

# Sparse Memory Structures Detection

Final Project for COMP 652  
Alexandre Bouchard-Côté

# The Problem

- Reinforcement learning algorithms' memory requirements tend to explode when the dimensionality of the state space increase. :(
- Most interesting tasks have high dimensionality. :(
- This is known as the **Curse of Dimensionality**.

# A Solution

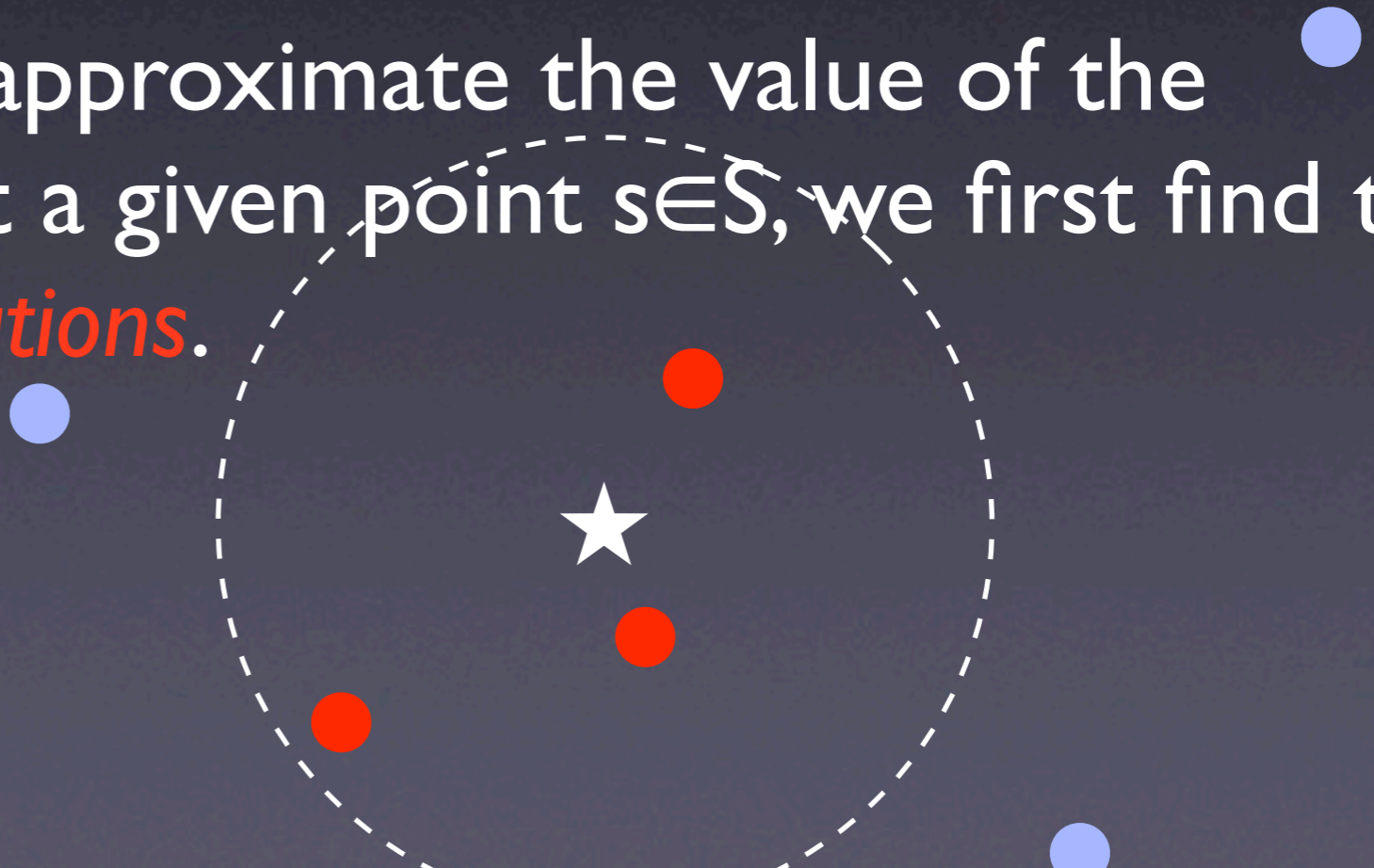
- Not all states are equal! Good approximation architectures should attempt to locate regions of the state space that are “more important” and allocate proportionally more memory resources to model them accurately.
- This is what the Sparse Distributed Memories architecture [1] tries to do.

---

[1] Kanerva, P. (1993). Sparse distributed memory and related models. In M. Hassoun (Ed.), *Associated neural memories: Theory and implementation*, 50-76. N.Y.: Oxford University Press.

# SDM in 2 nutshells

- A SDM architecture is equipped with:
  - a *similarity function*  
 $\sigma : S \times S \rightarrow [0, 1]$  s.t.  $(1 - \sigma)$  is a metric on  $S$
  - A set of *basis points or hard locations*  
 $B := \{s_1, \dots, s_K : s_i \in S\}$  with weights  $w_i$ .
- When we want to approximate the value of the targeted function at a given point  $s \in S$ , we first find the set  $H$  of *actived locations*.



# SDM in 2 nutshell

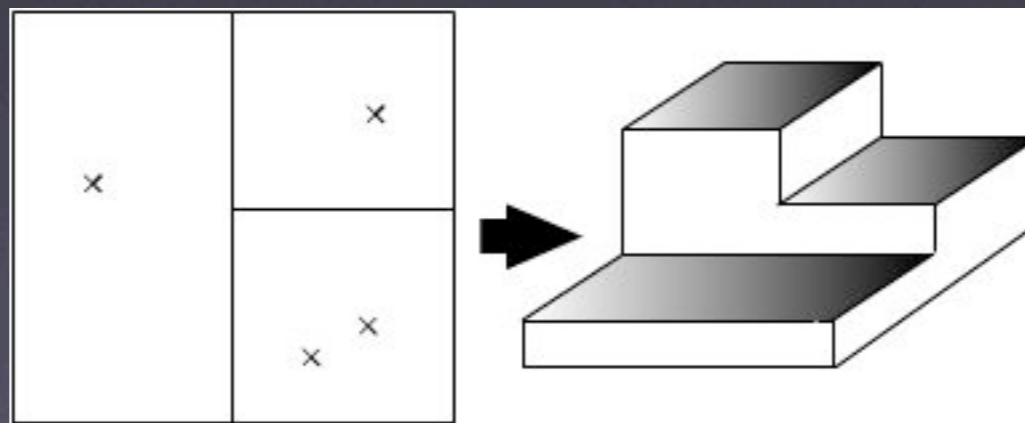
- Then the weights corresponding to the active locations are summed: (this rule simply gives more importance to the locations that are close).
$$\frac{\sum \sigma(s_k, s) w_k}{\sum \sigma(s_k, s)}$$
- Training is done using canonical gradient descent.

# My Project

- When the SDM is ran, the positions of the hard locations is determined dynamically and (hopefully) the architecture detects the interesting states.
- A step towards understanding and discovering good algorithms for dynamic memory allocation would be **to verify that the distribution of hard location obtained by existing algorithms is not just a plain uniform distribution.**

# The Ruler

- In order to do that, we need to estimate the underlying probability density function.
- Difficulty: need for variable resolution.
- Solution: an approach based on the same idea as kd-tree.



# The Results

- The probability density function estimator works very well.
- However, the first application of this tool to our problem gives at the first glance a disappointing result: L2 distance from uniform distribution close to 0. :(
- A qualitative analysis of the distribution of the locations suggests that it is caused by a too small state space: all is not lost :)

