The life cycle of a person's training of an assistive machine

(Both person and machine are goal-seeking agents)

1. At the beginning, all the person can do is reward and punish; no other signal may have an agreed meaning

2. Using rewards, the machine can learn a value function from the person's face, body language, tone of voice, and other cues; this enables further training with much less tedium

3. Using values, the person teaches the machine a language for instructing it at a low level

   - commands, cues, pointing, gated control, ...

4. Using instructions, higher-level commands, cues, and shorthands can be established ad infinitum

Building communicative capital

Toward a New View of Action Selection

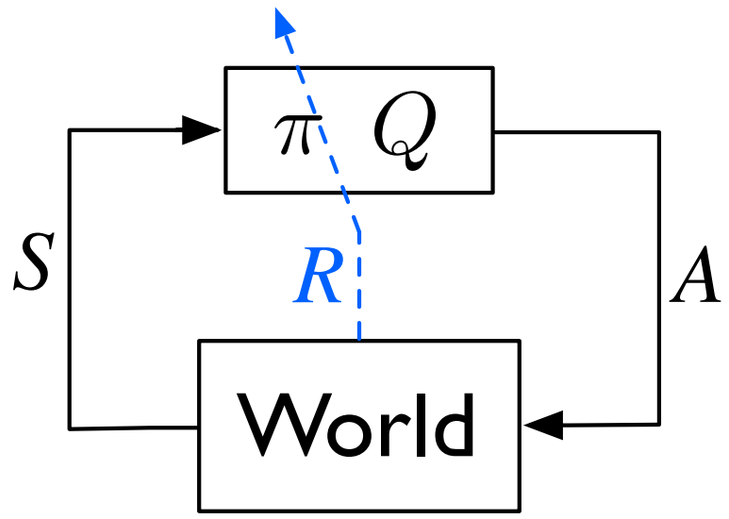
The Subgoal Keyboard

Rich Sutton

Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computing Science
University of Alberta

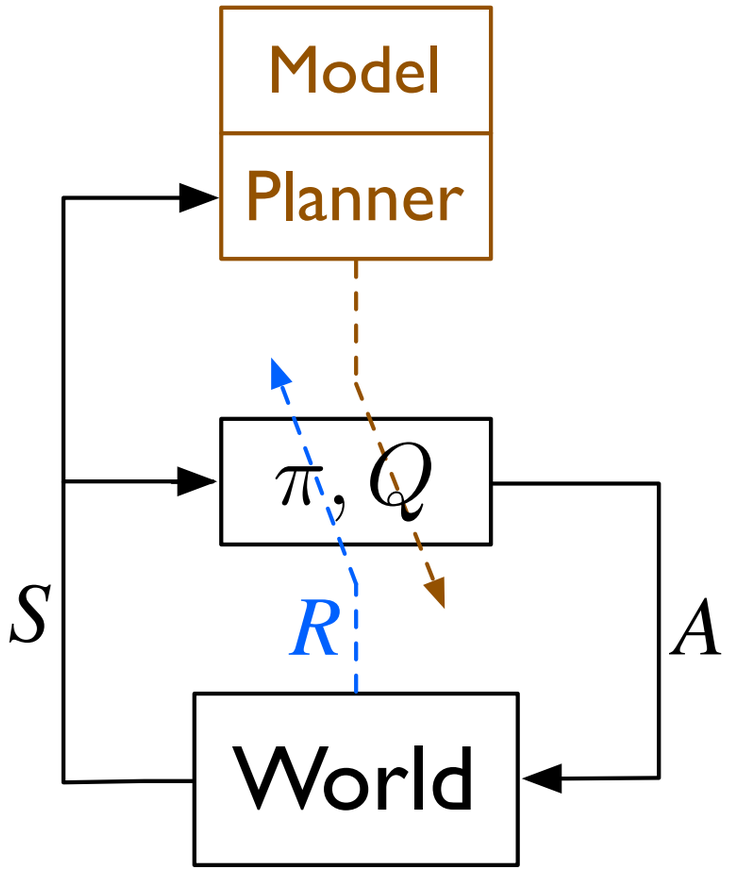
Old views of action selection

- ε-greedy with action-value functions
- Policy-gradient methods



Old views of action selection

- ε-greedy with action-value functions
- Policy-gradient methods
- The foreground-background architecture: planning in back, policy in front
- Planning with options,
  then executing options with interruption
- Greedy action selection with boredom (pursuit of novelty, curiosity, intrinsic motivation)—the changing goal could give rise to sequences of coherent behavior



Goals for action selection

1. Action selection should be quick to shape coherent behavior to current opportunities

    - Suppose cakes are passing by you. It should be easy to grab them. It should be easy to decide on one to grab, then pursue it

2. You should be able to think about ways of behaving (options), decide on which you will do in future situations, then do that one when you get there

    - Is there a decision level separate from the low-level actions? A level where we decide between alternate coherent courses of behavior? (This is different from options because it has no termination aspect)

Maybe we need to create such a level!

Goals for action selection (2)

3. Action selection should be combinatorial, not monolithic

    - Rather than choosing between full policies 1, 2, 3, …
    - You want to parameterized policies with policy aspects 1, 2, 3, …
    - That you can put together in many combinations and degrees
    - So that you can get great flexibility without an enormous number of policies that must be learned about all separately

What about options?

- An option is a policy and a way of terminating it
  (and sometimes a set of states in which it can start)

  - open the door, grasp the object, walk to work, fly to Barbados

- The common view is that each option is monolithic, distinct, unitary, of one piece

  - you might refer to them by a number, index, symbol (**`walk-to-work`**, **`open-the-door`**, **`grasp-the-object`**)

- Options give us coherence, but no blending of coherent things, thus limited flexibility

Options can be combinatorial

- They are a mathematical abstraction, not a data structure
- There can be parameterized families of options (just as there can be parameterized functions)
  - e.g., **`walk-to-work`** *with varying degrees of urgency*
  - e.g., **`grasp-the-object`** *at this location*
  - e.g., **`run`** *plus* **`chew-gum`** *plus* **`keep-eye-on-ball`**
  - e.g., **`walk-stairs`** *plus* **`dont-spill-coffee`**
- In general, you have many option aspects/features, every <u>combination</u> of which is a complete option
  - place the glass on the table, parameterized by speed, care, location, orientation, degree of body motion, degree of visual monitoring, or with various joints constrained

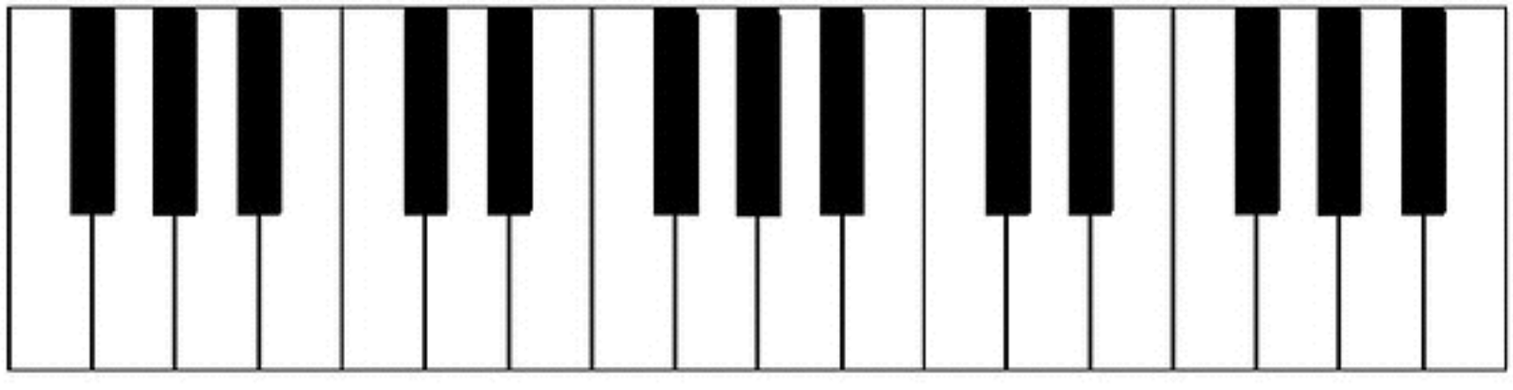
Executing options?

- If options are distinct and you have just a finite set, then in real life you would never actually do exactly any of them. What you actually do should not be constrained to one of them

- But if the set of options is infinite and combinatorial, then maybe anything you would ever want to do would be one of them. In fact, they could be the key to generating behavior with a good mix of preplanned coherence and sensitivity to the current situation

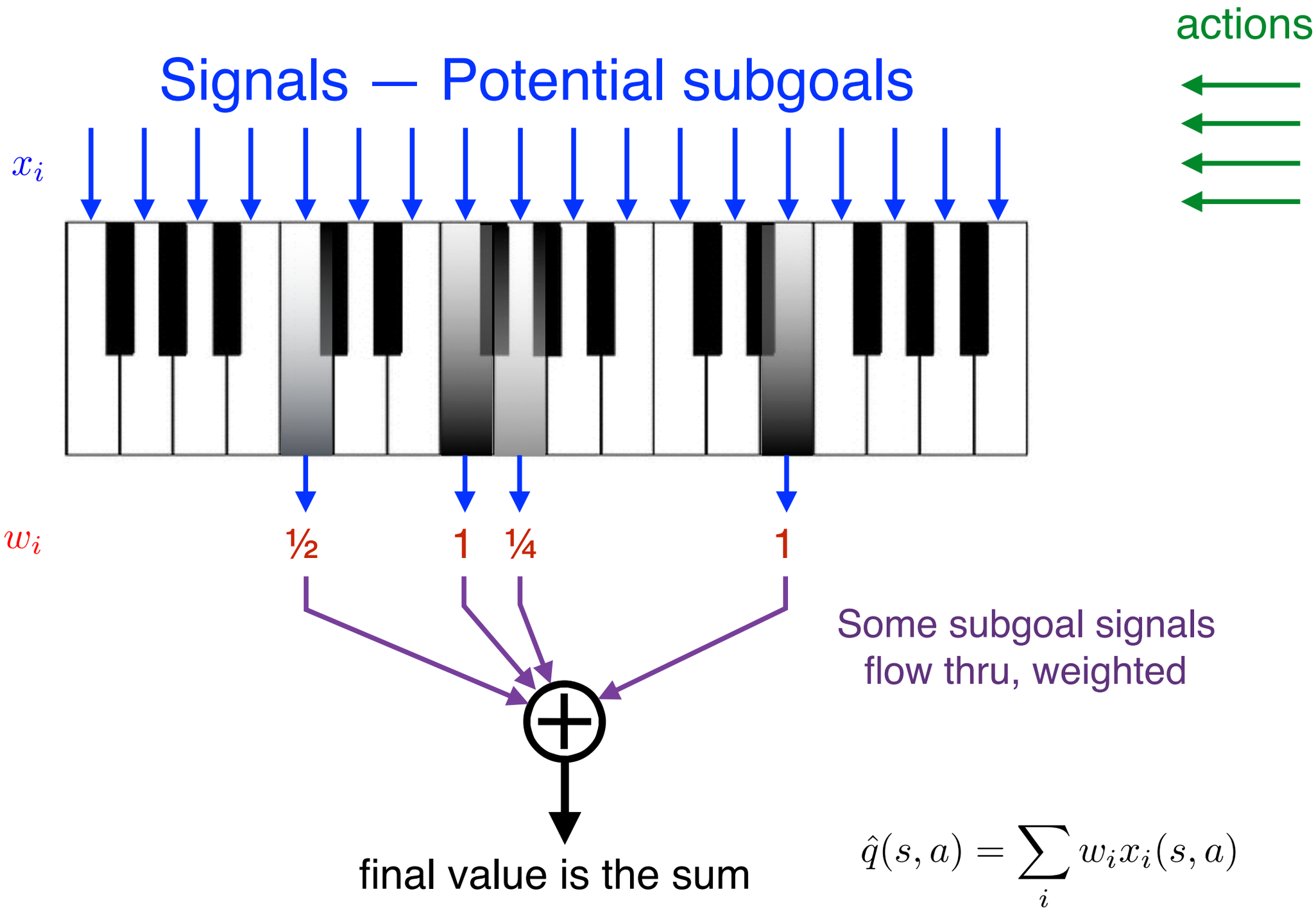
- That is, you might make decisions at the level of option aspects/features/descriptors, and use these to select actions

The Subgoal Keyboard



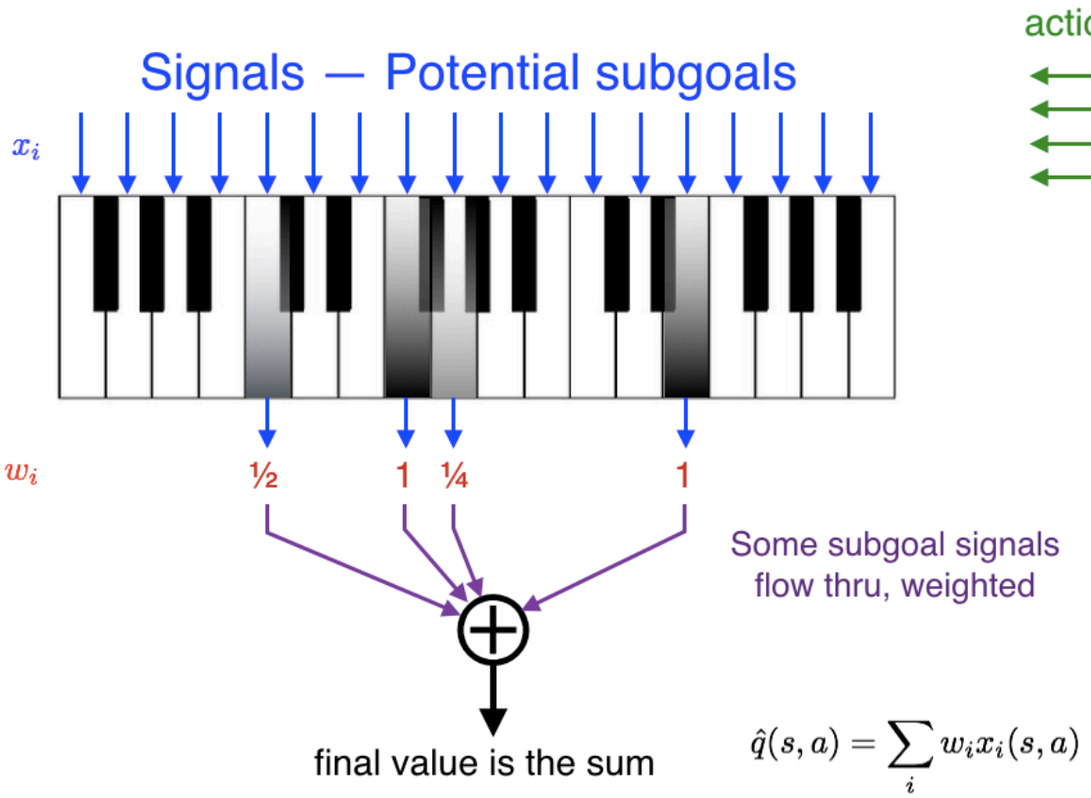
- Each note is a subgoal, a signal that the agent knows how to achieve to some extent
    - e.g., a prediction (GVF), a sensory event, the light switch position, a grasp, a sensation
- A chord comprises the overall goal
- Chord = vector of subgoal strengths

Subgoal keyboard via action values



$$\hat{q}(s,a) = \sum_i w_i x_i(s,a)$$

Subgoal keyboard via action values



- Any chord (any setting of the weights) instantly gives us an action-value function directed towards the corresponding subgoals
- As usual, we can take the max over actions to determine action selections
- Thus we can smoothly switch from one coherent policy to another using the subgoal weights
- A new language for flexibly generating coherent action selections

The SK achieves our goals

- (1) We can quickly shape coherent behavior to current opportunities (by hitting a new chord)

- (3) We can blend coherent policies combinatorially

- (2) This goal was to make decisions in advance about future situations (to plan) and store a record so that we do the planned things when we get there

    - This requires a modifiable mapping from states to the combination weights

How *could* we set the combination weights?

- We could learn their settings as a function of state
- We could just learn them from experience in the normal way, as we try to learn an accurate overall value function

Conclusions

- RL seems to need some more structure to its action selections

- We seem to need abstract chunks on the *policy* level
    - Something larger than actions, but more combinatorial than the conventional notion of options

    - Something like a feature-based policy descriptor

- One idea is that the features correspond to subgoals or intentions

    - The subgoal keyboard is a simple idea along these lines