Chapter 1

Jan/24: The dynamic programming problem

In this course, we are reading Bertsekas' RL book 11 pages at a time.

The book starts by referring to the recent advances in the RL, especially AlphaGo and AlphaZero. Even more recent development is the that DeepMind founder, Demis Hassabis, won 2024 Nobel prize in Chemistry for his contributions to AlphaFold2, an RL algorithm for predicting protein folding.

To understand AlphaGo which came out in 2016, here's nice clip from the movie with the same name

1.0.1 Two stages: Offline training and online play

To understand the overall structure of AlphaZero, and its connection to our DP/RL methodology, it is useful to divide its design into two parts: *off-line training*, which is an algorithm that learns how to evaluate chess positions, and how to steer itself towards good positions with a default/base chess player, and *on-line play*, which is an algorithm that generates good moves in real time against a human or computer opponent, using the training it went through off-line. We will next briefly describe these algorithms, and relate them to DP concepts and principles.

1.0.2 Policy network and value Network

This is the part of the program that learns how to play through off-line self-training, and is illustrated in Fig. 1.1.1. The algorithm generates a sequence of *chess players* and *position evaluators*. A chess player assigns "probabilities" to all legal moves in a given position, representing the likelihood of each move being chosen. A position evaluator assigns a numerical score to a given position, predicting the player's chances of winning from that position. The chess player and the position evaluator are represented by two neural networks, a *policy network* and a *value network*, which accept a chess position and generate a set of move probabilities and a position evaluation, respectively.†

1.0.3 Policy evaluation and policy improvement

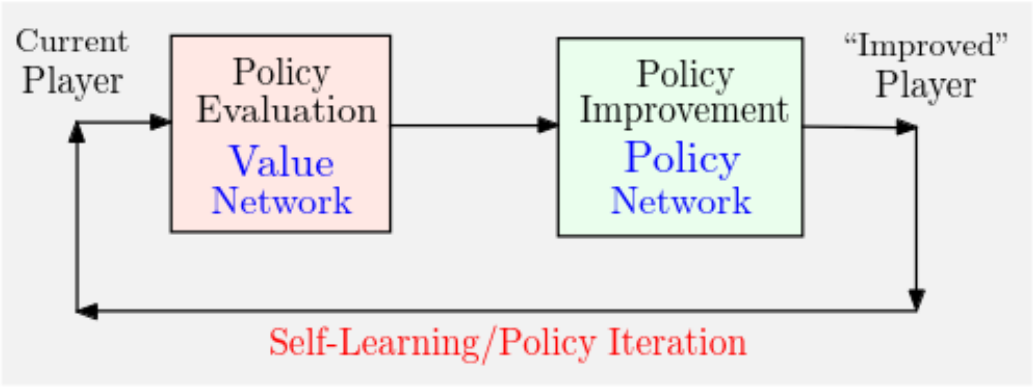

Figure 1.1.1 Illustration of the AlphaZero training algorithm. It generates a sequence of position evaluators and chess players. The position evaluator and the chess player are represented by two neural networks, a value network and a policy network, which accept a chess position and generate a position evaluation and a set of move probabilities, respectively.

1.0.4 Offline training and Online play

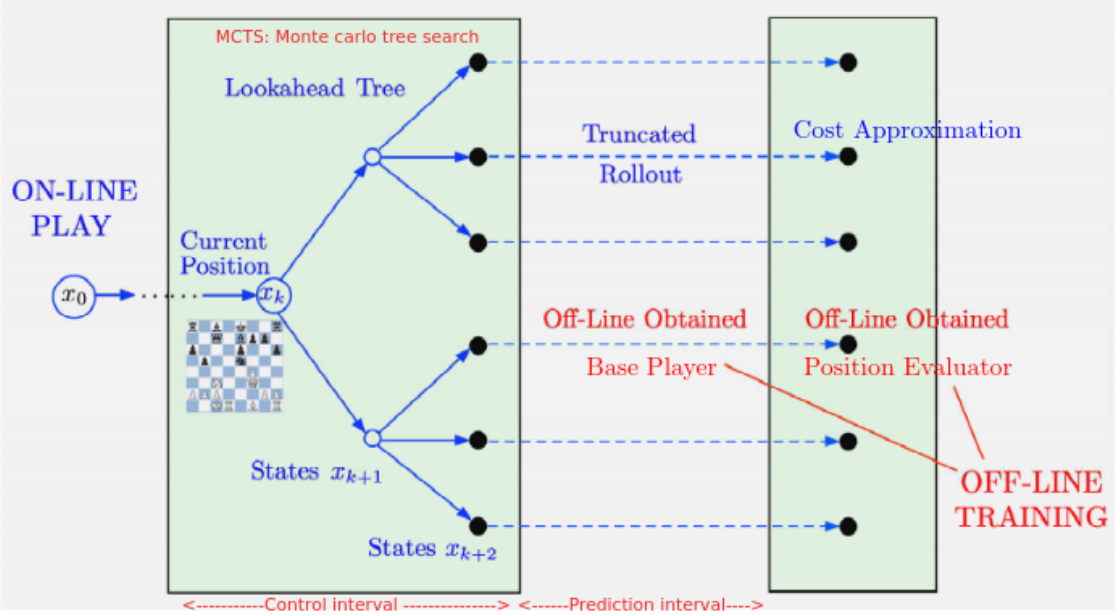

Figure 1.1.2 Illustration of an on-line player such as the one used in AlphaGo, AlphaZero, and Tesauro's backgammon program [TeG96]. At a given position, it generates a lookahead tree of multiple moves up to some depth, then runs the off-line obtained player for some more moves, and evaluates the effect of the remaining moves by using the position evaluator of the off-line player.

1.0.5 Alternative names of RL

Approximations in the value space, also known as *approximate dynamic programming* or *neuro-dynamic programming*. *Reinforcement learning* includes both approximations in the value space and in the policy space. This can be used in control theory as *model predictive control*.

1.0.6 1.2.1 Finite Horizon Problem Formulation

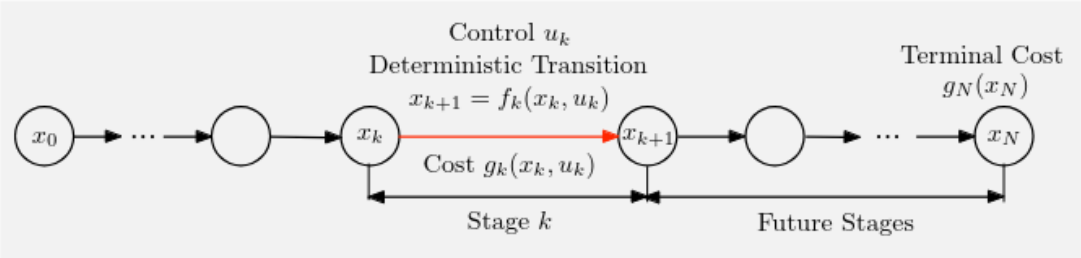

Figure 1.2.1 Illustration of a deterministic N-stage optimal control problem. Starting from state x_k, the next state under control u_k is generated nonrandomly, according to

$$x_{k+1} = f_k(x_k, u_k),$$

and a stage cost $g_k(x_k, u_k)$ is incurred.

The *cost to go* is defined as the summation of step cost $g_k(x_k, u_k)$ with a *terminal cost* $g_N(x_N)$.

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{1.1}$$

$$x_{k+1} = f_k(x_k, u_k) \tag{1.2}$$

We want to minimize the cost over all sequences $\{u_0, \ldots, u_{N-1}\}$ to find the optimal cost-to-go of x_0,

$$J^*(x_0) = \min_{u_k, k \in \{0,\ldots,N-1\}} J(x_0; u_0, \ldots, u_{N-1}) \tag{1.3}$$