AA203: Optimal and Learning-based Control Combined Course Notes

James Harrison

April 1, 2019

Introduction

These notes accompany the newly revised (Spring 2019) version of AA203 at Stanford. The goal of this new course is to present a unified treatment of optimal control and reinforcement learning (RL), with an emphasis on model-based reinforcement learning. The goal of the instructors is to unify the subjects as much as possible, and to concretize connections between these research communities.

How is this course different from a standard class on Optimal Control? First, we will emphasize practical computational tools for real world optimal control problems, such as model predictive control and sequential convex programming. Beyond this, the last third of the course focuses on the case in which an exact model of the system is not available. We will discuss this setting both in the online context (typically referred to as adaptive optimal control) and in the episodic context (the typical setting for reinforcement learning).

How is this course different from a standard class on Reinforcement Learning? Many classes on reinforcement learning focus primarily on the setting of discrete Markov Decision Processes (MDPs), whereas we will focus primarily on continuous MDPs. More importantly, the focus on discrete MDPs leads planning with a known model (which is typically referred to as "planning" or "control" in RL) to be relatively simple. In this course, we will spend considerably more time focusing on planning with a known model in both continuous and discrete time. Finally, the focus of this course will primarily be on model-based methods. We will touch briefly on model-free methods at the end, and combinations of model-free and model-based approaches.

A Note on Notation

The notation and language used in the control theory and reinforcement learning communities vary substantially, as so we will state all of the notational choices we make in this section. First, optimal control problems are typically stated in terms of minimizing a cost

function, whereas reinforcement learning problems aim to maximize a reward. These are mathematically identical statements, where one is simply the negation of the other. Herein, we will use the control theoretic approach of cost minimization. We write c for the cost function, f for the system dynamics, and denote the state and action at time t as $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ respectively. We write scalars as lower case letters, vectors as bold lower case letters, and matrices as upper case letters. We write a deterministic policy as $\pi(\boldsymbol{x})$, and a stochastic policy as $\pi(\boldsymbol{u} \mid \boldsymbol{x})$. We write the cost-to-go (negation of the value function) associated with policy π at time t and state $\boldsymbol{x}$ as $J_t^\pi(\boldsymbol{x})$. We will also sometimes refer to the cost-to-go as the value, but in these notes we are always referring to the expected sum of future costs. For an in-depth discussion of the notational and language differences between the artificial intelligence and control theory communities, we refer the reader to [Pow12].

Prerequisites

While these notes aim to be almost entirely self contained, familiarity with undergraduate level calculus, differential equations, and linear algebra (equivalent to CME 102 and EE 263 at Stanford) are assumed. We will briefly review nonlinear optimization in the first section of these notes, but previous experience with optimization (e.g. EE 364A) will be helpful. Finally, previous experience with machine learning (at the level of CS 229) is beneficial.

Contents

1 Nonlinear Optimization

In this section we discuss the generic nonlinear optimization problem that forms the basis for the rest of the material presented in this class. We write the minimization problem as

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad f(\boldsymbol{x})$$

where f is the cost function, usually assumed twice continuously differentiable, $x \in \mathbb{R}^n$ is the optimization variable, and $\mathcal{X} \subset \mathbb{R}^n$ is the constraint set. The special case in which the cost function is linear and the constraint set is specified by linear equations and/or inequalities is *linear optimization*, which we will not discuss.

1.1 Unconstrained Nonlinear Optimization

We will first address the unconstrained case, in which $\mathcal{X} = \mathbb{R}^n$. A vector $\boldsymbol{x}$ is said to be an unconstrained *local minimum* if there exists $\epsilon > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{\boldsymbol{x} \mid \|\boldsymbol{x}-\boldsymbol{x}^*\| \leq \epsilon\}$, and $\boldsymbol{x}$ is said to be an unconstrained *global minimum* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $x \in \mathbb{R}^n$.

1.1.1 Necessary Conditions for Optimality

For a differentiable cost function, we can compare the cost of a point to its neighbors by considering a small variation $\Delta\boldsymbol{x}$ from $\boldsymbol{x}^*$. By using Taylor expansions, this yields a first order cost variation

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)^T \Delta\boldsymbol{x} \tag{1}$$

and a second order cost variation

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)^T \Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}^*)\Delta\boldsymbol{x}. \tag{2}$$

Setting $\Delta\boldsymbol{x}$ to be equal to positive and negative multiples of the unit coordinate vector, we have

$$\frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \geq 0 \tag{3}$$

where x_i denotes the i'th coordinate of $\boldsymbol{x}$, and

$$\frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \leq 0 \tag{4}$$

for all i, which is only satisfied by $f(\boldsymbol{x}^*) = 0$. This is referred to as the *first order necessary condition for optimality*. Looking at the second order variation, and noting that $f(\boldsymbol{x}^*) = 0$, we expect

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \geq 0 \tag{5}$$

and thus

$$\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}^*)\Delta\boldsymbol{x} \geq 0 \tag{6}$$

which implies $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite. This is referred to as the *second order necessary condition for optimality*. Stating these conditons formally,

Theorem 1.1 (Necessary Conditions for Optimality (NOC))**.** *Let $\boldsymbol{x}^*$ be an unconstrained local minimum of $f : \mathbb{R}^n \to \mathbb{R}$ and $f \in C^1$ in an open set S containing $\boldsymbol{x}^*$. Then*

$$\nabla f(\boldsymbol{x}^*) = 0. \tag{7}$$

If $f \in C^2$ within S, $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite.

Proof is given in Section 1.1 of [Ber16].

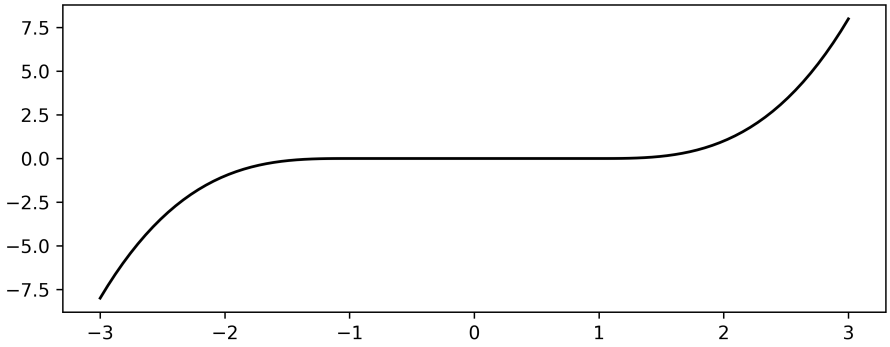

Figure 1: An example of a function for which the necessary conditions of optimality are satisfied but the sufficient conditions are not.

1.1.2 Sufficient Conditions for Optimality

If we strengthen the second order condition to $\nabla^2 f(\boldsymbol{x}^*)$ being positive definite, we have the sufficient conditions for $\boldsymbol{x}^*$ being a local minimum. Why is the second order necessary conditions not sufficient? An example function is given in figure 1. Formally,

Theorem 1.2 (Sufficient Conditions for Optimality (SOC))**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be C^2 in an open set S. Suppose a vector $\boldsymbol{x}^*$ satisfies the conditions $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then $\boldsymbol{x}^*$ is a strict unconstrained local minimum of f.*

Proof is again given in Section 1.1 of [Ber16]. There are several reasons why the optimality conditions are important. In a general nonlinear optimization setting, they can be used to filter candidates for global minima. They can be used for sensitivity analysis, in which the sensitivity of $\boldsymbol{x}^*$ to model parameters can be quantified [Ber16]. This is common in e.g. microeconomics. Finally, these conditions often provide the basis for the design and analysis of optimization algorithms.

1.1.3 Special case: Convex Optimization

A special case within nonlinear optimization is the set of *convex optimization* problems. A set $S \subset \mathbb{R}^n$ is called *convex* if

$$\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y} \in S, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in S, \forall \alpha \in [0, 1]. \tag{8}$$

For S convex, a function $f : S \to \mathbb{R}$ is called convex if

$$f(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha) f(\boldsymbol{y}). \tag{9}$$

This class of problems has several important characteristics. If f is convex, then

- A local minimum of f over S is also a global minimum over S. If in addition f is strictly convex (the inequality in (9) is strict), there exists at most one global minimum of f.

- If $f \in C^1$ and convex, and the set S is open, $\nabla f(\boldsymbol{x}^*) = 0$ is a necessary and sufficient condition for a vector $\boldsymbol{x}^* \in S$ to be a global minimum over S.

Convex optimization problems have several nice properties that make them (usually) computationally efficient to solve, and the first property above gives a certificate of having obtained global optimality that is difficult or impossible to obtain in the general nonlinear optimization setting. For a thorough treatment of convex optimization theory and algorithms, see [BV04].

1.1.4 Computational Methods

In this subsection we will discuss the class of algorithms known as *gradient methods* for finding local minima in nonlinear optimization problems. These approaches, rely (roughly) on following the gradient of the function "downhill", toward the minima. More concretely, these algorithms rely on taking steps of the form

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k \tag{10}$$

where if $\nabla f(\boldsymbol{x}) \neq 0$, $\boldsymbol{d}^k$ is chosen so that

$$\nabla f(\boldsymbol{x})^T \boldsymbol{d}^k < 0 \tag{11}$$

and $\alpha > 0$. Typically, the step size α^k is chosen such that

$$f(\boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k) < f(\boldsymbol{x}^k), \tag{12}$$

but generally, the step size and the direction of descent ($\boldsymbol{d}^k$) are tuning parameters.

We will look at the general class of descent directions of the form

$$\boldsymbol{d}^k = -D^k \nabla f(\boldsymbol{x}^k) \tag{13}$$

where $D^k > 0$ (note that this guarantees $\nabla f(\boldsymbol{x}^k)^T \boldsymbol{d}^k < 0$).

Steepest descent, $D^k = I$. The simplest choice of descent direction is directly following the gradient, and ignoring second order function information. In practice, this often leads to slow convergence and possible oscillation.

Newton's Method, $D^k = (\nabla^2 f(\boldsymbol{x}^k))^{-1}$. The underlying idea of this approach is to at each iteration, minimize the quadratic approximation of f around $\boldsymbol{x}^k$,

$$f^k(\boldsymbol{x}) = f(\boldsymbol{x}^k) + \nabla f(\boldsymbol{x}^k)^T(\boldsymbol{x} - \boldsymbol{x}^k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^k)^T \nabla^2 f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k). \tag{14}$$

Setting the derivative of this to zero, we obtain

$$\nabla f(\boldsymbol{x}^k) + \nabla^2 f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) = 0 \tag{15}$$

and thus, by setting $\boldsymbol{x}^{k+1}$ to be the $\boldsymbol{x}$ that satisfies the above, we get the

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - (\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k) \tag{16}$$

or more generally,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha(\nabla^2 f(\boldsymbol{x}^k))^{-1} \nabla f(\boldsymbol{x}^k). \tag{17}$$

Note that this update is only valid for $\nabla^2 f(\boldsymbol{x}^k) > 0$. When this condition doesn't hold, $\boldsymbol{x}^{k+1}$ is not a minimizer of the second order approximation (as a result of the SOCs).

Diagonally scaled steepest descent, $D^k = \mathbf{diag}(d_1^k, \ldots, d_n^k)$. Have $d_i^k > 0 \forall i$. A popular choice is

$$d_i^k = \left(\frac{\partial^2 f(\boldsymbol{x}^k)}{\partial x_i^2} \right)^{-1} \tag{18}$$

which is a diagonal approximation of the Hessian.

Modified Newton's method, $D^k = (\nabla^2 f(\boldsymbol{x}^0))^{-1}$. Requires $\nabla^2 f(\boldsymbol{x}^0) > 0$. For cases in which one expects $\nabla^2 f(\boldsymbol{x}^0) \approx \nabla^2 f(\boldsymbol{x}^k)$, this removes having to compute the Hessian at each step.

In addition to choosing the descent direction, there also exist a variety of methods to choose the step size α. A computationally intensive but efficient (in terms of the number of steps taken) is using a minimization rule of the form

$$\alpha^k = \text{argmin}_{\alpha \geq 0} f(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k) \tag{19}$$

which is usually solved via line search. Alternative approaches include a limited minimization rule, in which you constrain $\alpha^k \in [0, s]$ during the line search, or simpler approach such as a constant step size (which may not guarantee convergence), or a diminishing scheduled step size. In this last case, schedules are typically chosen such that $\alpha^k \to 0$ as $k \to \infty$, while $\sum_{k=0}^{\infty} \alpha^k = +\infty$.

1.2 Constrained Nonlinear Optimization

2 Dynamic Programming and the Linear Quadratic Regulator

References

[Ber16] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 2016.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Pow12] Warren B Powell. AI, OR and control theory: A rosetta stone for stochastic optimization. 2012.