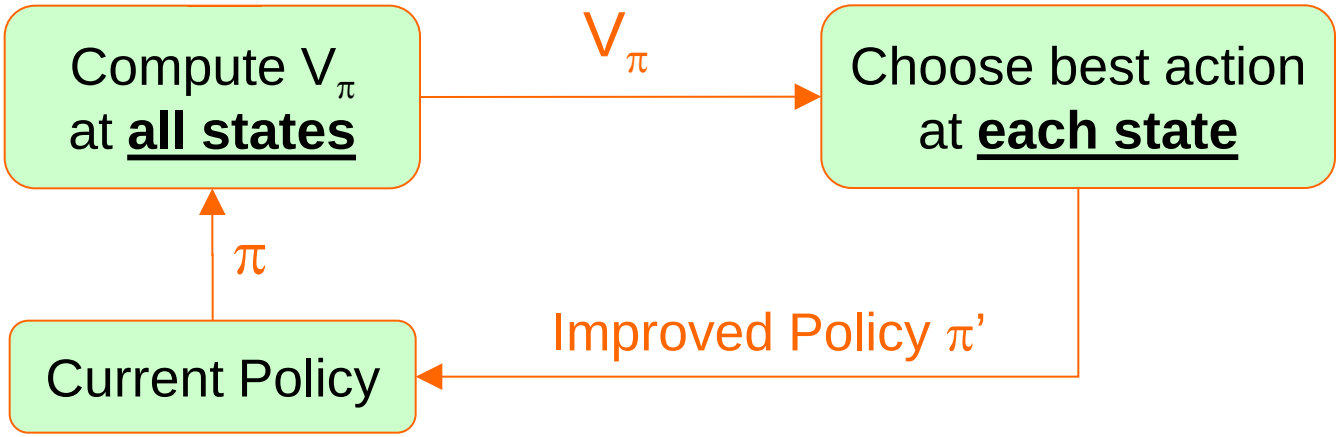
RL in Large State Spaces: Approximate Policy Iteration (aka Reduction to Classification Learning)*

* Slides based on Alan Fern

What about large state spaces?

- One approach is to map the original state space S to a much smaller state space S' via some hashing function.
  - Ideally "similar" states in S are mapped to the same state in S'
- Then do learning over S' instead of S.
  - Note that the world may not look Markovian when viewed through the lens of S', so convergence results may not apply
  - But, still the approach can work if a good enough S' is engineered (requires careful design), e.g.
  - *Empirical Evaluation of a Reinforcement Learning Spoken Dialogue System.* With S. Singh, D. Litman, M. Walker. *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000
- Many approaches for dealing with large state-spaces
  - Value function approximation (state or state-action => features)
  - Policy gradient methods
  - Least Squares Policy Iteration
  - **Approximate Policy Iteration – we will only cover this**

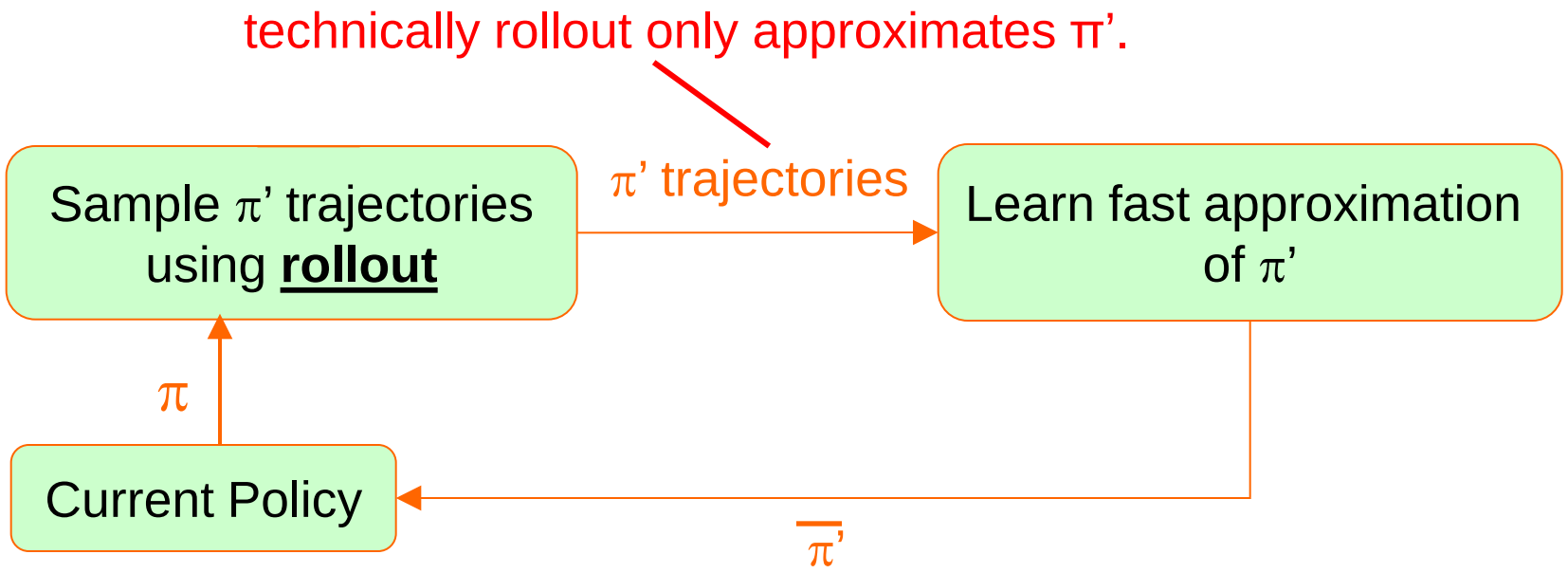
Return to Policy Iteration



Approximate policy iteration:

- Only computes values and improved action at some states.
- Uses those to infer a fast, compact policy over all states.

Approximate Policy Iteration

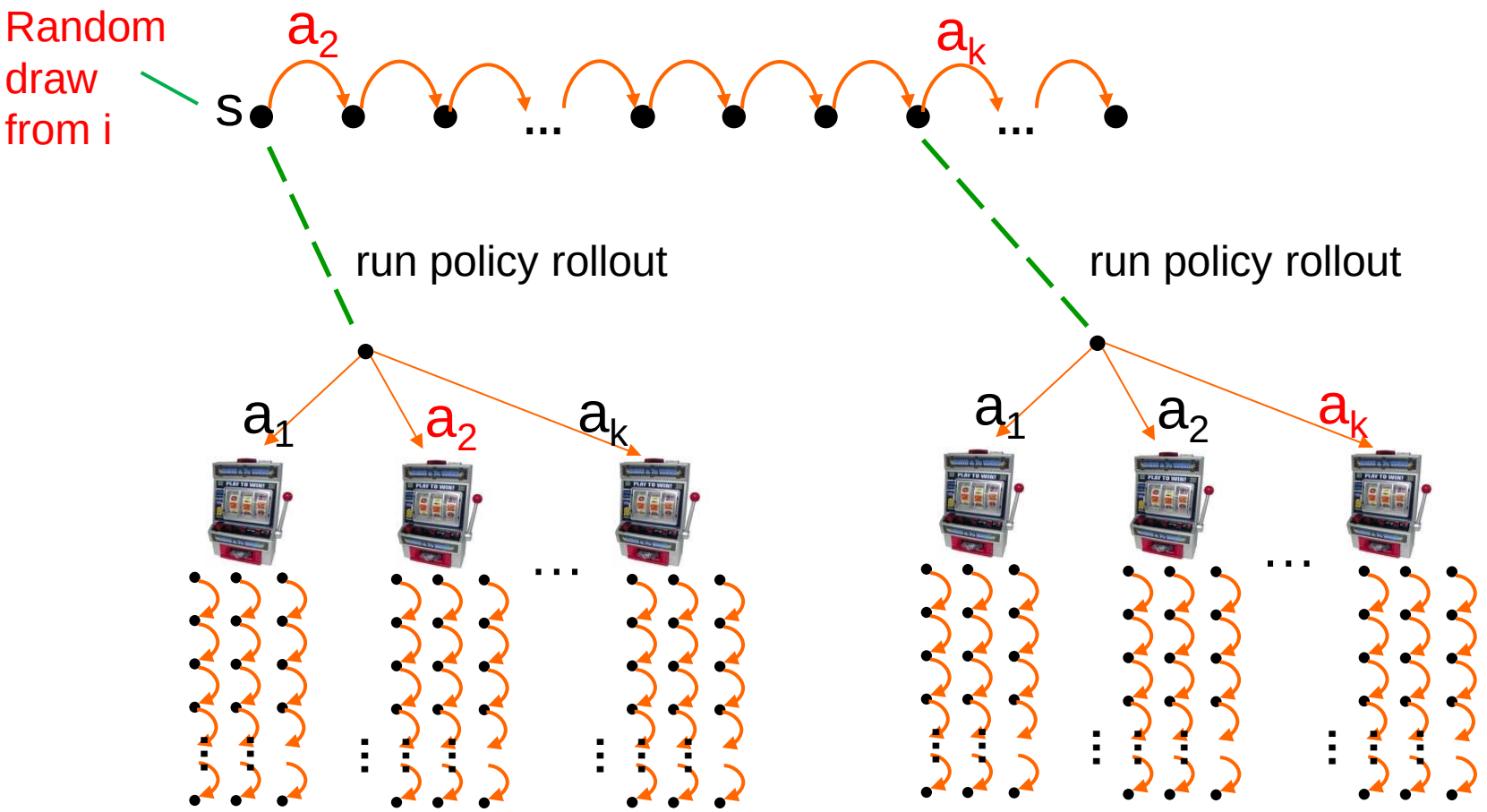


1. Generate trajectories of rollout policy
   (starting state of each trajectory is drawn from initial state distribution I)
2. "Learn a fast approximation" of rollout policy
3. Loop to step 1 using the learned policy as the base policy

What do we mean by generate trajectories?

Generating Rollout Trajectories

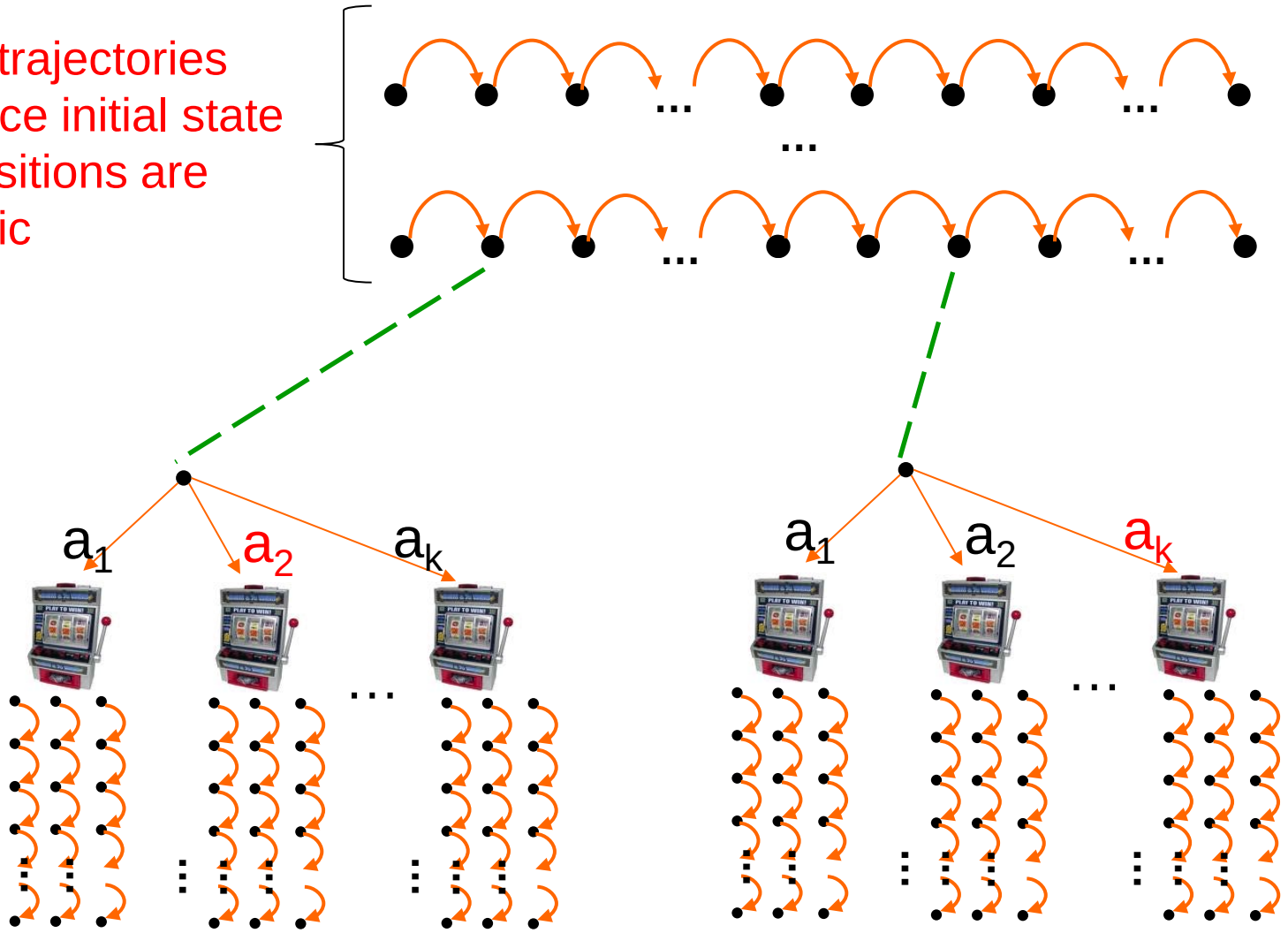
Get trajectories of current rollout policy from an initial state



Generating Rollout Trajectories

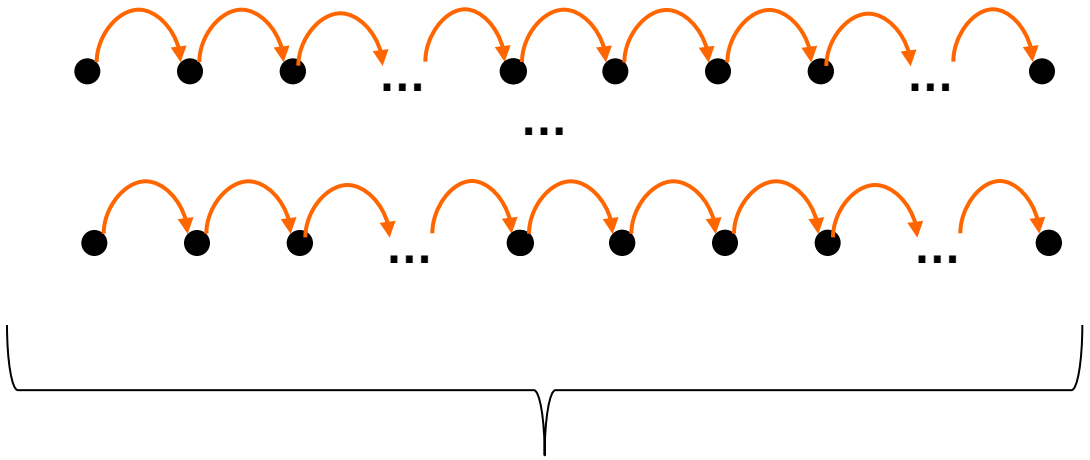
Get trajectories of current rollout policy from an initial state

Multiple trajectories differ since initial state and transitions are stochastic



Generating Rollout Trajectories

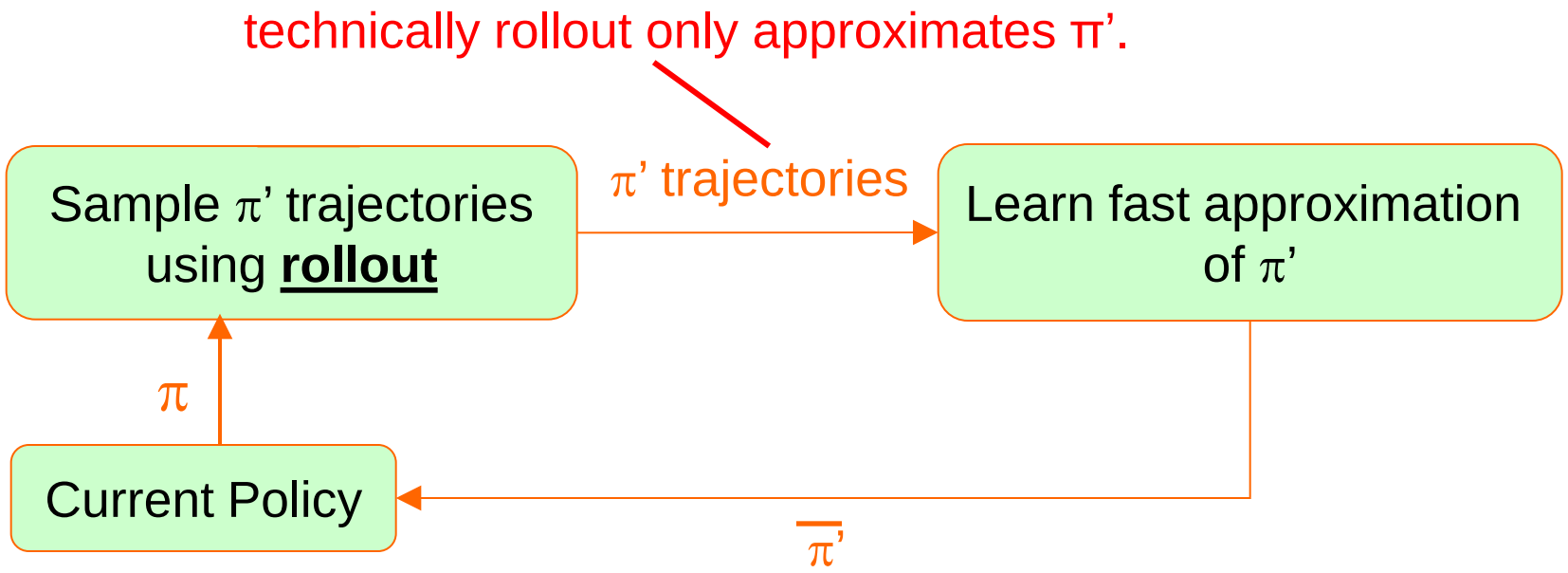
Get trajectories of current rollout policy from an initial state



Results in a set of state-action pairs giving the action selected by “improved policy” in states that it visits.

$$\{(s_1,a_1), (s_2,a_2),\ldots,(s_n,a_n)\}$$

Approximate Policy Iteration

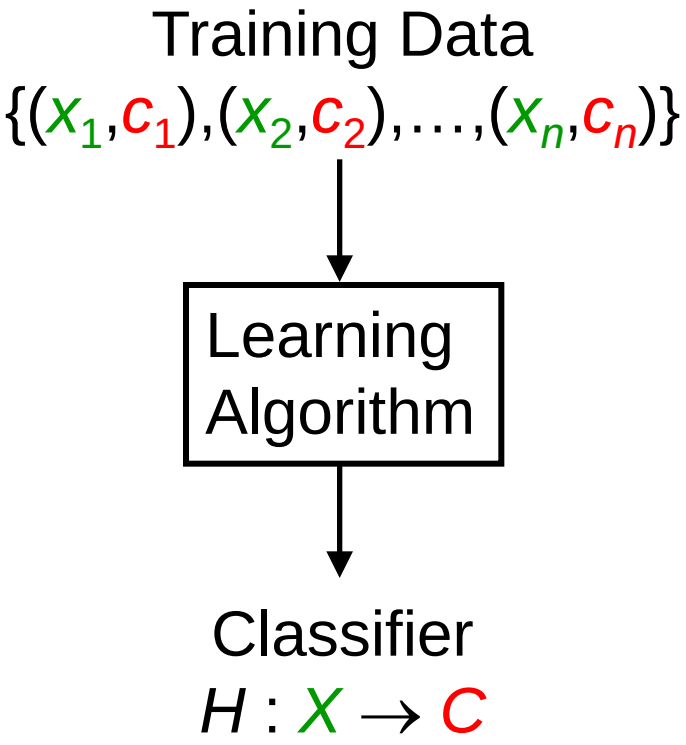


1. Generate trajectories of rollout policy
   (starting state of each trajectory is drawn from initial state distribution I)
2. "Learn a fast approximation" of rollout policy
3. Loop to step 1 using the learned policy as the base policy

What do we mean by "learn an approximation"?

Aside: Classifier Learning

- A **classifier** is a function that labels inputs with class labels.
- "Learning" classifiers from training data is a well studied problem (decision trees, support vector machines, neural networks, etc).



Example problem:

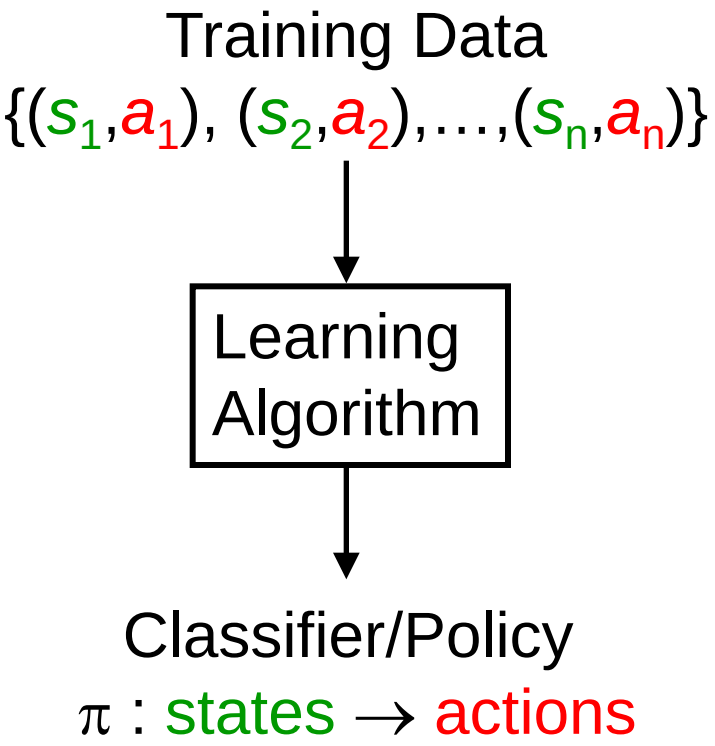
x_i - image of a face

$c_i \in$ {male, female}

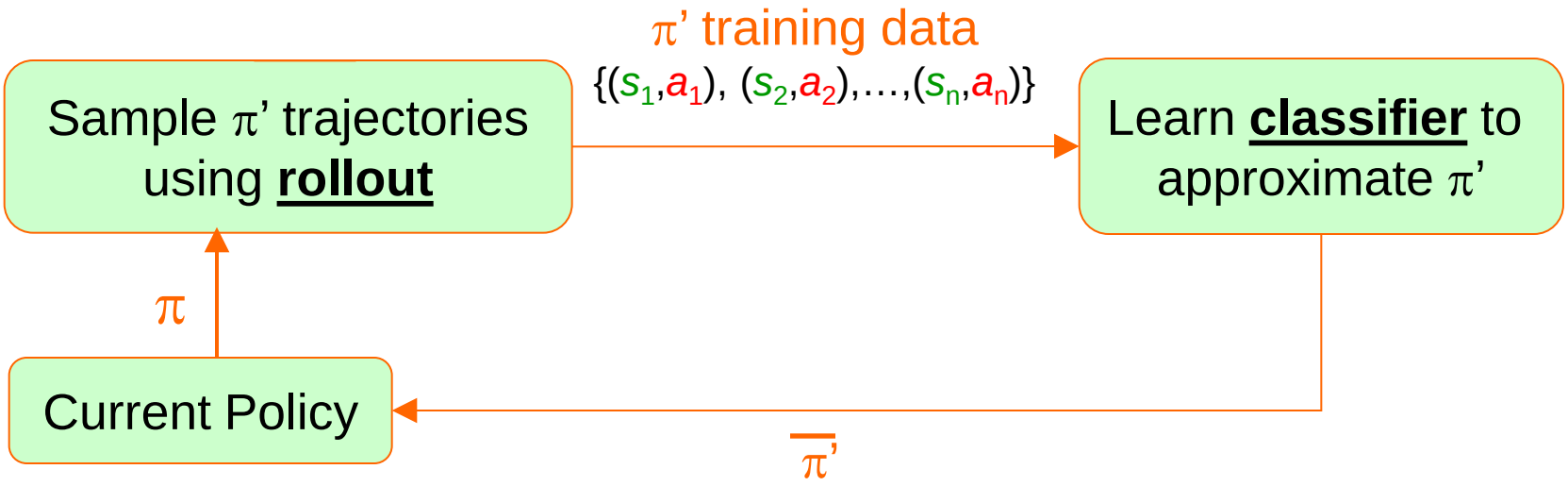
Aside: Control Policies are Classifiers

A **control policy** maps states and goals to actions.

π : states $\rightarrow$ actions



Approximate Policy Iteration

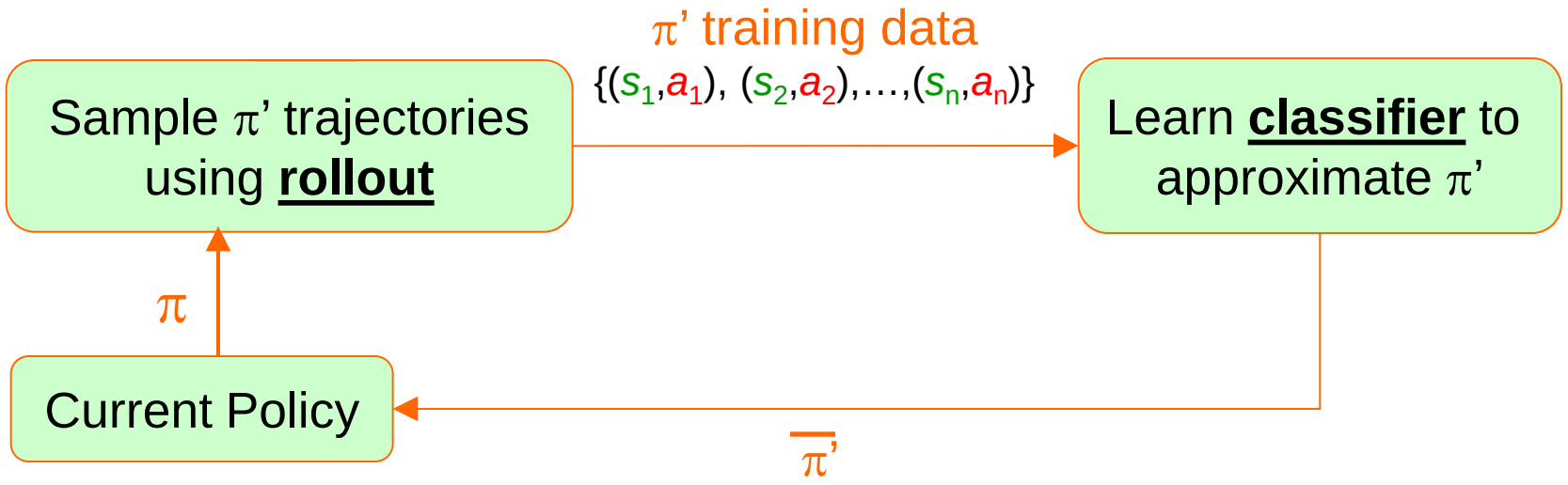


1. Generate trajectories of rollout policy
   Results in training set of state-action pairs along trajectories

$$T = \{(s_1,a_1), (s_2,a_2),\ldots,(s_n,a_n)\}$$

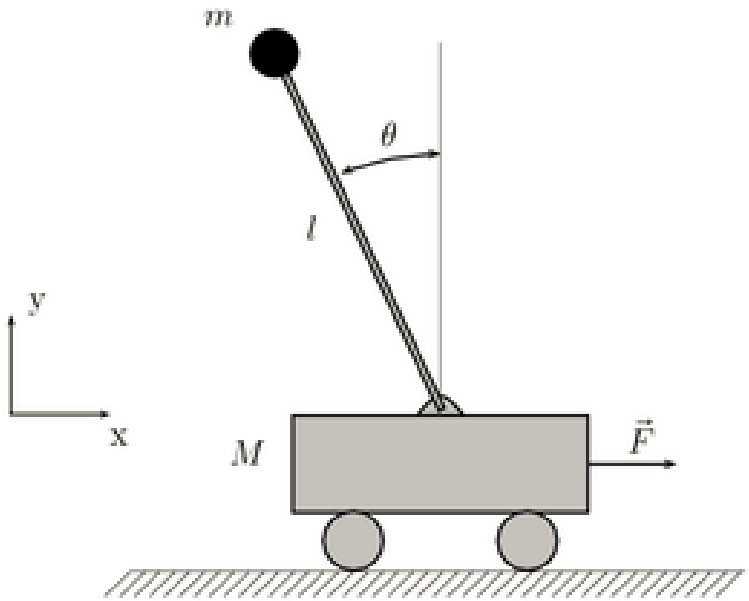
2. Learn a classifier based on T to approximate rollout policy
3. Loop to step 1 using the learned policy as the base policy

Approximate Policy Iteration



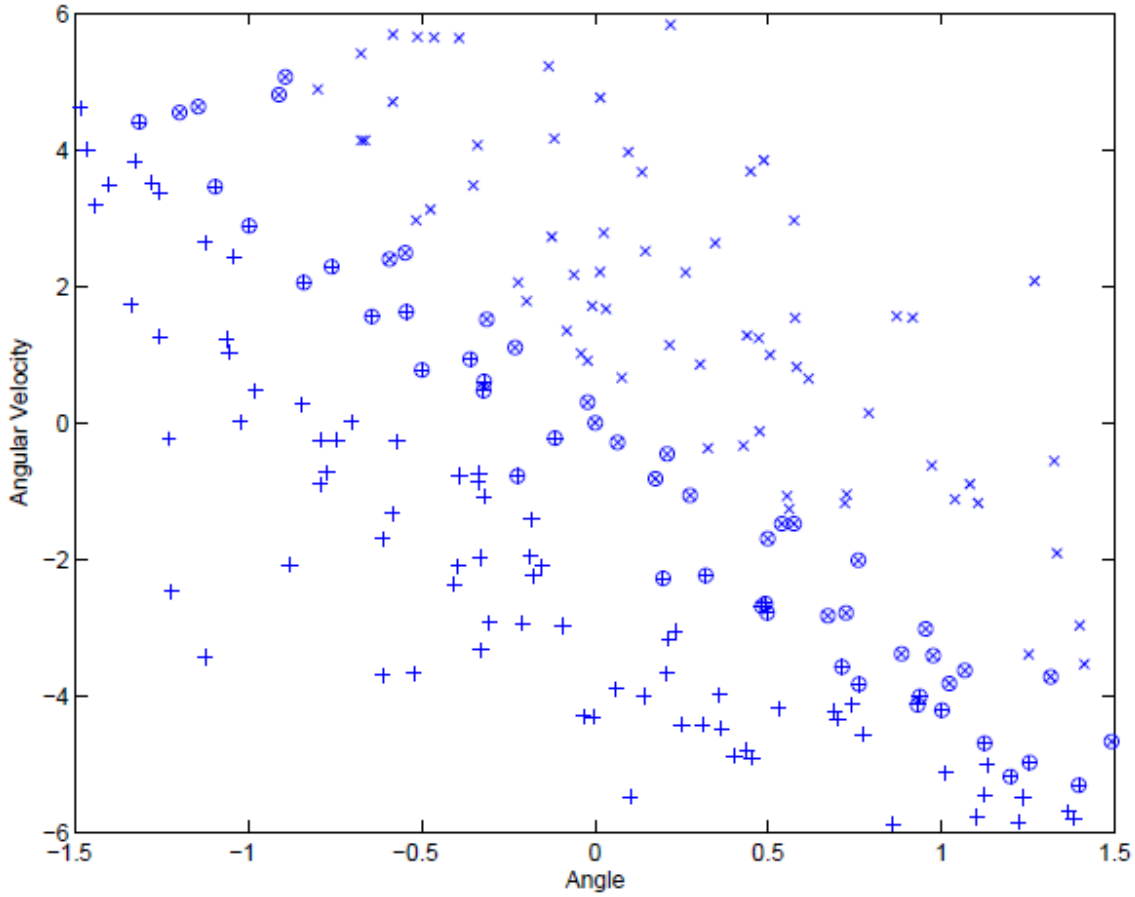
- The hope is that the learned classifier will capture the general structure of improved policy from examples
- Want classifier to quickly select correct actions in states outside of training data (classifier should generalize)
- Approach allows us to leverage large amounts of work in machine learning

API for Inverted Pendulum



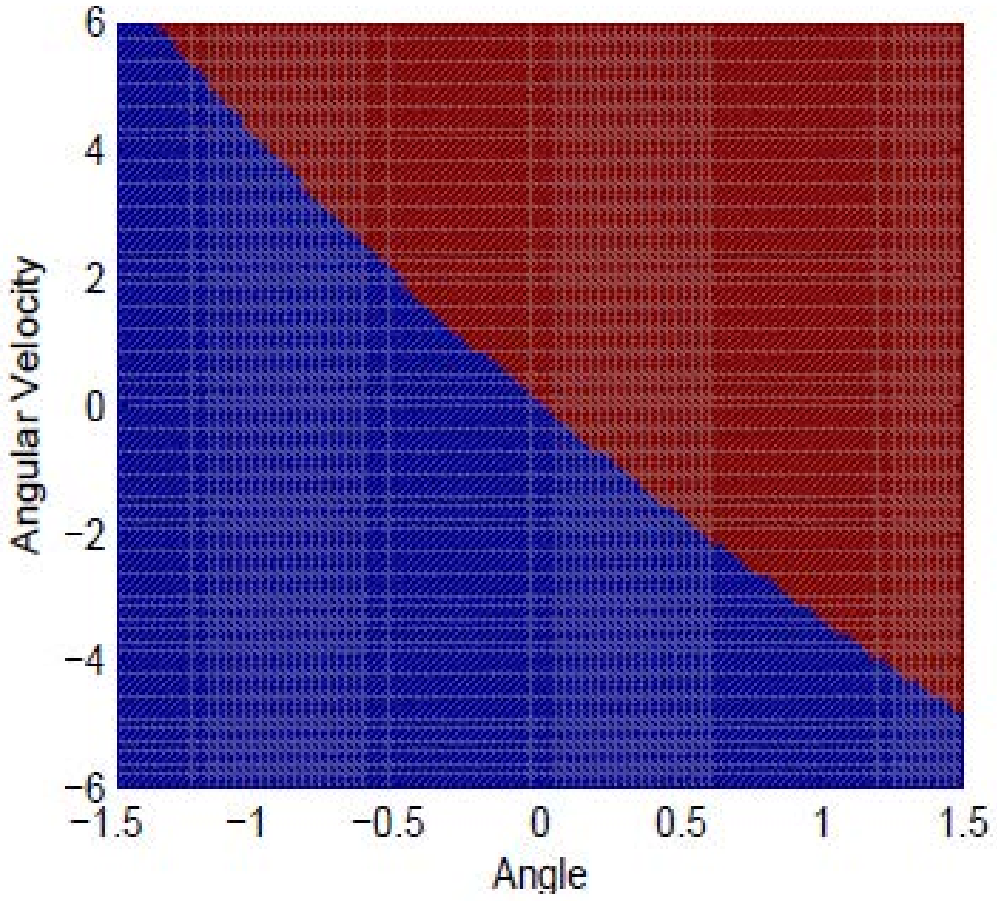
Consider the problem of balancing a pole by applying either a positive or negative force to the cart.
The state space is described by the velocity of the cart and angle of the pendulum.
There is noise in the force that is applied, so problem is stochastic.

Experimental Results



A data set from an API iteration. + is positive action, x is negative (ignore the circles in the figure)

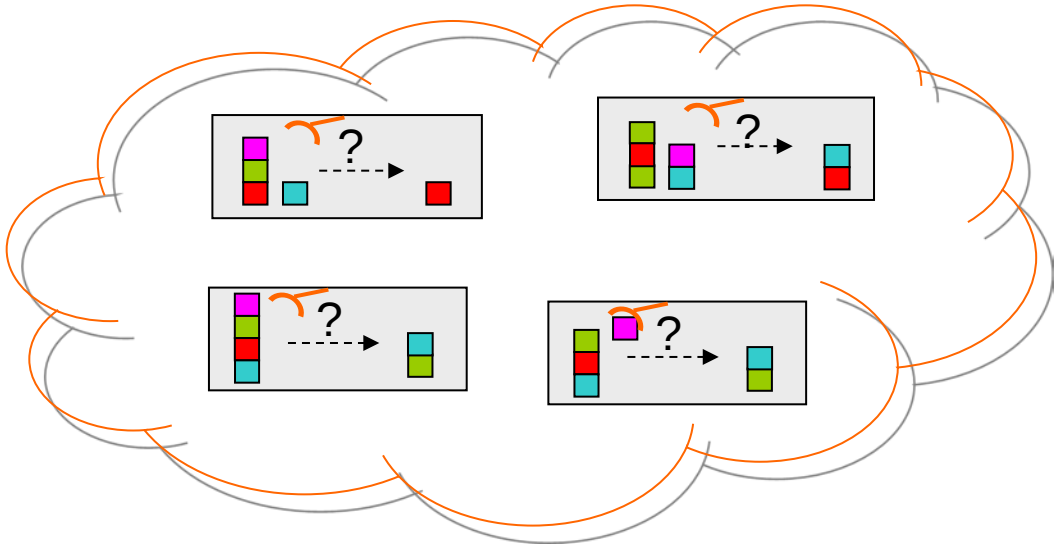
Experimental Results



Support vector machine used as classifier.
Maps any state to + or –

Learned classifier/policy after 2 iterations: (near optimal)
blue = positive, red = negative

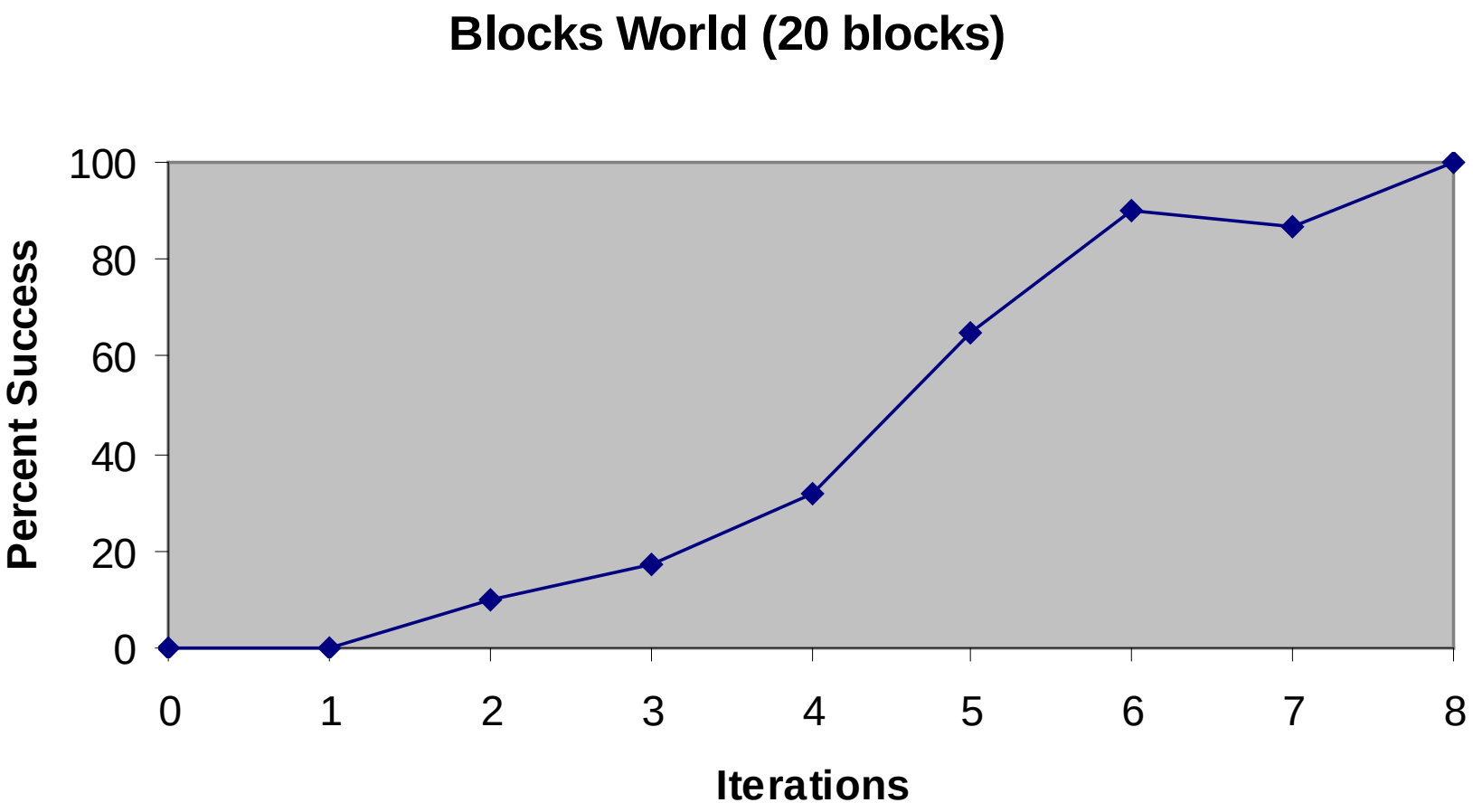
API for Stacking Blocks



Consider the problem of form a goal configuration of blocks/crates/etc. from a starting configuration using basic movements such as pickup, putdown, etc.

Also handle situations where actions fail and blocks fall.

Experimental Results



The resulting policy is fast near optimal. These problems are very hard for more traditional planners.

Summary of API

- Approximate policy iteration is a practical way to select policies in large state spaces
- Relies on ability to learn good, compact approximations of improved policies (must be efficient to execute)
- Relies on the effectiveness of rollout for the problem
- There are only a few positive theoretical results
  - convergence in the limit under strict assumptions
  - PAC results for single iteration
- But often works well in practice