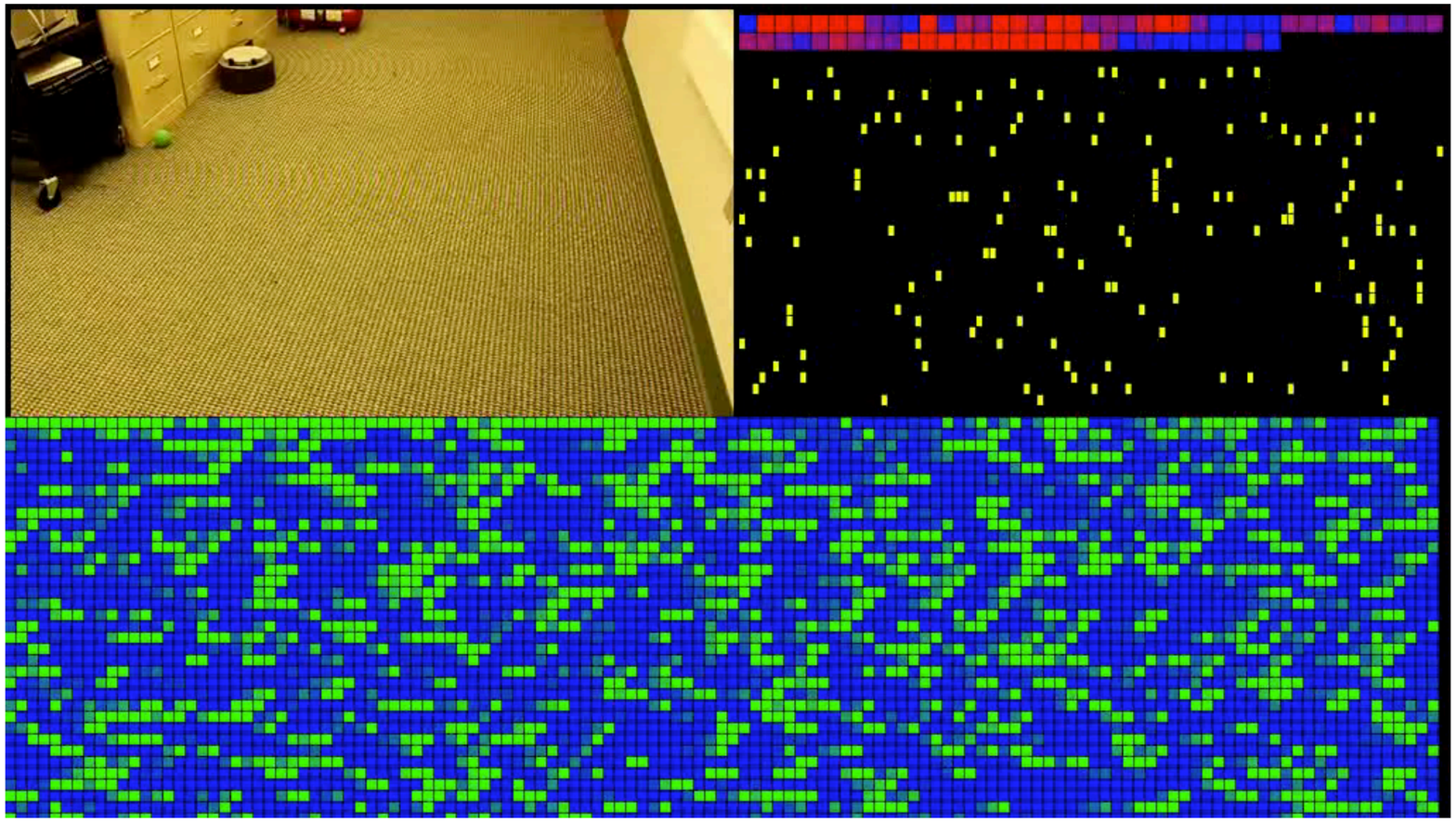
Constructing Features

CMPUT 655
Fall 2022

Robots, Never-ending RL, TD-learning, Tile coding

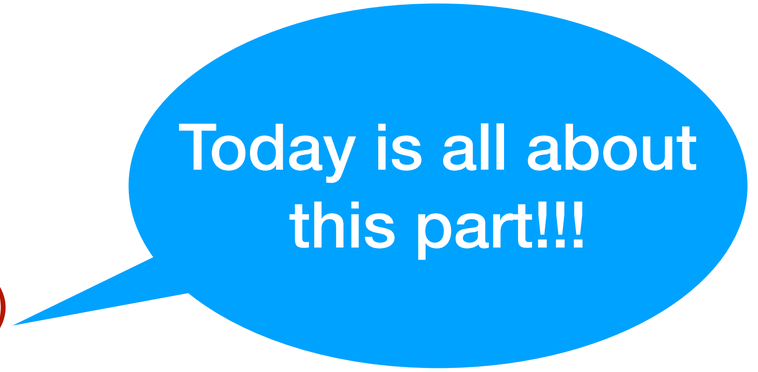


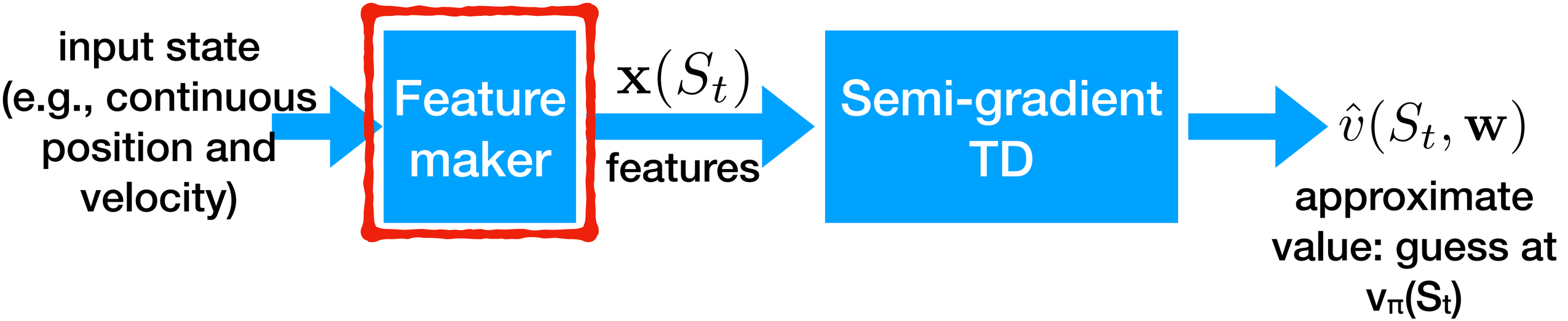
How to make an RL agent: tabular setting

- Pick an algorithm that is a good match for your problem (perhaps you choose Sarsa because it's an episodic cost-to-goal grid world task)
- Decide an exploration method (e.g., e-greedy or optimistic initial values)
- Decide how to initialize the value function (setting Q_0)
- Decide how to set the other parameters (alpha, epsilon, etc)
- Decide how to run the experiment: number of episodes, steps per episodes, what to plot

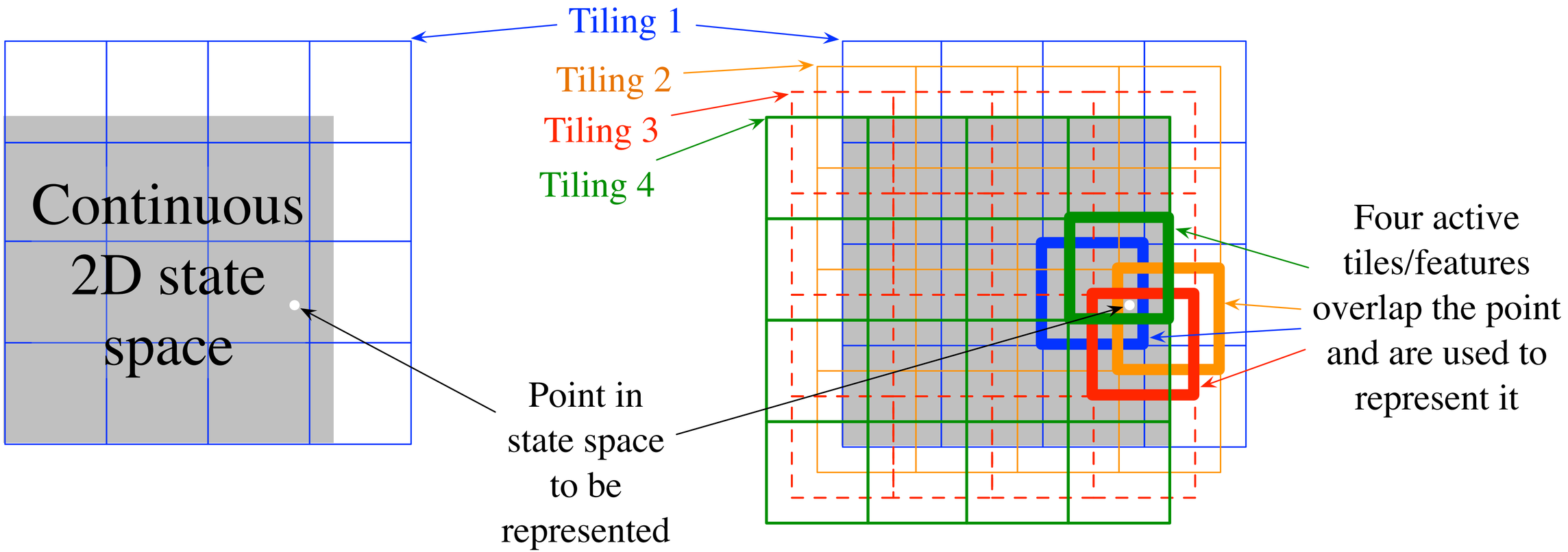
How to make an RL agent: function approximation setting

- Pick an algorithm that is a good match for your problem
- Decide on linear or non-linear function approximation
- Decide on the function approximator (tile coding, neural network)
  - AND the parameters of the function approximator.
  - **This impacts all the following choices …**
    - Decide on an exploration method (e.g., e-greedy)
    - Decide how to initialize the value function (setting Q_0)
    - Decide how to set the other parameters (alpha, epsilon, etc)
    - Decide how to run the experiment: number of episodes, steps per episodes, what to plot



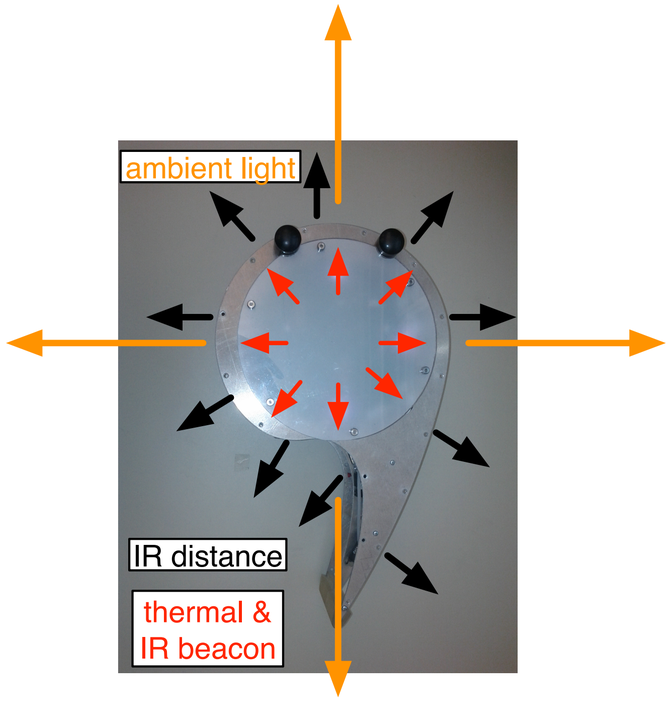

input state
(e.g., continuous
position and
velocity)
Feature
maker
$\mathbf{x}(S_t)$
features
Semi-gradient
TD
$\hat{v}(S_t, \mathbf{w})$
approximate
value: guess at
$v_\pi(S_t)$

More on tile coding



Tile coding on a robot

- Use multiple independent tilings of individual and pairs of sensors to create on large feature vector.

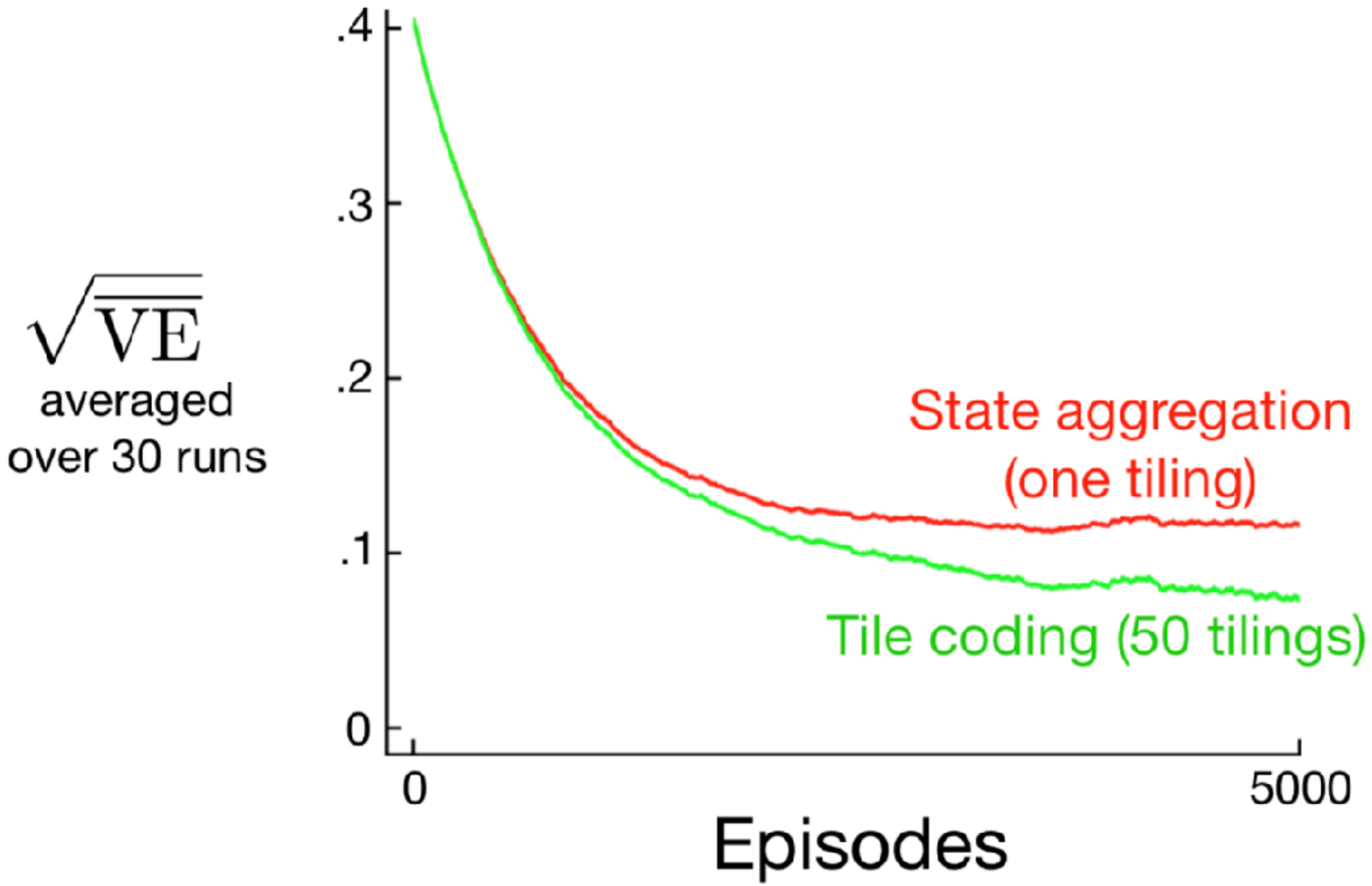


Sensor type	Num of sensors	tiling type	Num of intervals	Num of tilings
IRdistance	10	1D	8	8
		1D	2	4
		2D	4	4
		2D+1	4	4
Light	4	1D	4	8
		2D	4	1
IRlight	8	1D	8	6
		1D	4	1
		2D	8	1
		2D+1	8	1
Thermal	4(8)	1D	8	4
RotationalVelocity	1	1D	8	8
Magnetic	3	1D	8	8
Acceleration	3	1D	8	8
MotorSpeed	3	1D	8	4
		2D	8	8
MotorVoltage	3	1D	8	2
MotorCurrent	3	1D	8	2
MotorTemperature	3	1D	4	4
LastMotorRequest	3	1D	6	4
OverheatingFlag	1	1D	2	4

$$|\boldsymbol{x}| = 6065$$

Which approach do you think will work better on the Random Walk?

- State aggregation?
- Tile Coding?
- A NN?



The Bitter lesson

- http://www.incompleteideas.net/IncIdeas/BitterLesson.html

- “The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin….

- …In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on **massive, deep search**. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had **pursued methods that leveraged human understanding of the special structure of chess**….

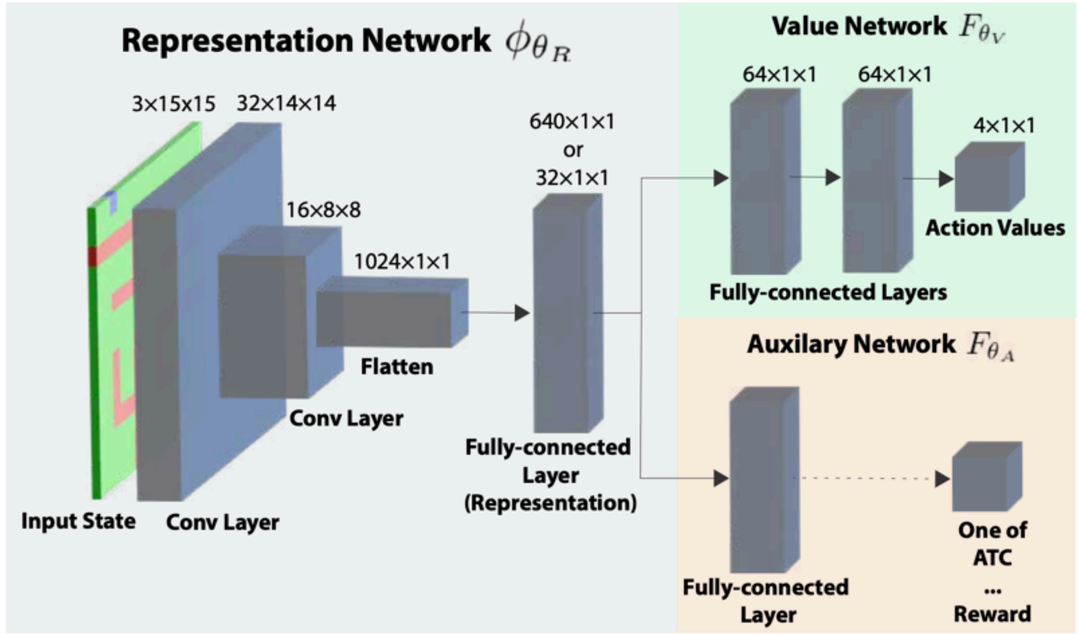
- … **The bitter lesson** is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning….”

Big Agent, Massive World

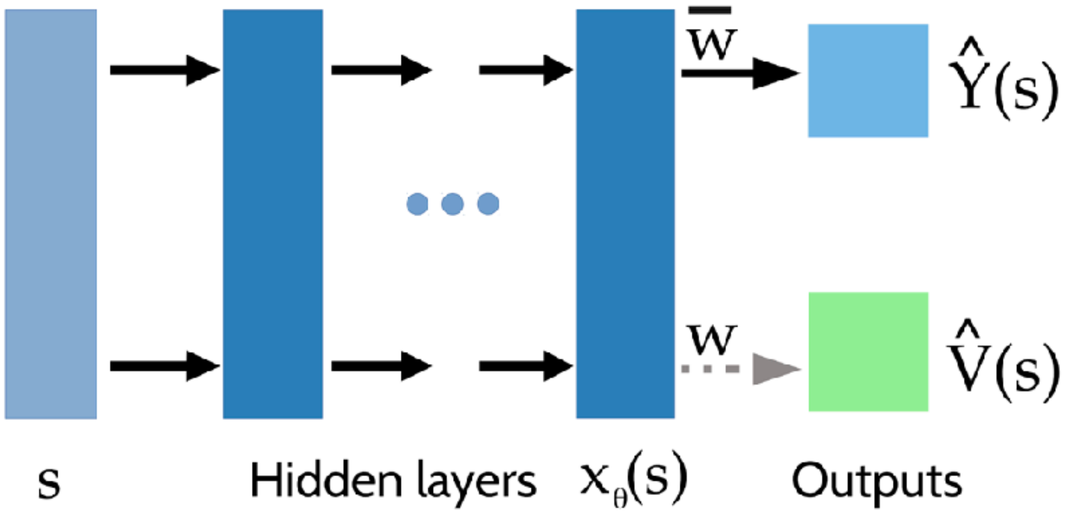
- Our current agents are huge!
- But the world is so much larger than any agent we can build
- However, our current agents operate in the over-parameterize regime
  - The number of parameterizes in the function approximator exceeds the training data (or number of states)
  - We use massive clusters to train huge architectures! And it works well it seems
- Will the continue to be the case as we move toward more ambitious applications? Towards AI?

Discussion posts

- There are a great variety of network architectures possible
  - Auxiliary task learning
  - Two-timescale networks
  - Cascade correlation nets
  - Tile-coding inputs to the NN
  - Etc



- There are a great variety of network architectures possible
  - Auxiliary task learning
  - Two-timescale networks
  - Cascade correlation nets
  - Tile-coding inputs to the NN
  - Etc



- There are a great variety of network architectures possible
  - Auxiliary task learning
  - Two-timescale networks
  - Cascade correlation nets
  - Tile-coding inputs to the NN
  - Etc

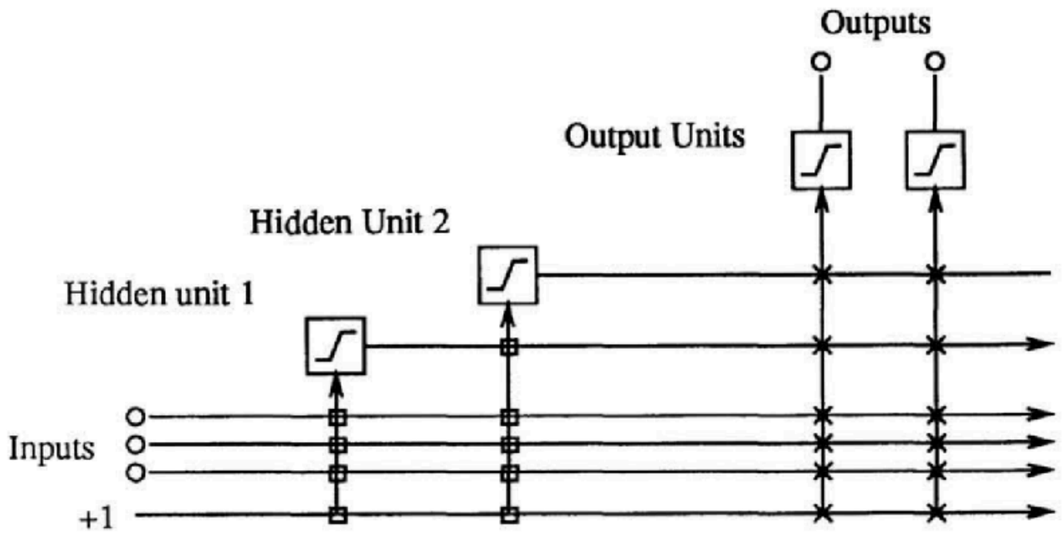


Figure 1: The Cascade architecture, after two hidden units have been added. The vertical lines sum all incoming activation. Boxed connections are frozen, X connections are trained repeatedly.

- There are a great variety of network architectures possible
  - Auxiliary task learning
  - Two-timescale networks
  - Cascade correlation nets
  - Tile-coding inputs to the NN
  - Etc

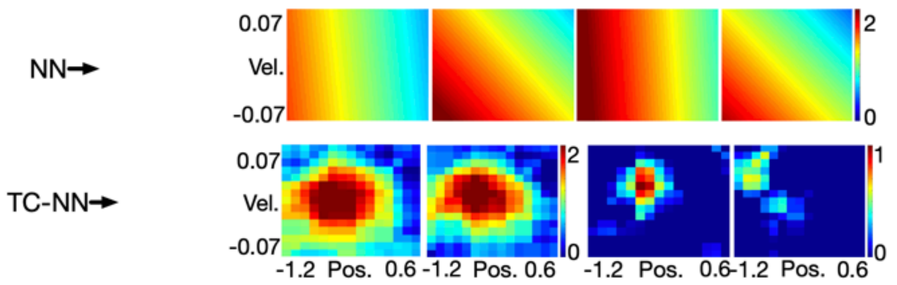


Figure 2: Response functions with raw inputs (top) and tile coding preprocessing (bottom) for Mountain Car control.

Appeared in *Proceedings of the Tenth Int. Conf. on Machine Learning*, pp. 314–321, Morgan Kaufmann, 1993.

Online Learning with Random Representations

Richard S. Sutton and Steven D. Whitehead
GTE Laboratories Incorporated
40 Sylvan Road
Waltham, MA 02254
sutton@gte.com swhitehead@gte.com

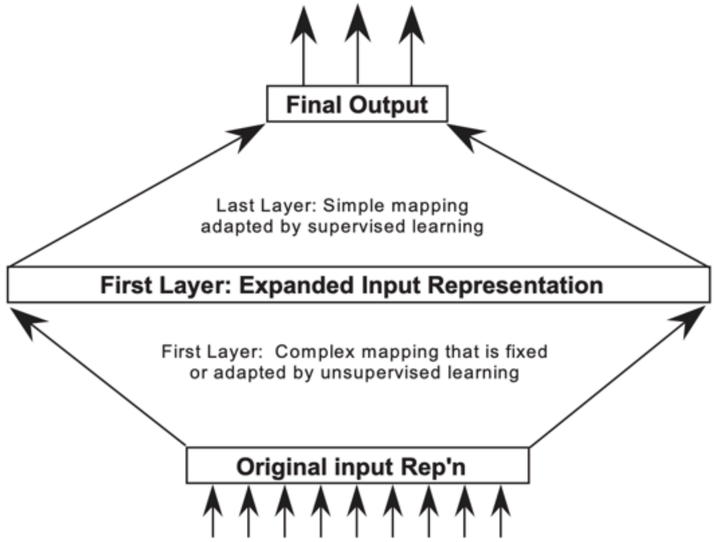


Figure 1: General architecture for learning with an Expanded Representation (ER).

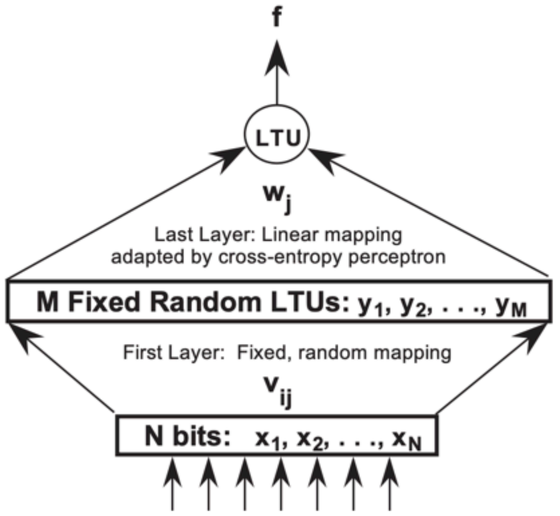
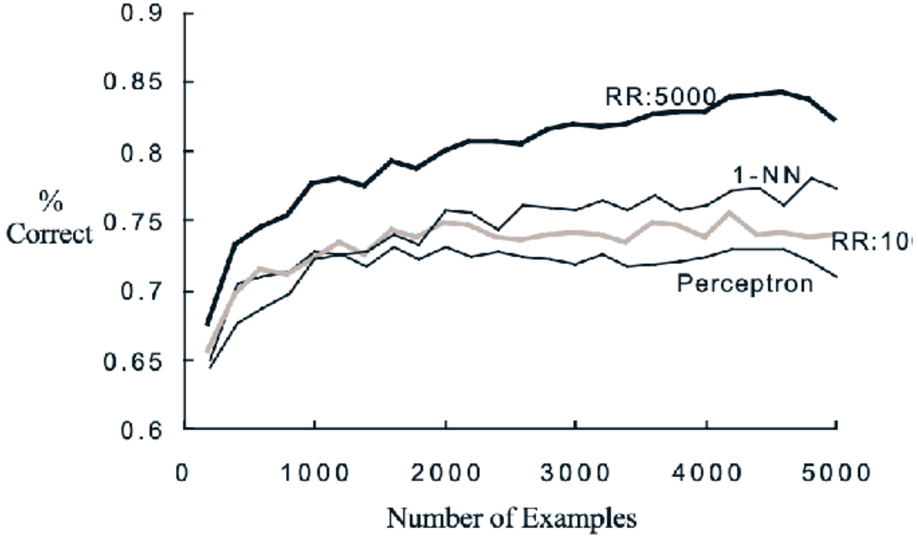
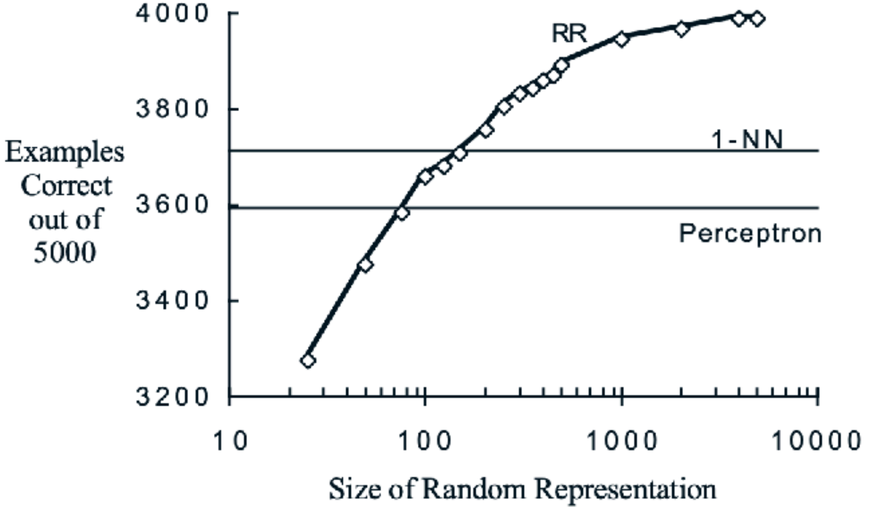


Figure 2: The basic Random Representation (RR) learning system used in this paper.





- What about overfitting in RL with Function Approximation
  - When could this happen?
- How to think about bias, injecting prior knowledge, relying on people in the design of our function approximators
  - Lets think about CNNs vs RNNs
- Related, isn't setting the hyper-parameters of tile-coding similar to building in prior knowledge
  - We hope to have general approaches to constructing tile coders
- If the main function is online RL, how to we build function approximators that are well suited for that setting?
  - Can we do some offline learning?

- What if you tile coded like this or that? Would it be better?
  - Try it out!
- If the main function is online RL, how to we build function approximators that are well suited for that setting?
  - Can we do some offline learning?
- What about non-parametric methods (ie kernel methods)
  - Locally weighted regression, lazy learning, instance based learning, prototypes of the input
    - Tracing the inputs, “imprinting”
  - Learning the distance metric is really hard, scaling is an issue

- NN’s only work with iid training data right? That is why we use experience replay
  - I am not so sure. See Lin’s original experience replay paper
- Should we make use of training / testing sets in RL
  - How would this arise and a life-long learning system?
- Vanishing gradients, exploding gradients, loss of plasticity, forgetting interference
  - Perhaps we need different architectures and training algorithms: sparse networks, continual back-prop, different activation functions (cReLu)
  - Generate and test?
    - https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.708.2404&rep=rep1&type=pdf

- Does anyone use course coding (the circles)?
  - ?
- What about adaptive tile coding
  - Shimon Whiteson’s Phd thesis
- What about RNNs, they seem more general and powerful
  - Totally! Training them is still a challenge
  - Mozer, M. C. (1993). Neural network architectures for temporal pattern processing. In A. S. Weigend & N. A. Gershenfeld (Eds.), Time series prediction: Forecasting the future and understanding the past (pp. 243–264). Redwood City, CA: Sante Fe Institute Studies in the Sciences of Complexity, Proceedings Volume XVII, Addison-Wesley Publishing.

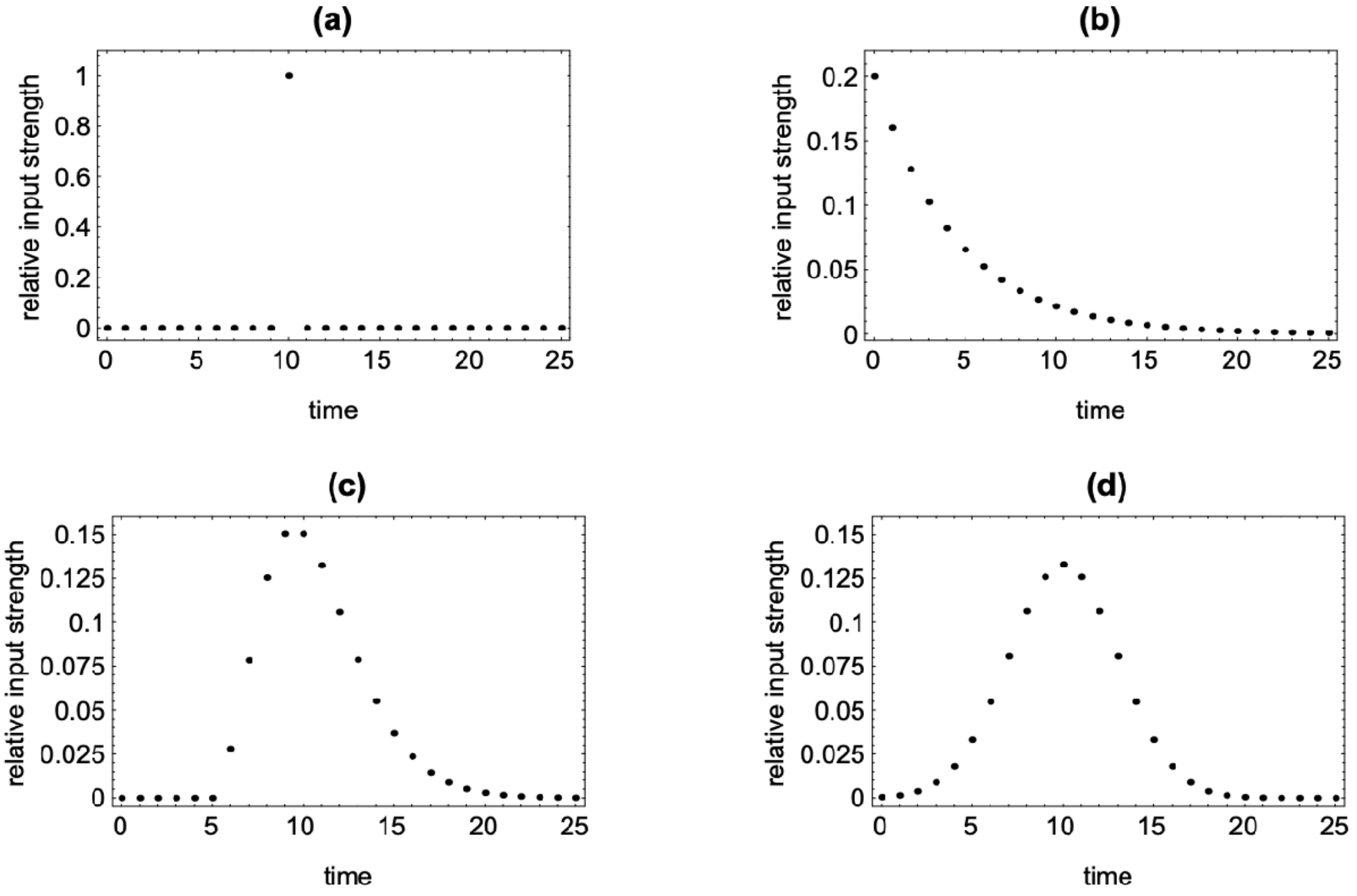


Figure 3: The kernel functions for (a) a delay-line memory, $\omega = 10$, (b) an expontial trace memory, $\mu = .8$, (c) a gamma memory, $\omega = 6$ and $\mu = .4$, and (d) a gaussian memory.

Q1

Assume we are given a fixed policy π. Recall that the mean-squared value error is

$$\overline{\text{VE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} \mu(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2.$$

Recall that we can use TD with linear function approximation to learn parameters $\mathbf{w}$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R_{t+1} + \gamma \mathbf{w}^\top \mathbf{x}(S_{t+1}) - \mathbf{w}^\top \mathbf{x}(S_t)]\mathbf{x}(S_t).$$

When linear TD converges, it converges to what we call the TD fixed-point. Let's denote the weight vector found by linear TD at convergence as $\mathbf{w}_{\text{TD}}$. We denote the estimated value as $\hat{v}(s, \mathbf{w}_{\text{TD}}) = \mathbf{w}_{\text{TD}}^\top \mathbf{x}(s)$. At the TD fixed point, we know that the VE is within a bounded expansion of the lowest possible error:

$$\overline{\text{VE}}(\mathbf{w}_{\text{TD}}) \leq \frac{1}{1-\gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$$

(a) Recall that $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$ is the minimal value error you can achieve under this value function parameterization. If we have a tabular parameterization, then what might $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$ be? What if the parameterization is a state aggregation?

Q1

Assume we are given a fixed policy π. Recall that the mean-squared value error is

$$\overline{\text{VE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} \mu(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2.$$

Recall that we can use TD with linear function approximation to learn parameters $\mathbf{w}$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R_{t+1} + \gamma \mathbf{w}^\top \mathbf{x}(S_{t+1}) - \mathbf{w}^\top \mathbf{x}(S_t)]\mathbf{x}(S_t).$$

When linear TD converges, it converges to what we call the TD fixed-point. Let's denote the weight vector found by linear TD at convergence as $\mathbf{w}_{\text{TD}}$. We denote the estimated value as $\hat{v}(s, \mathbf{w}_{\text{TD}}) = \mathbf{w}_{\text{TD}}^\top \mathbf{x}(s)$. At the TD fixed point, we know that the VE is within a bounded expansion of the lowest possible error:

$$\overline{\text{VE}}(\mathbf{w}_{\text{TD}}) \leq \frac{1}{1-\gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$$

(b) If $\gamma = 0.9$ and $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}) = 0$, then what is the minimum and maximum value of $\overline{\text{VE}}(\mathbf{w}_{\text{TD}})$?

Q1

Assume we are given a fixed policy π. Recall that the mean-squared value error is

$$\overline{\text{VE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} \mu(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2.$$

Recall that we can use TD with linear function approximation to learn parameters $\mathbf{w}$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R_{t+1} + \gamma \mathbf{w}^\top \mathbf{x}(S_{t+1}) - \mathbf{w}^\top \mathbf{x}(S_t)]\mathbf{x}(S_t).$$

When linear TD converges, it converges to what we call the TD fixed-point. Let's denote the weight vector found by linear TD at convergence as $\mathbf{w}_{\text{TD}}$. We denote the estimated value as $\hat{v}(s, \mathbf{w}_{\text{TD}}) = \mathbf{w}_{\text{TD}}^\top \mathbf{x}(s)$. At the TD fixed point, we know that the VE is within a bounded expansion of the lowest possible error:

$$\overline{\text{VE}}(\mathbf{w}_{\text{TD}}) \leq \frac{1}{1-\gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$$

(c) If $\gamma = 0.9$ and $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}) = 1$, then what is the minimum and maximum value of $\overline{\text{VE}}(\mathbf{w}_{\text{TD}})$?

(d) If $\gamma = 0.99$ and $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}) = 1$, then what is the minimum and maximum value of $\overline{\text{VE}}(\mathbf{w}_{\text{TD}})$?

Q1

Assume we are given a fixed policy π. Recall that the mean-squared value error is

$$\overline{\text{VE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} \mu(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2.$$

Recall that we can use TD with linear function approximation to learn parameters $\mathbf{w}$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R_{t+1} + \gamma \mathbf{w}^\top \mathbf{x}(S_{t+1}) - \mathbf{w}^\top \mathbf{x}(S_t)]\mathbf{x}(S_t).$$

When linear TD converges, it converges to what we call the TD fixed-point. Let's denote the weight vector found by linear TD at convergence as $\mathbf{w}_{\text{TD}}$. We denote the estimated value as $\hat{v}(s, \mathbf{w}_{\text{TD}}) = \mathbf{w}_{\text{TD}}^\top \mathbf{x}(s)$. At the TD fixed point, we know that the VE is within a bounded expansion of the lowest possible error:

$$\overline{\text{VE}}(\mathbf{w}_{\text{TD}}) \leq \frac{1}{1-\gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$$

(e) We have seen that if we can perfectly represent the value function, then $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}) = 0$. How about the other direction: if $\min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}) = 0$, then does that mean we can represent true value function? Consider **two cases**: (1) where $\mu(s)$ can be zero for some states, and (2) where $\mu(s) > 0$.