CSE 574 Lecture 12: Midterm Wrapup

Professor: Stephanie Gil

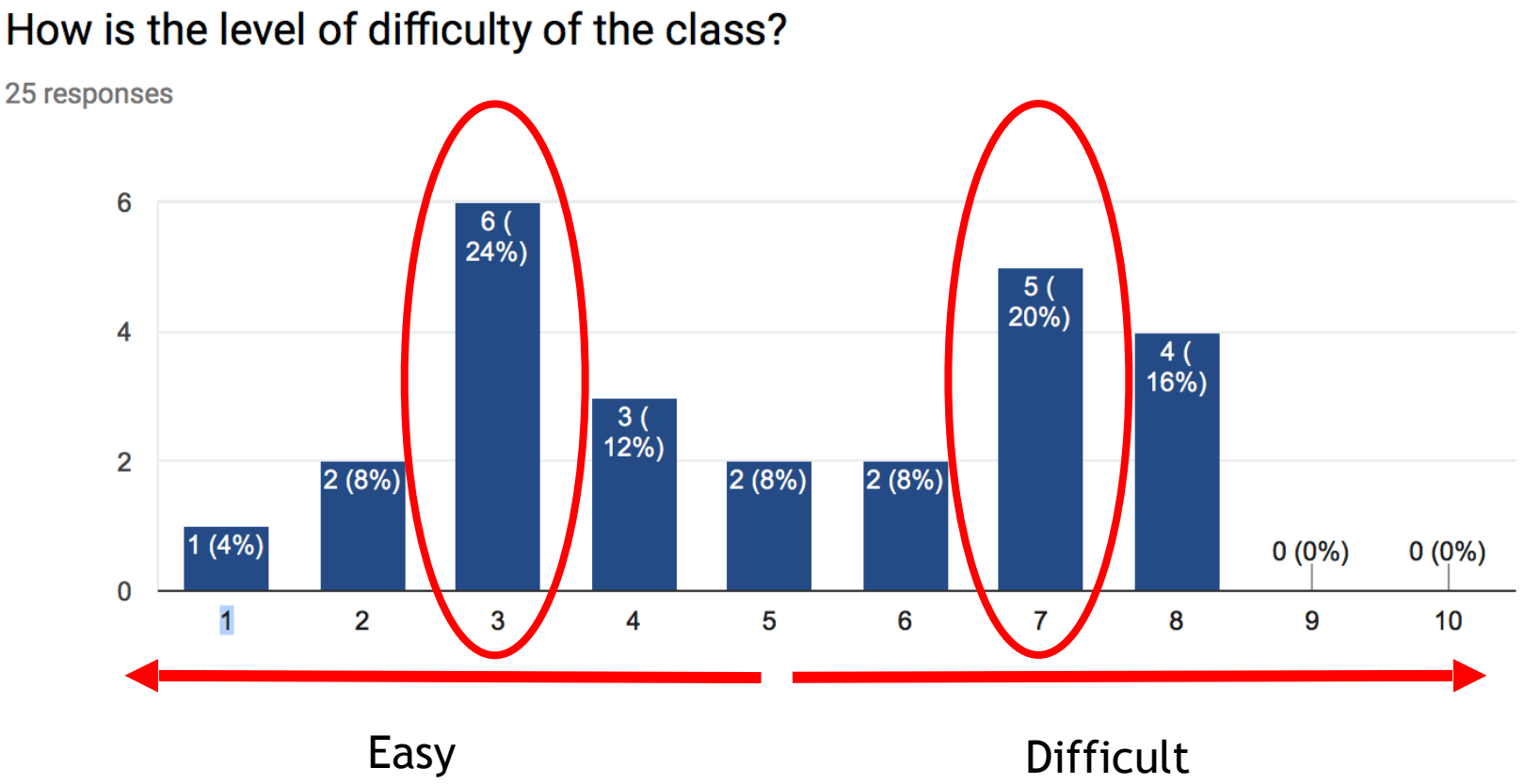
Email: sgil@asu.edu (Office hours M 12-1p BYENG 386)

TAs: Sushmita Bhattacharya sbhatt55@asu.edu (Office hours M 5-6 BYENG 392)

Weiying Wang wwang239@asu.edu (Office hours Th 2:30-3:30 BYENG 392)

Class Survey

- Difficulty level of the course



Course Feedback

- Thank you for your feedback and suggestions!

- Some common feedback and how we will incorporate it moving forward:
  - More examples of the topic material
    - We will have a few “recitation” lectures with the sole purpose of reviewing material and working through examples
  - More review of Dynamic Programming
    - We are happy that you like DP! We will go over this more in class today before moving to reinforcement learning (which builds upon the principles of DP)

- Pop quizzes
  - Seems like you are enjoying these – great!
  - Continue to perform well on these as a way to improve your score

Where We've Been…

CSE 574: Tentative Class Topic List (subject to change)

I. Problem Representation and Search
   a. State space and how to model problems
   b. SAT/CSP/IP based planning and graph search
   c. Adversarial search

I. Markov Processes
   a. Markov models
   b. Markov decision processes
   c. Online planning

I. Intro to Dynamic Programming
   a. Shortest path problems
   b. Optimal control
   c. Minimax problems

I. Hidden Markov Models
   a. Modeling using HMMs
   b. Partially Observable Markov Decision Processes (POMDPs)

II. Multi-agent Planning

III. Motion Planning
   a. Bug Algorithms
   b. Potential Field Methods
   c. Sampling Based Motion Planning

IV. Localization and Mapping
   a. Robot Localization
   b. Intro to Simultaneous Localization and Mapping (SLAM)

V. Reinforcement Learning (Approximate Dynamic Programming)

Where We're Going…

CSE 574: Tentative Class Topic List (subject to change)

I. Problem Representation and Search
   a. State space and how to model problems
   b. SAT/CSP/IP based planning and graph search
   c. Adversarial search

I. Markov Processes
   a. Markov models
   b. Markov decision processes
   c. Online planning

I. Intro to Dynamic Programming
   a. Shortest path problems
   b. Optimal control
   c. Minimax problems

I. Reinforcement Learning (Approximate Dynamic Programming)

II. Hidden Markov Models
   a. Modeling using HMMs
   b. Partially Observable Markov Decision Processes (POMDPs)

III. Localization and Mapping
   a. Robot Localization
   b. Intro to Simultaneous Localization and Mapping (SLAM)

IV. Multi-agent Planning

V. Motion Planning
   a. Bug Algorithms
   b. Potential Field Methods
   c. Sampling Based Motion Planning

Final Project Important Dates

- **Wednesday October 24th:** Final project literature review and preliminary results due (*2 weeks from today*)

- **Wednesday November 14th:** Project presentations begin

- **Friday November 30th:** Final project report due

Final Project Guidelines

Anybody still looking for project partners?
Opportunity to pitch your ideas…

Final Project Guidelines

1. Literature review Wednesday October 24th
   i. What is your project topic? What has been done in the field (survey of at least 8 related papers)?
   ii. What are the major challenges for this topic area?

Final Project Guidelines

1. Literature review Wednesday October 24th
   i. What is your project topic? What has been done in the field (survey of at least 8 related papers)?
   ii. What are the major challenges for this topic area?

2. Benchmark paper and algorithm Wednesday October 24th
   i. Implement algorithm from existing paper. Show plots and figures with preliminary results for this algorithm (you will be asked to submit your code).
   ii. What are the pros and cons of this implementation?

Final Project Guidelines

1. Literature review Wednesday October 24th
   i. What is your project topic? What has been done in the field (survey of at least 8 related papers)?
   ii. What are the major challenges for this topic area?

2. Benchmark paper and algorithm Wednesday October 24th
   i. Implement algorithm from existing paper. Show plots and figures with preliminary results for this algorithm (you will be asked to submit your code).
   ii. What are the pros and cons of this implementation?

3. Project presentation – Includes your extension to the benchmark paper Wednesday November 14th
   i. Extend the existing algorithm in a novel way. Please use your creativity here. This is your chance to really explore and learn something new!
   ii. Some ideas:
      - Implement your benchmark algorithm towards a new application space. Is it effective? What are the strengths? Where does it fall short?
      - What are the assumptions of your benchmark algorithm? What happens if you violate these assumptions? For example, can it handle uncertainty? Can it be improved to handle uncertainty?
      - Are there any results that your chosen algorithm did not implement and you would have liked to see? Implement these results
   iii. You will need to provide figures/plots and results that showcase your extension to the benchmark algorithm. You will also be asked to submit your code for this.

Final Project Guidelines

4. Report Friday November 30th
   i. Your report should use the IEEE double column 6-8 page format (6 is the min, 8 is the max)
   ii. Your report should include an introduction, background literature review, benchmark implementation and results, your extension implementation and results, discussion/ conclusion. *You must specify how you have extended the benchmark algorithm and what elements of your work is novel.
   iii. Your report should include peer feedback from your presentation
   iv. Format https://www.ieee.org/conferences/publishing/templates.html

How to Use the IEEEtran LaTeX Class

Michael Shell, *Member, IEEE*

(Invited Paper)

***Abstract*—This article describes how to use the IEEEtran class with LaTeX to produce high quality typeset papers that are suitable for submission to the Institute of Electrical and Electronics Engineers (IEEE). IEEEtran can produce conference, journal and technical note (correspondence) papers with a suitable choice of class options. This document was produced using IEEEtran in journal mode.**

***Index Terms*—Class, IEEEtran, LaTeX, paper, style, template, typesetting.**

I. Introduction

WITH a recent IEEEtran class file, a computer running LaTeX, and a basic understanding of the LaTeX language, an author can produce professional quality typeset research papers very quickly, inexpensively, and with minimal effort. The purpose of this article is to serve as a user guide of IEEEtran LaTeX class and to document its unique features and optional packages along with more complex usage techniques, can be found in `bare_adv.tex`.

It is assumed that the reader has at least a basic working knowledge of LaTeX. Those so lacking are strongly encouraged to read some of the excellent literature on the subject [4]–[6]. In particular, Tobias Oetiker's *The Not So Short Introduction to LaTeX 2ε* [5], which provides a general overview of working with LaTeX, and Stefan M. Moser's *How to Typeset Equations in LaTeX* [6], which focuses on the formatting of IEEE-style equations using IEEEtran's IEEEeqnarray commands, are both available for free online.

General support for LaTeX related questions can be obtained in the internet newsgroup comp.text.tex. There is also a searchable list of frequently asked questions about LaTeX [7].

Please note that the appendices sections contain information on installing the IEEEtran class file as well as tips on how to

A Review of Dynamic Programming

- Fundamental building blocks for reinforcement learning

- From here the topics get conceptually less defined – we must *learn* the structure (before we used this to solve for optimal policies)

- Review of optimal policies and problem types…

Types of Dynamic Programming

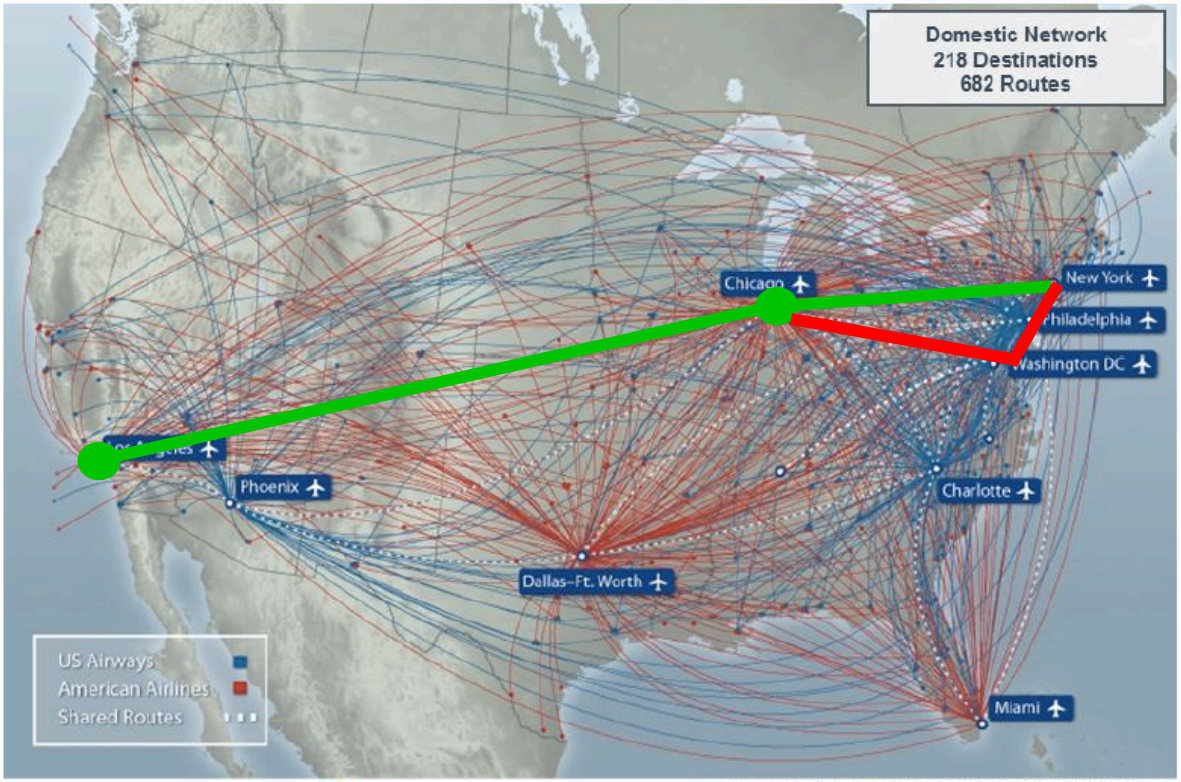
- Finite horizon

- Infinite horizon
  - Stochastic shortest path
  - Discounted problems
  - Average cost per stage problems

➢**Goal** of dynamic programming: find an optimal policy (sequence of actions that are optimal)

➢**Key concept** of dynamic programming: principle of optimality (structure of the problem is key)

Intuitive Explanation



Example: Midterm Question

Step 1: Formulate the problem (N horizon, infinite horizon?)

States= j, the number of consecutive days the star quarterback plays

Available actions based on your state

Actions ={play, don't play}

Costs/rewards

Reward of P for playing quarterback

Reward of S for playing 2nd quarterback

Cost of C for star quarterback getting injured

Probabilities

is the probability that the quarterback gets injured

Step 2: Write the DP recursion (over states and actions)

Step 3: Start solving from the last stage

Plug in your problem parameters to the DP recursion starting at the last stage

You are the coach of an American football team and you have to decide for which games you should play your star quarterback over the course of a 3-game season (so that k=0,1,2 for this problem). For each day that you play your star quarterback you earn a profit P, but there is a probability that he gets injured in the game and you incur a cost C. This probability increases for each consecutive day that you play your star quarterback (and returns to if you decide not to play him one game). You may assume that he can play again even if he gets hurt during a game. If you decide not to play your star quarterback and instead play your second-in-line quarterback, you earn a profit of S and there is no probability that this player gets injured during the game. Assume that the terminal cost is zero for your problem. Your objective is to *maximize* your profit over the season.

Example: Midterm Question

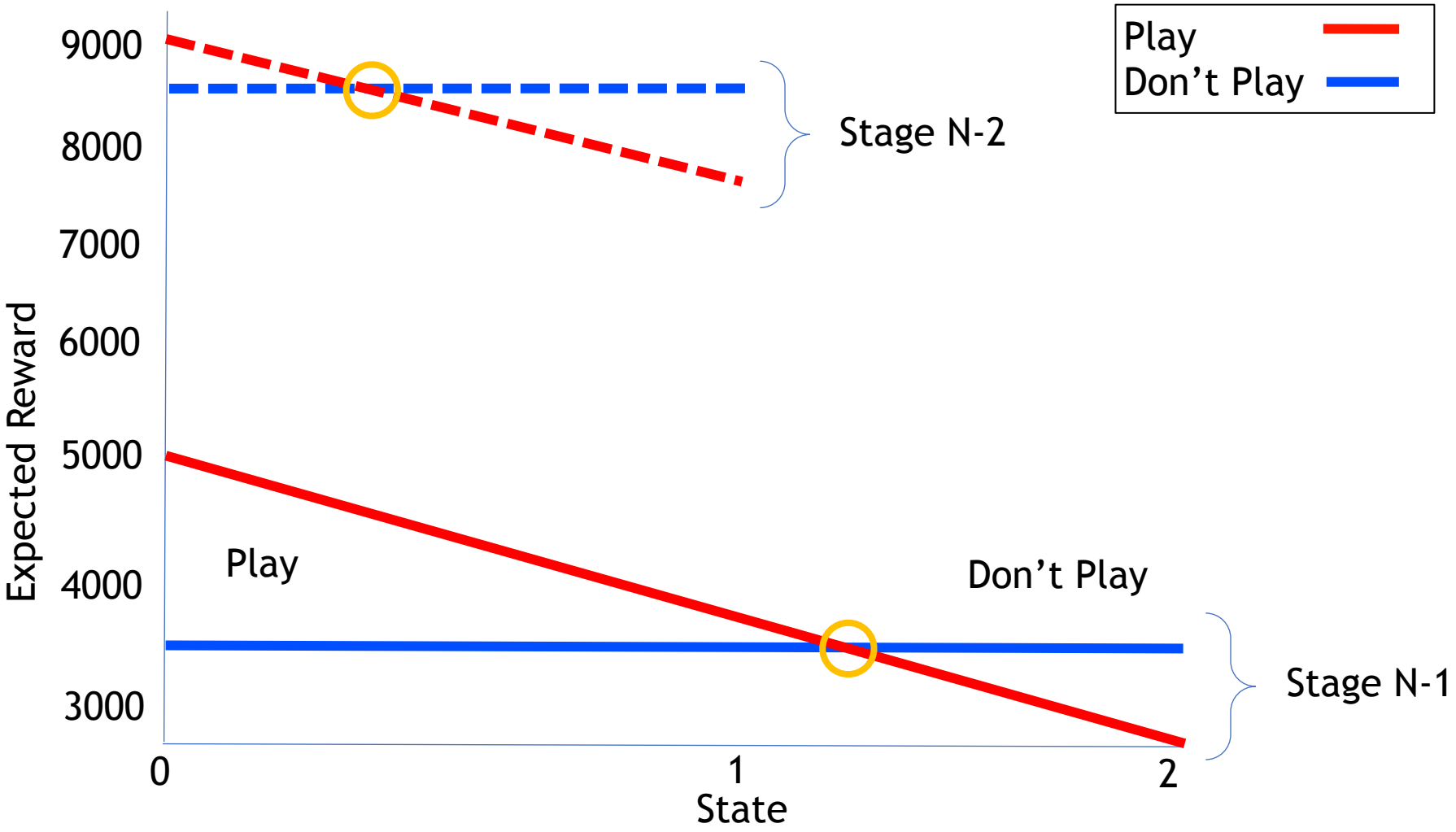
Step 2: Write the DP recursion (over states and actions)

$$J_N(j) = 0 \quad \forall j$$

$$J_k(j) = \max\Big[\underbrace{p - p_j C + J_{k+1}(j+1)}_{\text{play}},\ \underbrace{S + J_{k+1}(0)}_{\text{don't play}}\Big]$$

Example: Midterm Question

- Reasoning about the optimal policy:

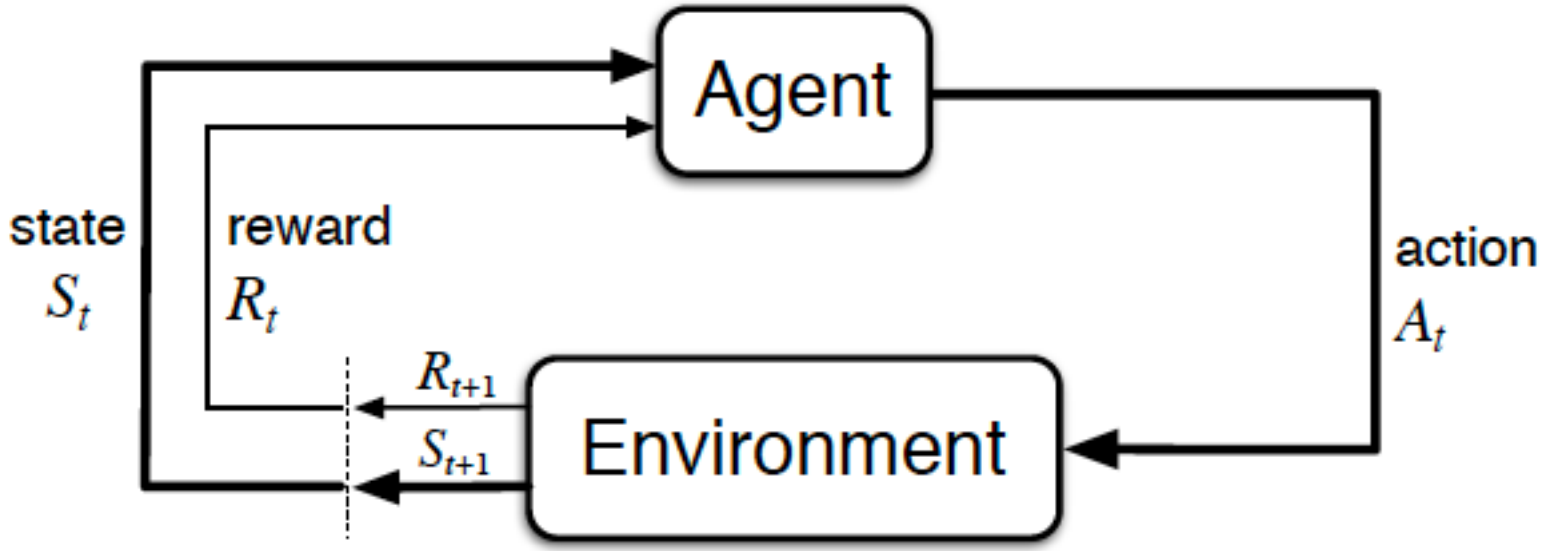


Introduction to Reinforcement Learning

- Difference to what we were doing before:
  - Dynamic programming looks for the *optimal policy* given a model of
    - Environment (distribution over uncertainties)
    - Cost/reward structure
    - Actions

    ➢Use this knowledge to learn an evaluation function

Agent-Environment Interface



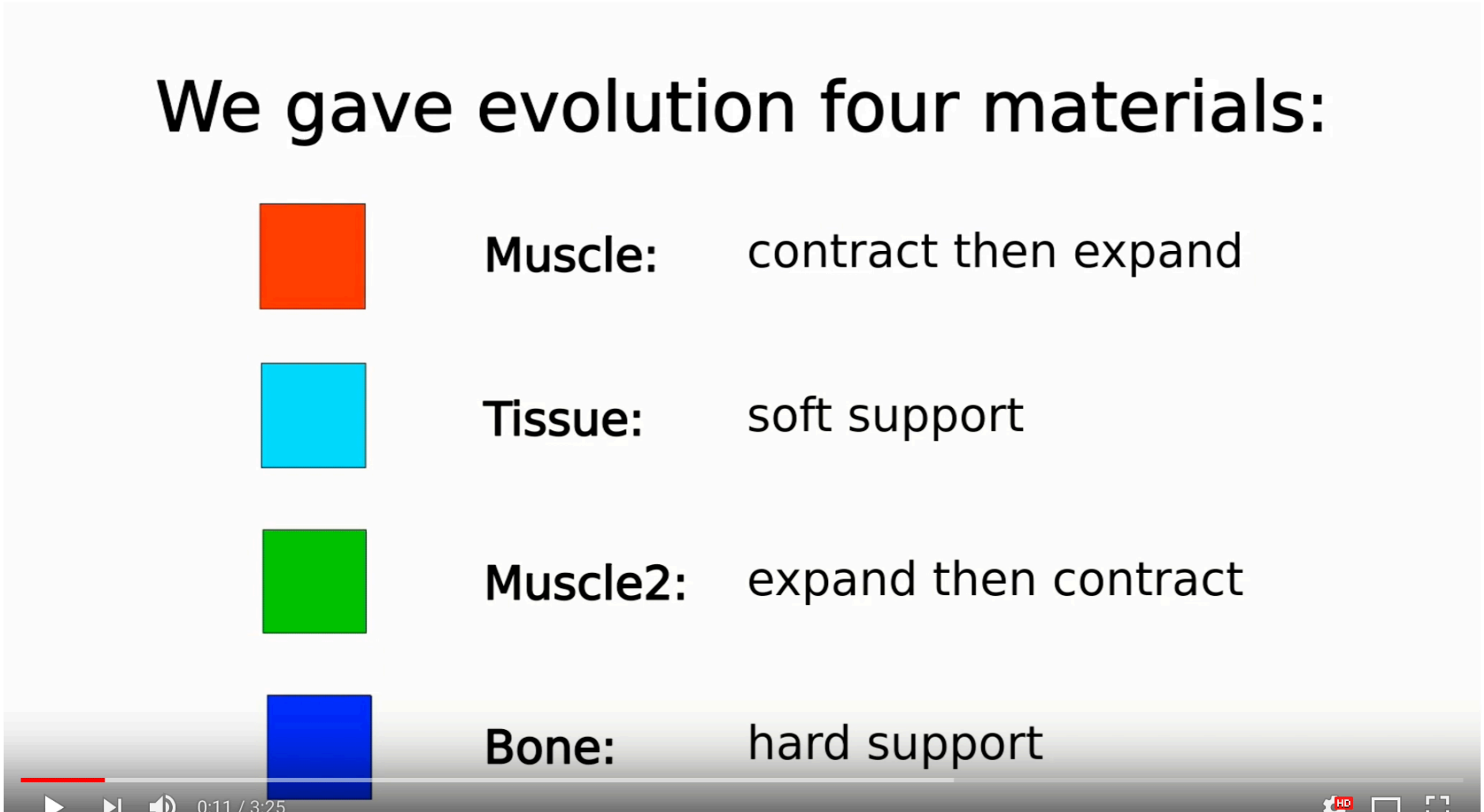
A trajectory here, or sequence, looks like this:
S_0, A_0, R_1, S_1, A_1, R_2,…

Introduction to Reinforcement Learning

- Difference to what we were doing before:
  - Dynamic programming looks for the *optimal policy* given a model of
    - Environment (distribution over uncertainties)
    - Cost/reward structure
    - Actions
    - ➢Use this knowledge to learn an evaluation function
  - What if we don't know these things?
    - Complex problems
    - Large search spaces
  - Idea: *train* the evaluation function from examples

Some Examples

- Hod Lipson 2013 – genetic algorithms and



Some Examples (cont)

- Andrew Ng Helicopter flight

Types of Learning

- "Swiss-cheese method" – can't learn everything at once. What do we keep fixed?

1. **Passive learning** – the agent's policy is fixed and the task is to learn the utilities of the states (or state-action pairs)

2. **Active learning** – the agent must learn what to do (which actions to take)

3. **Exploration** – an agent must experience as much as possible of its environment in order to learn how to behave in it

Next Few Lectures

- Readings from
  - Russel and Norvig text chapter 21
  - Sutton and Barto text chapter 6.1-6.2, 6.5