct lookahead approximations (DLAs) – Finally we optimize the immediate cost or reward plus an estimate of downstream costs or rewards computed by solving an approximate model of the future.

Note that sequential decision problems arise throughout human activities. PFAs are the simplest class and are the most widely used. Most important, these are parameterized functions, just like the parametric models in statistics that we saw in Topic 2.

Below is the slide I use to present the elements of a sequential decision problem. It illustrates the notation of states S_t (what we know at time t), decisions x_t (what decision we choose from a set of feasible decisions), and the exogenous information W_{t+1} that we learn only after we make the decision x_t . Decisions are made with a method (policy) that we designate as $X^\pi(S_t|\theta)$ that often depends on tunable parameters θ . Also shown is the contribution (if we are maximizing) or cost (if minimizing) $C(S_t, x_t)$ which may depend on information in the state S_t (such as dynamically changing prices or costs) in addition to the decision x_t . The transition function $S^M(S_t, x_t, W_{t+1})$ gives the updated state S_{t+1} given the information in S_t , the decision x_t , and the exogenous information W_{t+1} .

The next slide (below) is one I use to compare machine learning with sequential decisions. The difference between machine learning

Modeling sequential decision problems

Any sequential decision problem can be written:

$(S_0, x_0, W_1, S_1, x_1, W_2, \dots, S_t, x_t, W_{t+1}, S_{t+1}, \dots, S_T)$

What we know (or believe)

What we observe (or learn)

The decision

Each time we make a decision, we receive a contribution $C(S_t, x_t)$.

Decisions are made with a method or *policy* $X^\pi(S_t)$ which we design later.

State variables evolve using a transition function: $S_{t+1} = S^M(S_t, x_t, W_{t+1})$.

The goal is to find the policy that maximizes expected contributions:

$$\max_{\pi} \mathbb{E}\{\sum_{t=0}^T C(S_t, X^\pi(S_t)) | S_0\}$$

and sequential decisions is that with machine learning, you need a training dataset to fit a general-purpose function, whereas sequential decisions (which do not need a training dataset) need a model of the underlying problem (such as the evolution of inventories). It is important to emphasize this difference, since it is often overlooked (especially in discussions of “AI”).

BRIDGING MACHINE LEARNING & SEQUENTIAL DECISIONS

Machine learning

$$\min_{f \in F, \theta \in \Theta'} \frac{1}{N} \sum_{n=1}^N (y^n - f(x^n | \theta))^2$$

Searching over functions

“Big dataset”

Sequential decisions

$$\max_{\pi = (f \in F, \theta \in \Theta')} \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^T C(S_t^n, X^\pi(S_t^n | \theta))$$

$S_{t+1} = S^M(S_t, x_t, W_{t+1})$

Searching over policies

System model

Lookup tables

Parametric models

Neural networks

Nonparametric models

Policy function approximations

Cost function approximations

Value function approximations

Direct lookahead approximations

Analytical functions

Optimization problem

Optimization problem

Optimization problem

There is a wide range of sequential decision problems that will be familiar to students (unlike linear programs). The challenge is that sequential decision problems involve the arrival of new information,

 Documents
Document Sharing Hub

Full document details: IEEE A Modern Approach to Teaching an Introduction to Optimization

which puts even simple decision problems into a class that has been treated under a wide variety of names, with the most common being “dynamic programs.” This will produce gasps of “oh, that is *stochastic optimization*. . . you can’t teach *stochastic optimization* in an introductory optimization course.” For most faculty, “dynamic programming” means teaching Bellman’s equation.

We are not going to go down that path. First, we observe that both of the machine learning problems in Topics 1 and 2 are forms of stochastic optimization problem. We avoid the complexities that would normally arise by tuning our statistical models used a training dataset that we assumed was given to us.

We are going to follow the same strategy for sequential decision problems, emphasizing three types of policies that are widely used, and which students will typically be familiar with. These are:

- PFAs – These are the simplest policies such as buy low, sell high in finance or order-up-to policies for inventory planning. Note that PFAs include every possible model that might be used in machine learning, which might be anything from a linear model to a deep neural network.
- CFAs – These will be simplified (always deterministic) optimization problems that might involve nothing more than a sort. For more complex problems, we will introduce linear programs, and then show how we can parameterize a linear program to make it work well over time. This powerful idea is widely used in practice but has been completely ignored in the academic literature.
- Deterministic DLAs – Google maps is a nice example of a deterministic DLA, since it plans a path to a destination (which requires looking into the future). For problems where we need to plan into the future, the most widely used strategy is to use a deterministic approximation. We *might* use tunable parameters to make these perform better over time, but not always.

Below we are going to illustrate some simple sequential decision problems using PFAs, which require tuning parameters. The approach exactly parallels what we did in Topic 1 for machine learning.

2.2 Asset Selling

Readings: SDAM Chapter 2.

We are going to start with a simple buy-low, sell-high policy in finance, which offers an important simplification to help us get started: the random information (the price at which we can sell our asset) is drawn from history, allowing us to treat it just as we did the observations y^n in our machine learning problems in Topic 1.

We are going to assume that we are given a historical sequence of prices

$$(p_{t-H}, p_{t-H+1}, \dots, p_{t'}, \dots, p_t)$$

where time t' is any point in time in the history where we might sell an asset (assume for simplicity that we can only sell at the end of each day). The price p_t would be the most recent price we have available. A nice feature of this problem is that we can assume that the prices are independent of any decisions that we make.

Next imagine that we are going to use the following policy that determines when to buy, sell or hold a single asset

$$X^{\pi_1}(S_t|\theta) = \begin{cases} +1 & \text{Sell if } p_t > \bar{p}_{t-1} + \theta_1 \\ 0 & \text{Hold if } \bar{p}_{t-1} - \theta_2 \leq p_t \leq \bar{p}_{t-1} + \theta_1 \\ -1 & \text{Buy if } p_t < \bar{p}_{t-1} - \theta_2 \end{cases} \quad (2.1)$$

where

$$\bar{p}_t = .5p_t + .35p_{t-1} + 0.15p_{t-2} \quad (2.2)$$

is a smoothed estimated of prices. The variable S_t , which is called the “state variable” captures all the information we know at time t , that we need to make a decision (that is, compute our policy $X^{\pi_1}(S_t|\theta)$), as well as any other information we might need. For this problem, the state variable consists of

$$S_t = (p_t, p_{t-1}, p_{t-2}).$$

It is easy to see that our policy $X^{\pi_1}(S_t|\theta)$ is one of many possible strategies we could use. This policy has two tunable parameters. To tune the parameters, we need an objective function.