Part II:

Approximate Solution Methods

Approximate Solution Methods

- Extend the tabular methods in Part I to apply to problems with arbitrarily large state spaces.
  - The goal is instead to find a good approximate solution using limited computational resources.
- Combine RL with existing ***generalization*** methods
  - function approximation from supervised learning
- RL with function approximation involves a number of new issues that do not normally arise in conventional supervised learning:
  - nonstationarity, bootstrapping, and delayed targets.

Chapter 9: On-policy Prediction with Approximation

- Problem: approximating v_π from experience generated using a known policy π.
  - The approximate value function is parameterized by weight vector $\boldsymbol{w} \in \mathbb{R}^n$: $\hat{v}\left(s, \boldsymbol{w}\right) \approx v_\pi\left(s\right), n \ll |\mathcal{S}|$
  - Changing any component of $\boldsymbol{w}$ will have an effect on more than one state at a time.
- View each backup $s \mapsto g$ as a conventional training example – apply supervised learning methods.
  - Requires ability to handle nonstationary target functions – not all methods are equally well suited for use in RL.
- Objective: Mean Squared Value Error (MSVE):

$$\mathrm{MSVE}\left(\boldsymbol{w}\right) \doteq \sum_{s \in \mathcal{S}} \mu\left(s\right)\left[v_\pi\left(s\right) - \hat{v}\left(v, \boldsymbol{w}\right)\right]^2$$

  - The weighting $\mu\left(s\right) \geq 0$ is typically chosen as the on-policy distribution: the fraction of time spent in s under the target policy π.

Stochastic-gradient Methods

- Observe example $S_t \mapsto U_t$, a state with an approximation of its value $v_\pi(S_t)$.
- $\boldsymbol{w}_{t+1} \doteq \boldsymbol{w}_t - \frac{1}{2}\alpha\nabla\left[v_\pi(S_t) - \hat{v}(S_t\boldsymbol{w}_t)\right]^2 \doteq \boldsymbol{w}_t + \alpha\left[U_t - \hat{v}(S_t, \boldsymbol{w}_t)\right]\nabla\hat{v}(S_t, \boldsymbol{w}_t)$
- If $\mathbb{E}[U_t] = v_\pi(S_t), \forall t$ (unbiased estimate), $\boldsymbol{w}_t$ is guaranteed to converge to a local optimum under the usual stochastic approximation conditions for decreasing α.

Gradient Monte Carlo Algorithm for Approximating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^n \mapsto \mathbb{R}$

Initialize value-function weights $\boldsymbol{w}$ as appropriate (e.g., $\boldsymbol{w} = \mathbf{0}$)
Repeat forever:
    Generate an episode: $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
    For $t = 0, 1, \ldots, T-1$:
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\left[G_t - \hat{v}(S_t, \boldsymbol{w})\right]\nabla\hat{v}(S_t, \boldsymbol{w})$

Semi-gradient Methods

- The update step in GSD relies on the target being independent of $\boldsymbol{w}_t$, which is not valid if a bootstrapping estimate of $v_\pi(S_t)$ is used as U_t (e.g., n-step returns $G_t^{(n)}$ or DP target).
- Semi-gradient methods: take into account the effect of changing the weight vector $\boldsymbol{w}_t$ on the estimate, but ignore its effect on the target.

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

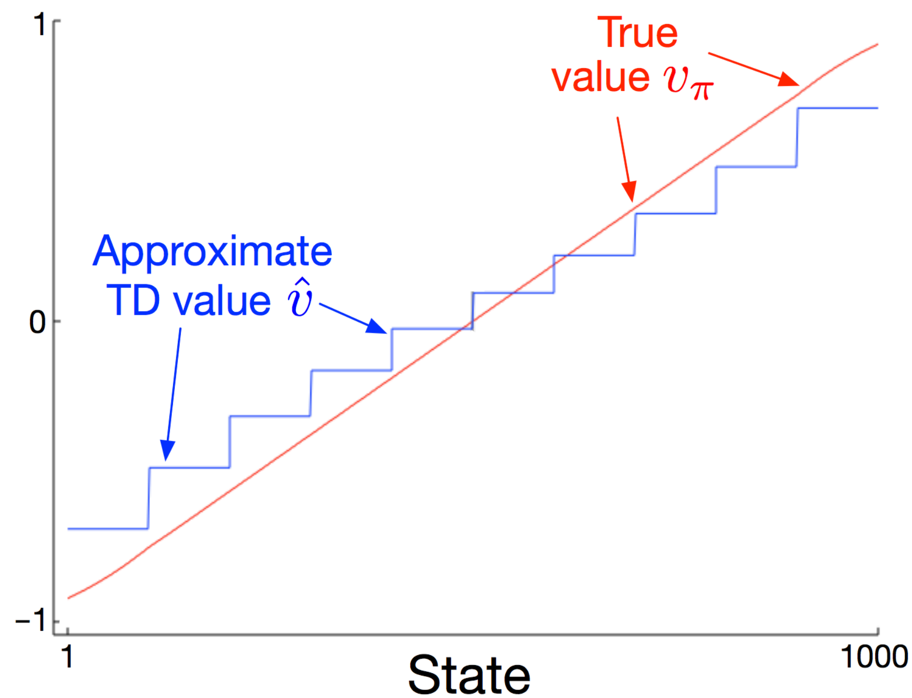
Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^n \mapsto \mathbb{R}$ such that $\hat{v}(\text{terminal}, \cdot) = 0$

Initialize value-function weights $\boldsymbol{w}$ as arbitrarily (e.g., $\boldsymbol{w} = \mathbf{0}$)
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose $A \sim \pi(\cdot|S)$. Take action A, observe R, S'
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \left[R + \gamma \hat{v}(S', \boldsymbol{w}) - \hat{v}(S, \boldsymbol{w})\right] \nabla \hat{v}(S, \boldsymbol{w})$
        $S \leftarrow S'$
    until S' is terminal

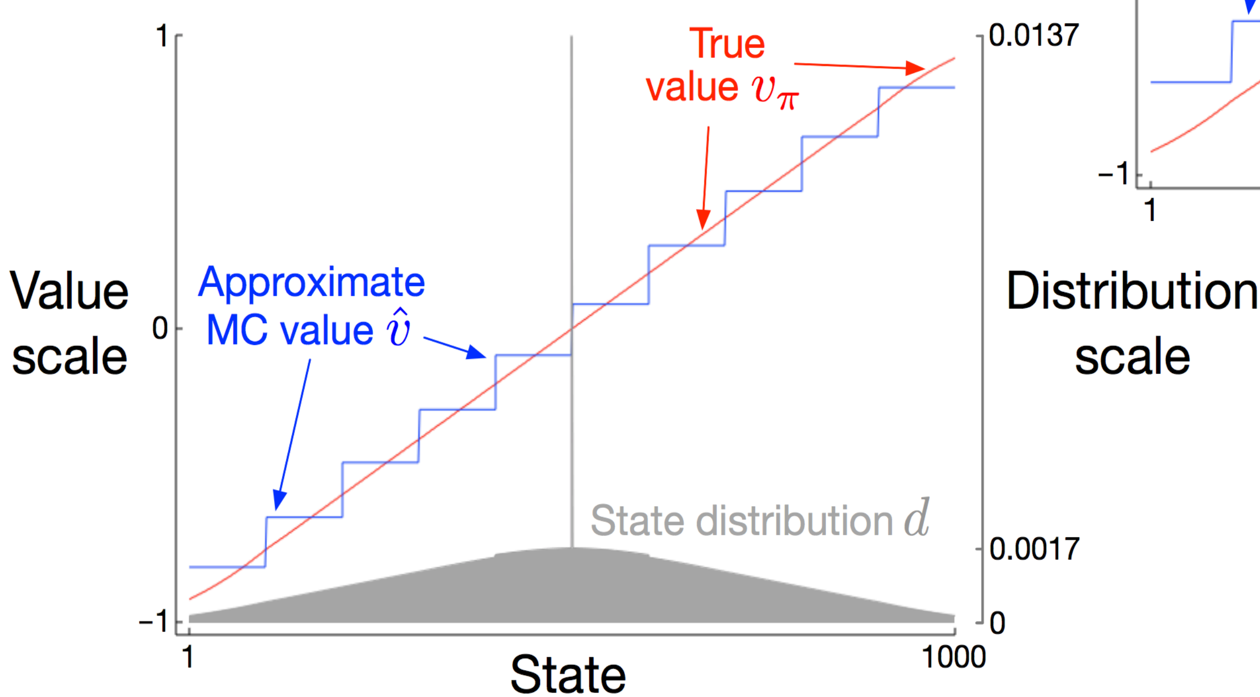
State Aggregation on 1000-state Random Walk

- 100 states are grouped together, with one component of the weight vector $\boldsymbol{w}$ for each group.

Semi-gradient TD(0)



Gradient MC algorithm



Linear Methods

- $\hat{v}(\cdot, \boldsymbol{w})$ is a linear function of $\boldsymbol{w}$: $\hat{v}(\cdot, \boldsymbol{w}) \doteq \boldsymbol{w}^\top \boldsymbol{x}(s) \doteq \sum_{i=1}^{n} w_i x_i(s)$
- $x_i : \mathcal{S} \to \mathbb{R}$: basis functions
  - Convergence guarantees
  - e.g., semi-gradient TD(0), TD fixedpoint
- Basis functions for d-dimensional state $s : (s_1, s_2, \ldots, s_d)^\top$
  - Can be used to add prior domain knowledge to RL systems.
  - Order-N Polynomial basis:

    $$x_i(s) = \prod_{j=1}^{d} s_j^{c_{i,j}}, \;\; c_{i,j} \in \{0, 1, \ldots, N\}$$

    - can take interaction of the state variables into account.
  - Fourier basis: $s_i \in [0, 1]$

    $$x_i(\boldsymbol{s}) = \cos\left(\pi \boldsymbol{c}^i \cdot \boldsymbol{s}\right), \text{ where } \boldsymbol{c}^i = \left(c_1^i, \ldots, c_d^i\right)^\top, c_j^i \in \{0, \ldots, N\}$$

    - suitable for RL problems with multi-dimensional continuous state spaces.

Semi-gradient TD (0)

- Semi-gradient TD(0)

$$\boldsymbol{w}_{t+1} \doteq \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \boldsymbol{w}_t^\top \boldsymbol{x}_{t+1} - \boldsymbol{w}_t^\top \boldsymbol{x}_t\right) \boldsymbol{x}_t$$
$$= \boldsymbol{w}_t + \alpha \left(R_{t+1}\boldsymbol{x}_t - \boldsymbol{x}_t \left(\boldsymbol{x}_t - \gamma \boldsymbol{x}_{t+1}\right)^\top \boldsymbol{w}_t\right)$$

- TD fixedpoint for steady state

$\mathbb{E}\left[\boldsymbol{w}_{t+1} | \boldsymbol{w}_t\right] = \boldsymbol{w}_t + \alpha\left(\boldsymbol{b} - \mathbf{A}\boldsymbol{w}_t\right)$, where

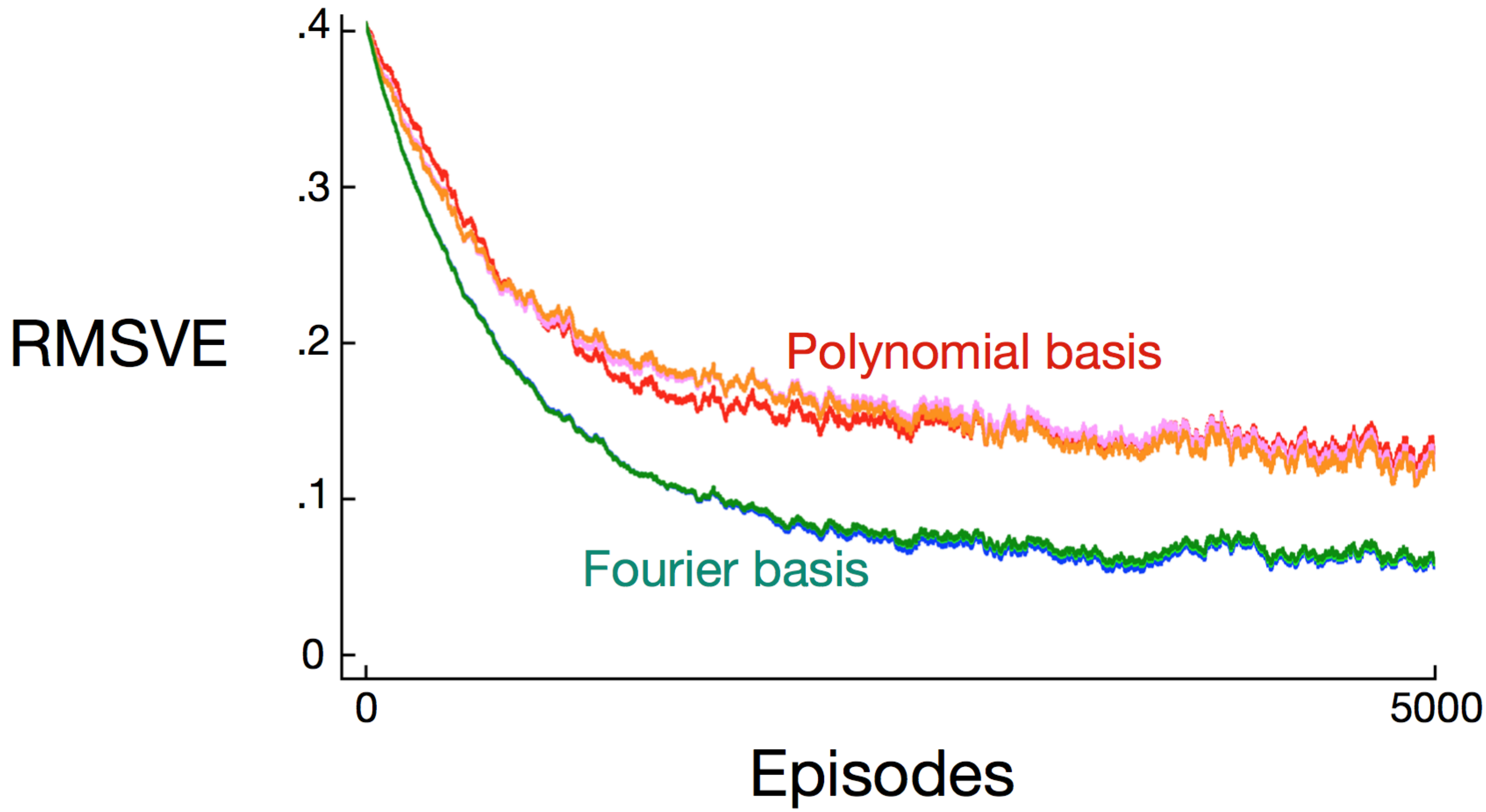
$$\boldsymbol{b} \doteq \mathbb{E}\left[R_{t+1}\boldsymbol{x}_t\right] \in \mathbb{R}^n, \;\; \mathbf{A} \doteq \mathbb{E}\left[\boldsymbol{x}_t \left(\boldsymbol{x}_t - \gamma \boldsymbol{x}_{t+1}\right)^\top\right] \in \mathbb{R}^{n \times n}$$

$$\Downarrow$$

$$\boldsymbol{w}_{TD} \doteq \mathbf{A}^{-1}\boldsymbol{b}$$

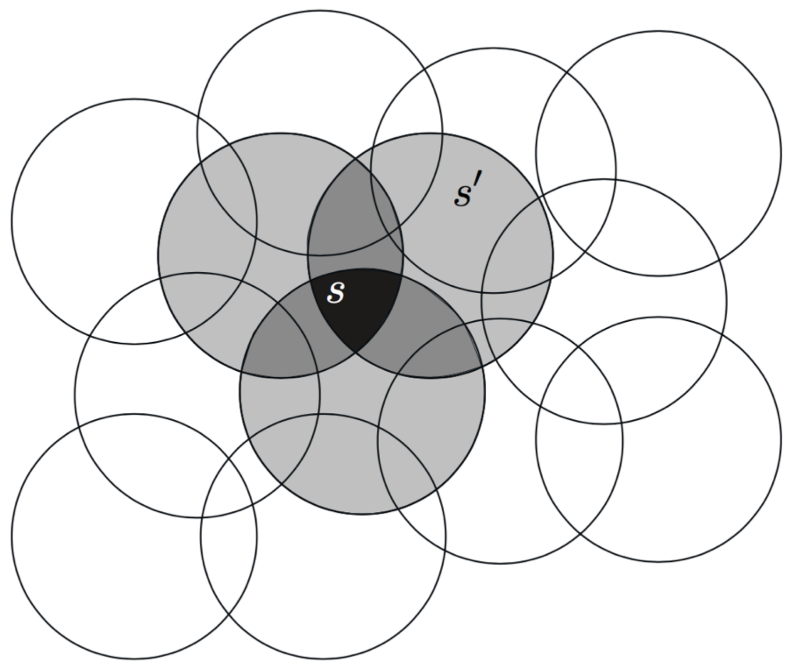
Fourier basis vs. polynomials on 1000-state Random Walk

- learning curves for the gradient MC method with Fourier and polynomial bases of degree 5, 10, and 20.

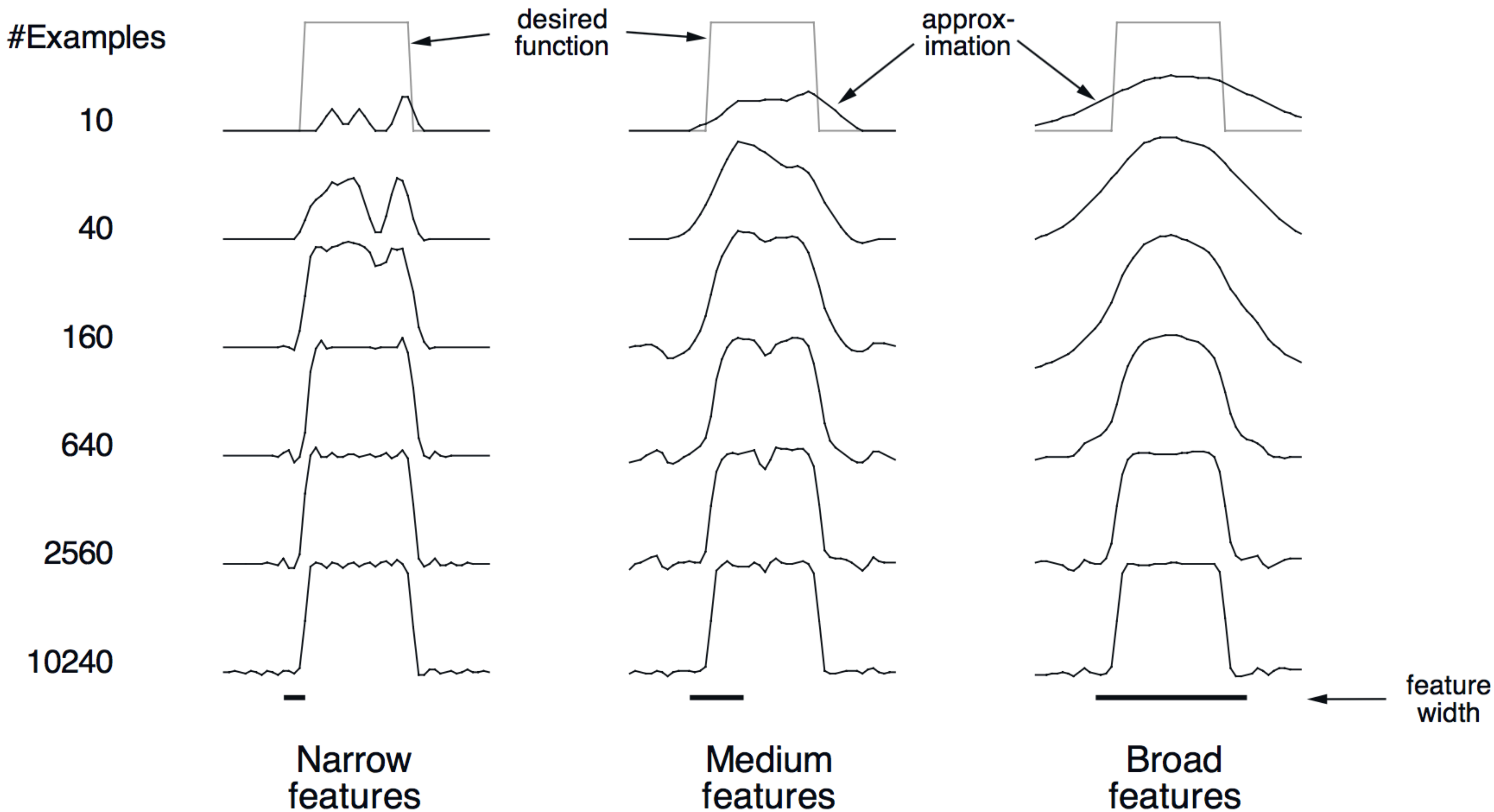


Coarse coding

- Binary feature: if the state is inside a circle, then the corresponding feature has the value 1 and is said to be *present*; otherwise the feature is 0 and is said to be *absent*.
- Coarse coding: representing a state with features whose receptive fields overlap (although they need not be circles or binary).
  - Large receptive field → broad generalization, but finest discrimination is controlled more by the total number of features.



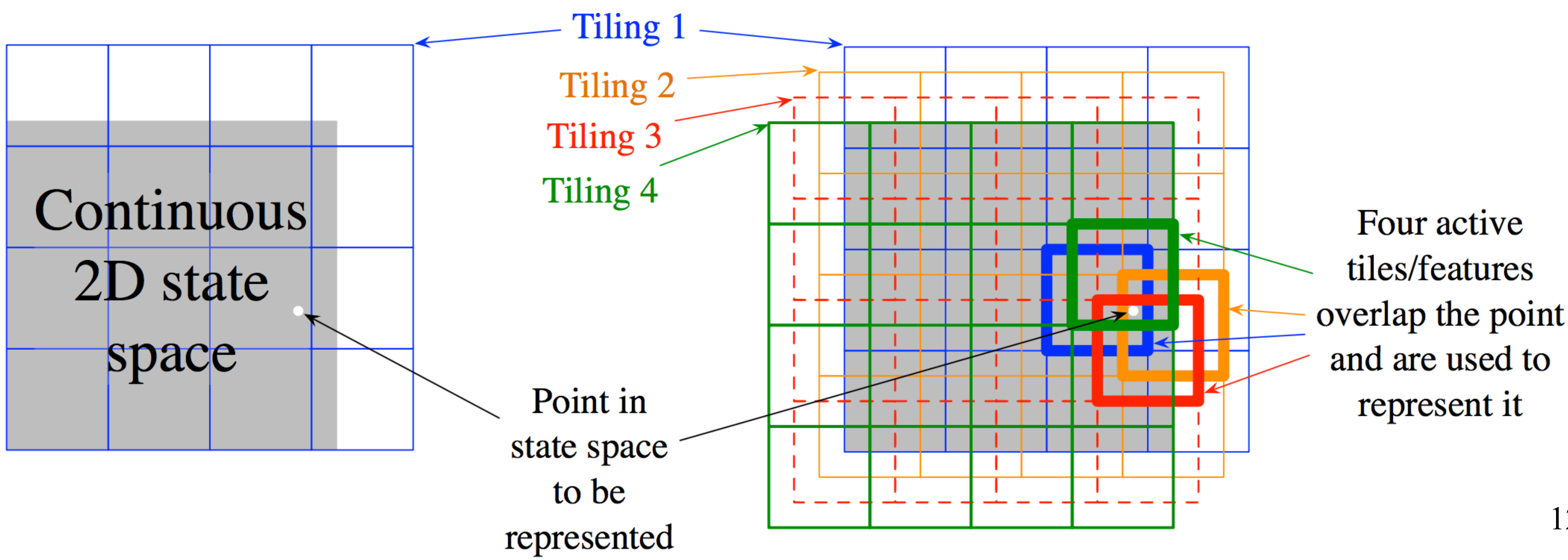
Coarseness of Coarse Coding



50 features in each case, state is time t in this interval

Tile coding

- A form of coarse coding for multi-dimensional continuous spaces.
- The receptive fields of the features are grouped into partitions (*tilings*) of the input space. Each element of the partition is called a *tile*.
  - Number of features present at any one time is constant.
  - Easy to compute over binary feature vectors.
  - Shape of tiles & offsets ⇒ generalization
  - Number of tilings ⇒ resolution of final approximation

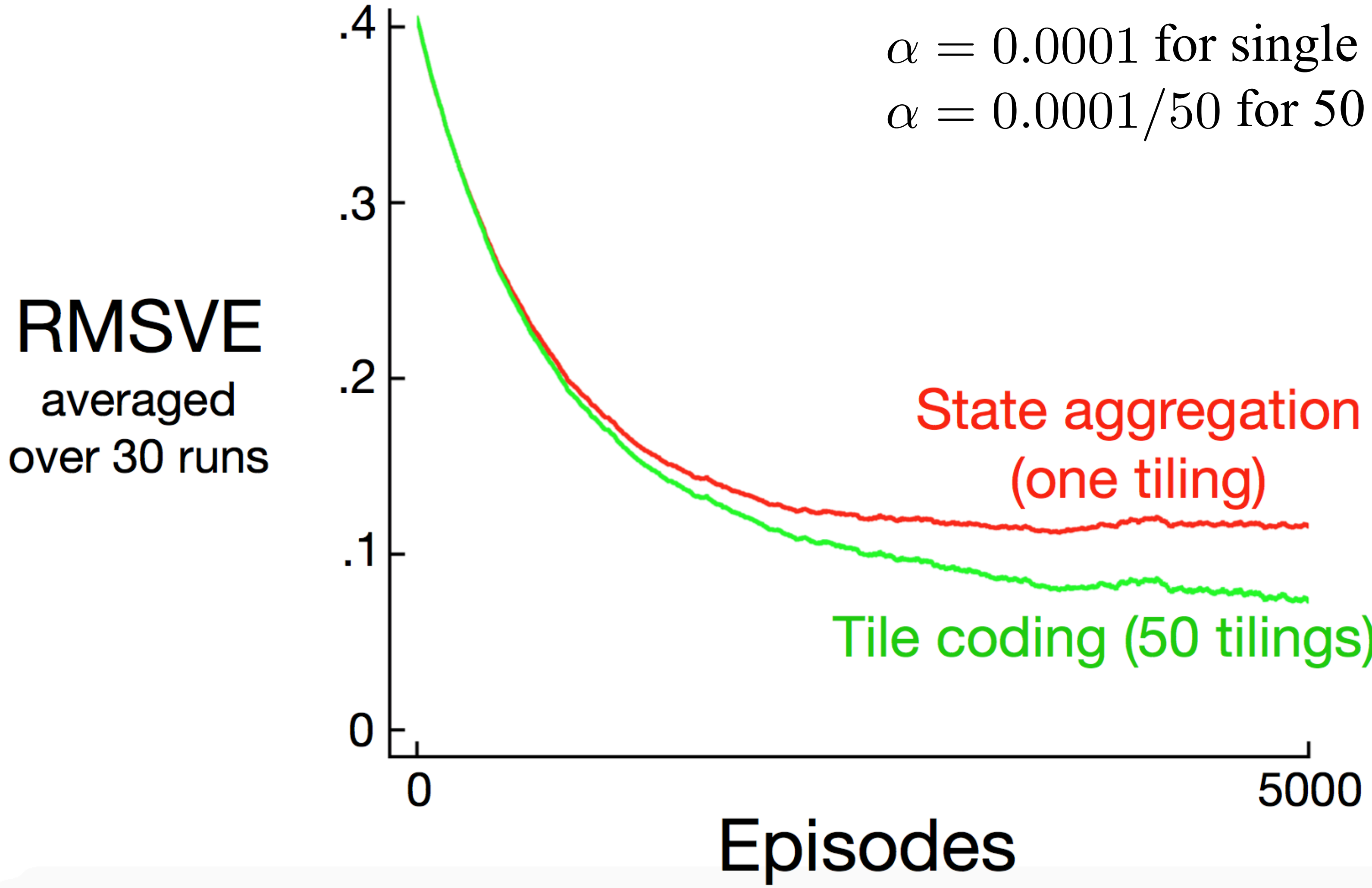


Tile Coding on 1000-state Random Walk

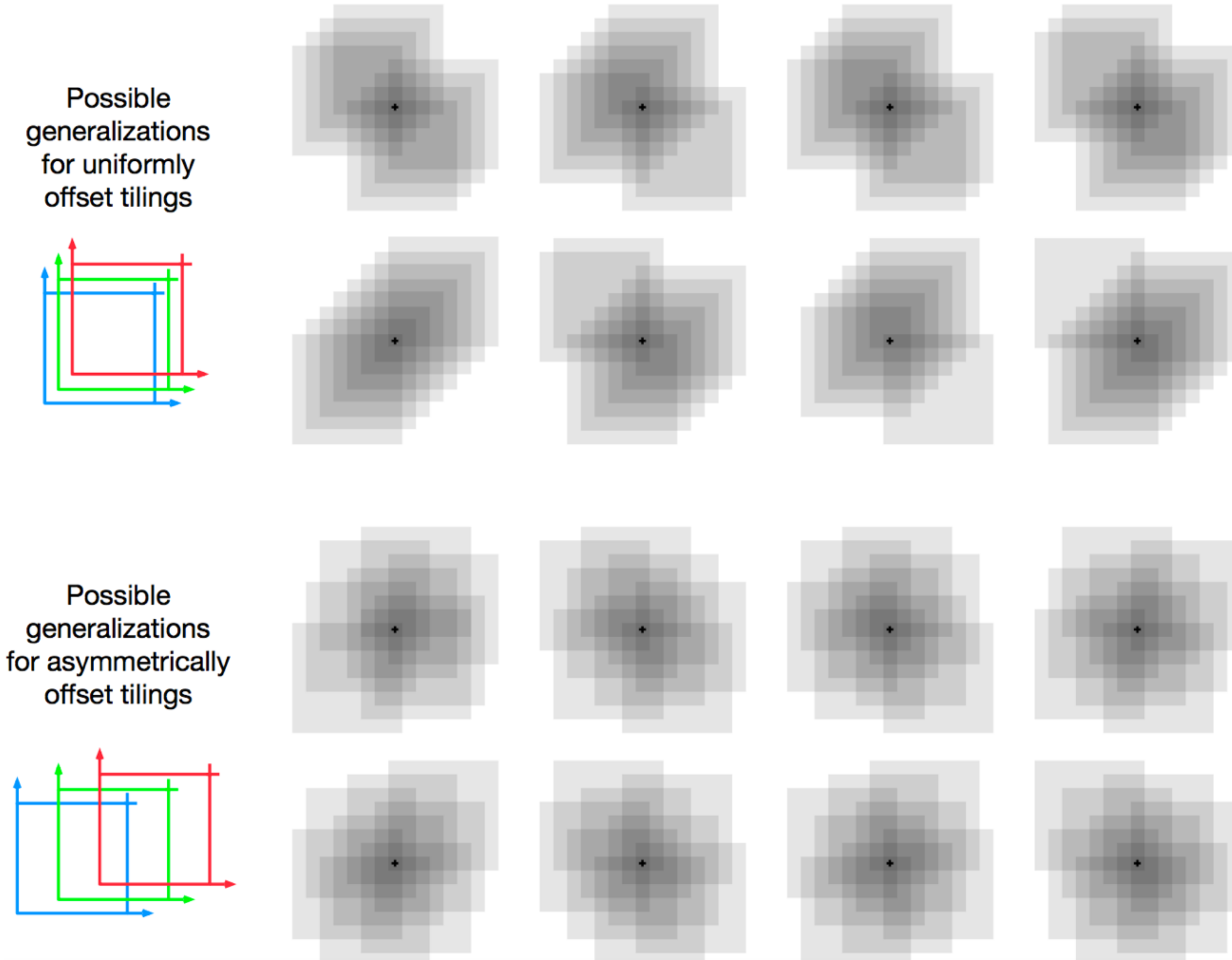
Each tile is 200 states wide

$\alpha = 0.0001$ for single tiling

$\alpha = 0.0001/50$ for 50 tilings

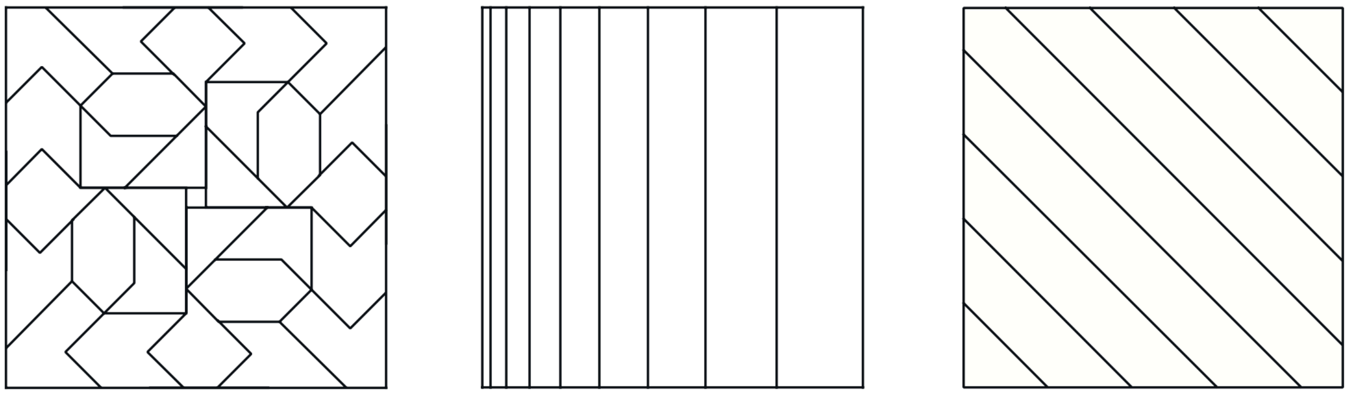


Offsets of Tilings Affect Generalization



Different Shaped Tiles

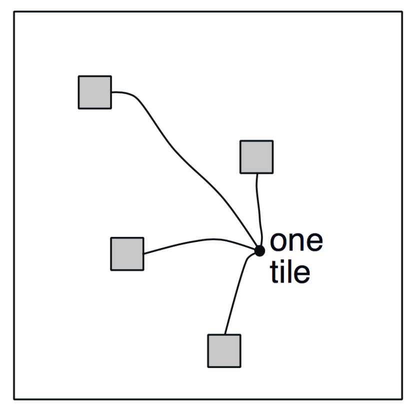
- Use different shaped tiles in different tilings ⇒ different generalization patterns, greater flexibility.



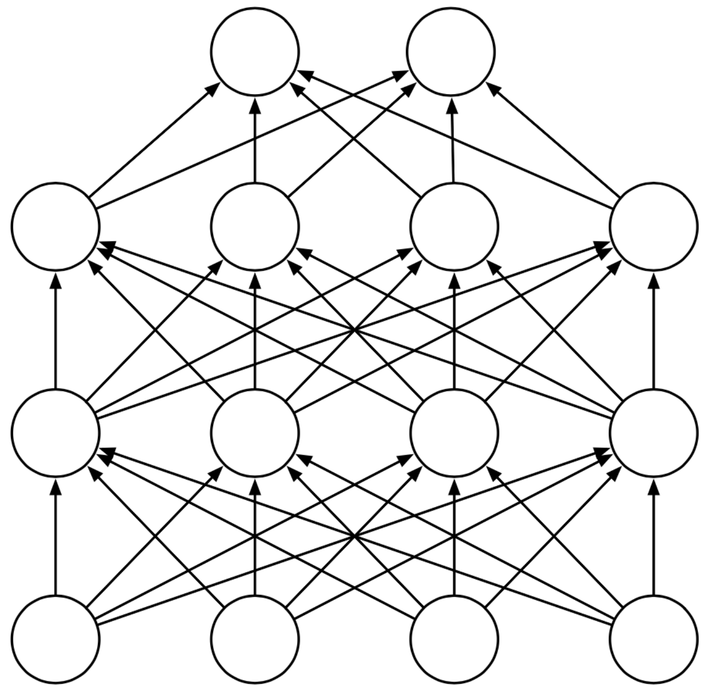
a) Irregular   b) Log stripes   c) Diagonal stripes

- Use hashing

  ⇒ reducing memory requirements



Nonlinear Function Approximation: Artificial Neural Networks



In RL, ANNs can use TD errors to learn value functions, or they can aim to maximize expected reward as in a gradient bandit or a policy-gradient algorithm (will be mentioned later).

Summary

- When state space is large, we need approximation
- Linear approximation has nice convergence properties
- Multiple choices of basis functions: key for performance
- Nonlinear approximation via ANN is promising