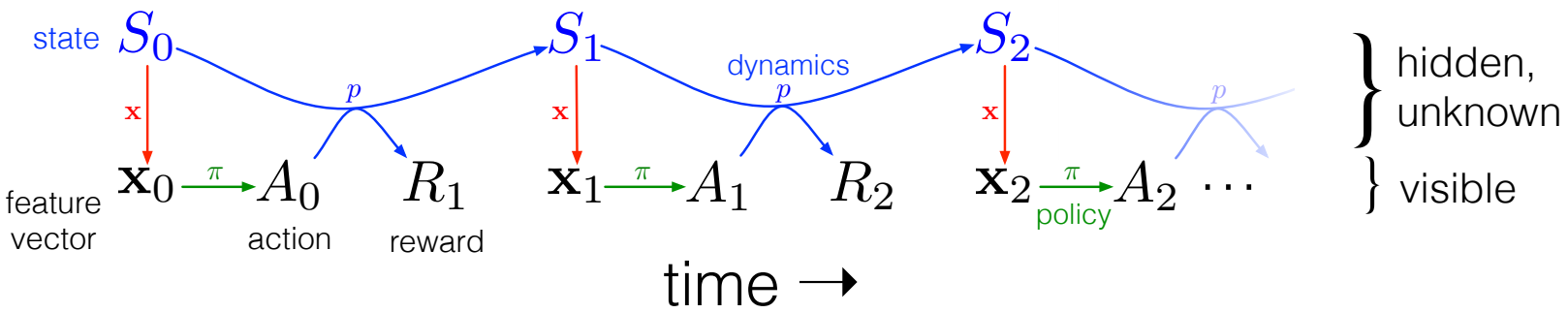
Introduction to Reinforcement Learning

Part 8:
RL with Approximations

Finite Markov decision processes (MDPs) with feature vectors



states $S_t \in \mathcal{S}$

actions $A_t \in \mathcal{A}$

rewards $R_t \in \mathcal{R}$

feature vectors $\mathbf{x}_t \in \Re^n \qquad \mathbf{x}_t = \mathbf{x}(S_t) \qquad \mathbf{x} : \mathcal{S} \to \Re^n$

policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1] \qquad \pi(a|s) = \mathcal{P}r\{A_t = a | S_t = s\}$

dynamics $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

$$p(s', r|s, a) = \mathcal{P}r\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\}$$

Value Prediction with FA

As usual: Policy Evaluation (the prediction problem):
for a given policy π, estimate the state-value function v_π

In earlier chapters, value functions were stored in lookup tables.

Here, the value function estimate at time t, V_t, depends on a *weight vector* $\mathbf{w}_t$, and only the weight vector is updated.

e.g., $\mathbf{w}_t$ could be the vector of connection weights of a neural network.

Linear Methods

Represent states as feature vectors:

for each $s \in \mathcal{S}$:

$$\hat{v}(s, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}(s) = \sum_{i=1}^{n} w_i x_i(s)$$

$$\nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w}) = \quad ?$$

Nice Properties of Linear FA Methods

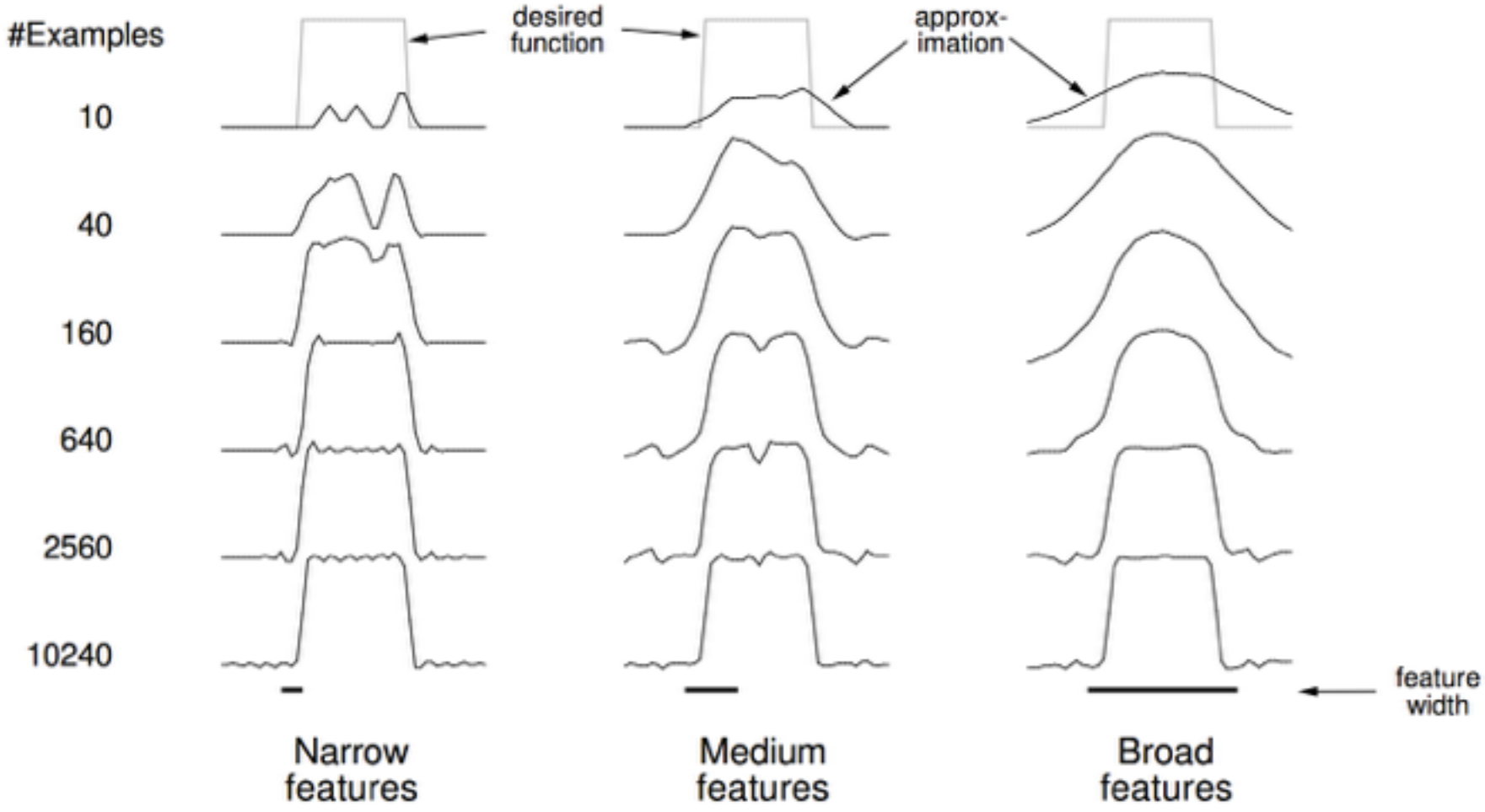
- The gradient is very simple: $\nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w}) = \mathbf{x}(s)$
- For MSE, the error surface is simple: quadratic surface with a single minimum.
- Linear gradient descent TD(λ) converges:
  - Step size decreases appropriately
  - On-line sampling (states sampled from the on-policy distribution)
  - Converges to weight vector $\mathbf{w}_\infty$ with property:

$$RMSE(\mathbf{w}_\infty) \quad \leq \quad \frac{1-\gamma\lambda}{1-\gamma} RMSE(\mathbf{w}^*)$$

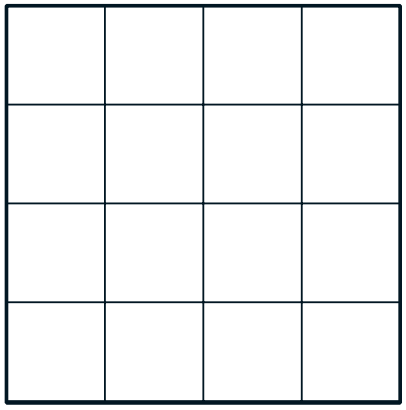
best weight vector

(Tsitsiklis & Van Roy, 1997)

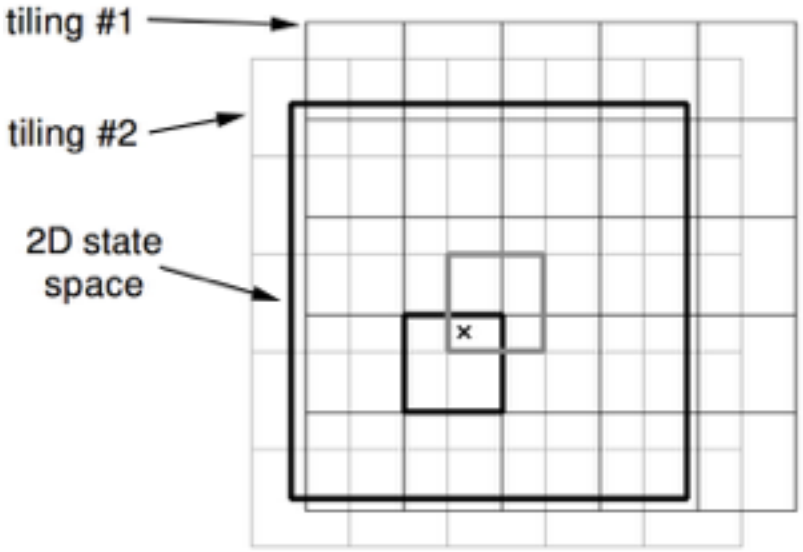
Learning and Coarse Coding



Tile Coding



- Binary feature for each tile
- Number of features present at any one time is constant
- Binary features means weighted sum easy to compute
- Easy to compute indices of the features present

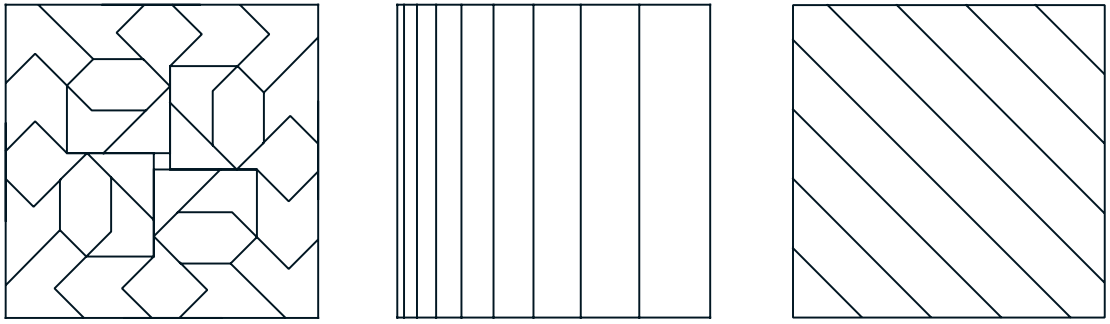


Shape of tiles ⇒ Generalization

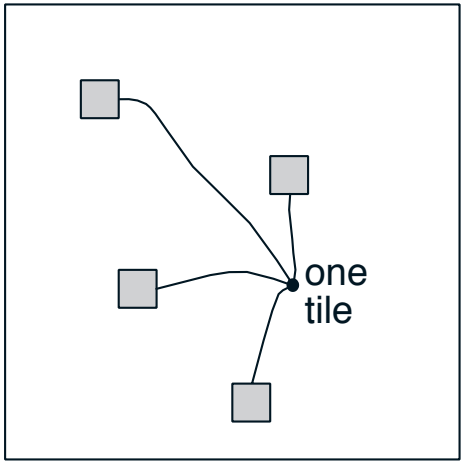
#Tilings ⇒ Resolution of final approximation

Tile Coding Cont.

Irregular tilings

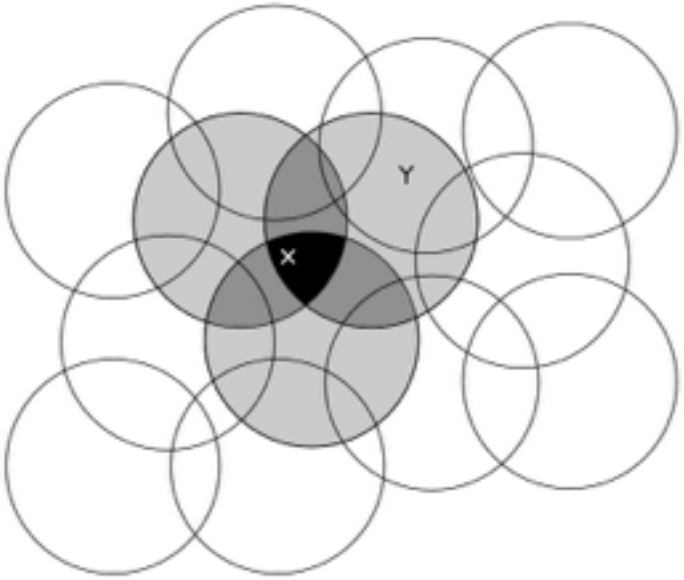


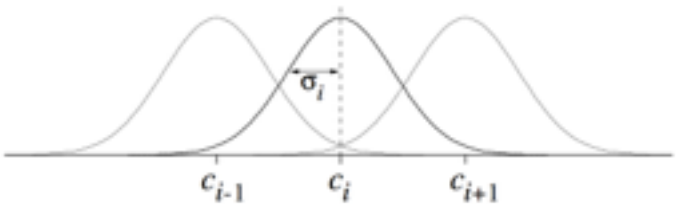
Hashing

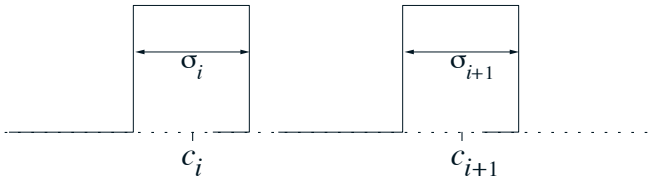


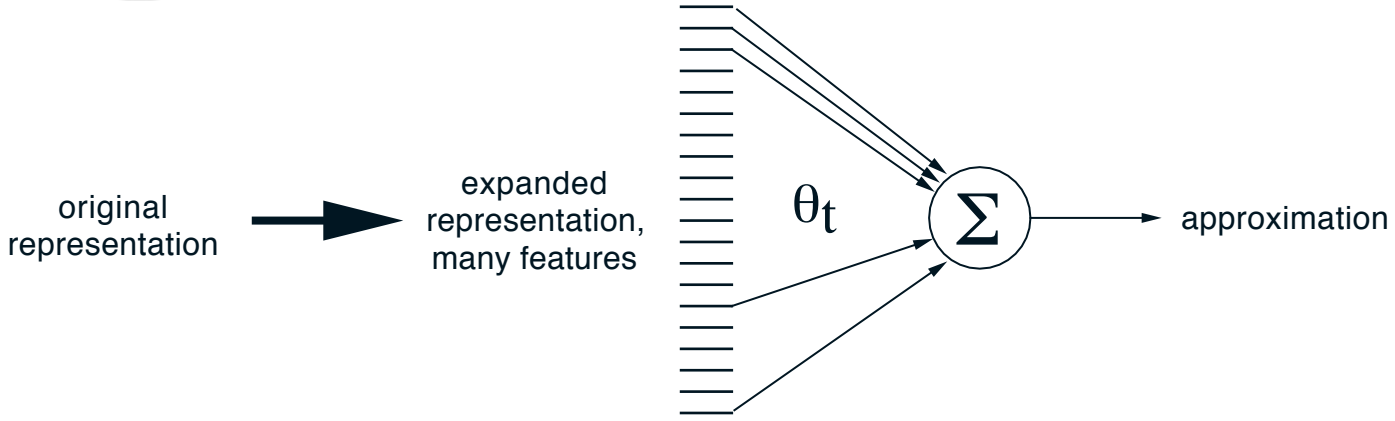
CMAC
"Cerebellar model arithmetic computer"
Albus 1971

Coarse Coding

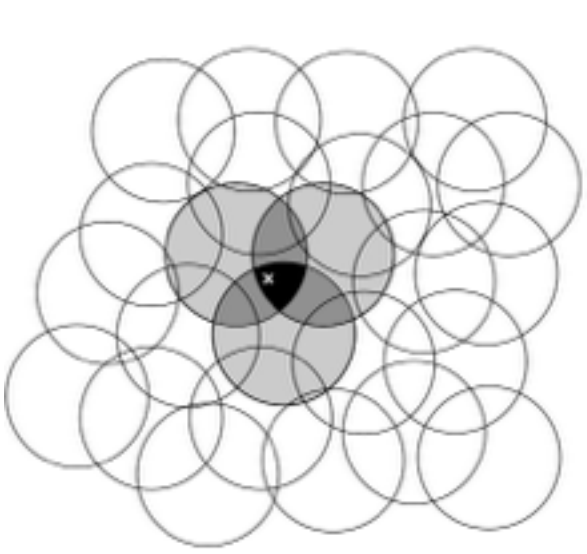




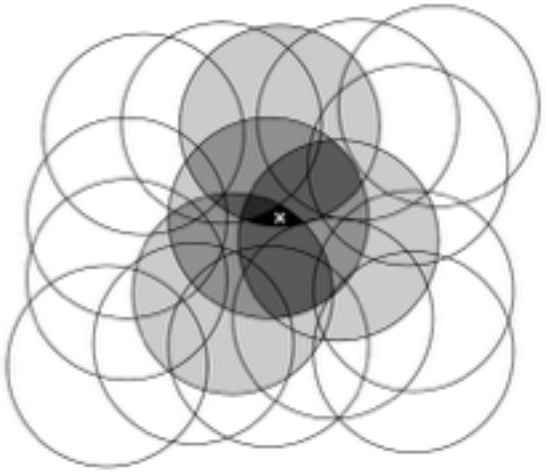




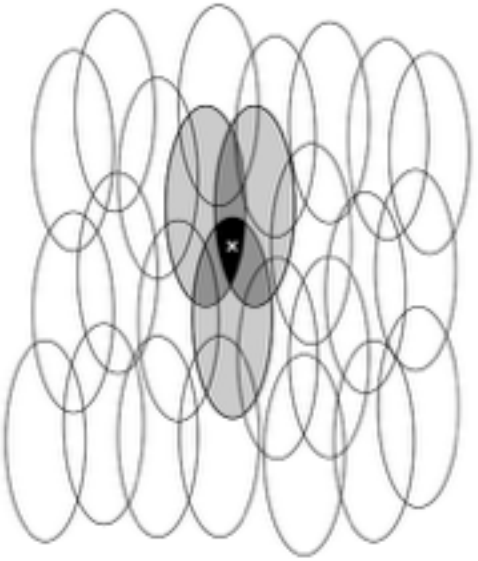
Shaping Generalization in Coarse Coding



a) Narrow generalization



b) Broad generalization



c) Asymmetric generalization

Gradient-based TD(λ), backwards view

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)$$

$$\mathbf{e}_t = \gamma \lambda \mathbf{e}_{t-1} + \nabla_{\mathbf{w}_t} \hat{v}(S_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \, \delta_t \, \mathbf{e}_t$$

Control with FA

- Learning state-action values

  Training examples of the form:

$$\Big\{\text{description of } (S_t, A_t),\ \ Q_t\Big\}$$

- The general gradient-descent rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\Big[Q_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla_{\mathbf{w}_t}\hat{q}(S_t, A_t, \mathbf{w}_t)$$

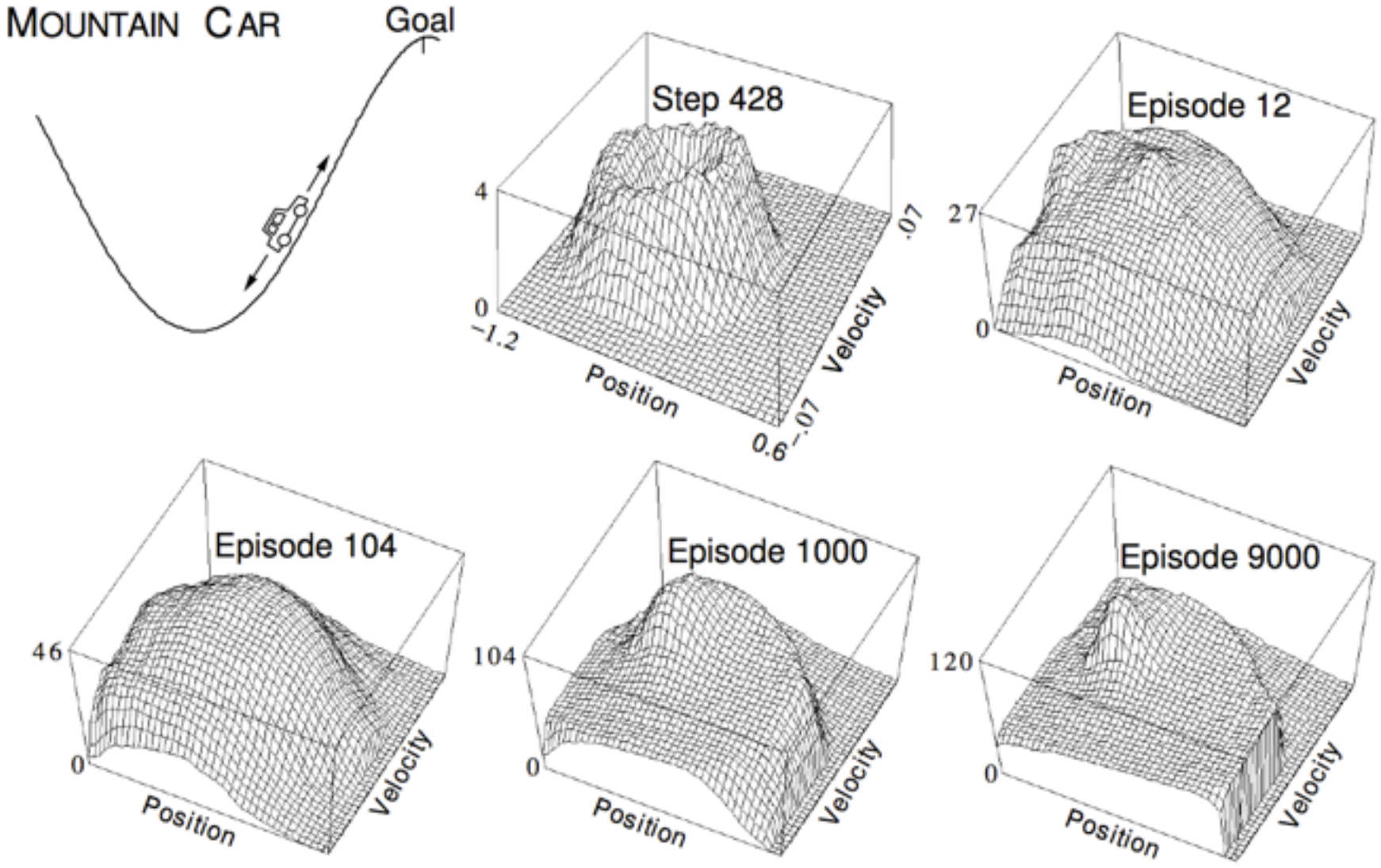
- Gradient-descent Sarsa(λ) (backward view):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\,\delta_t\,\mathbf{e}_t$$

where:
$$\delta_t = R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$$

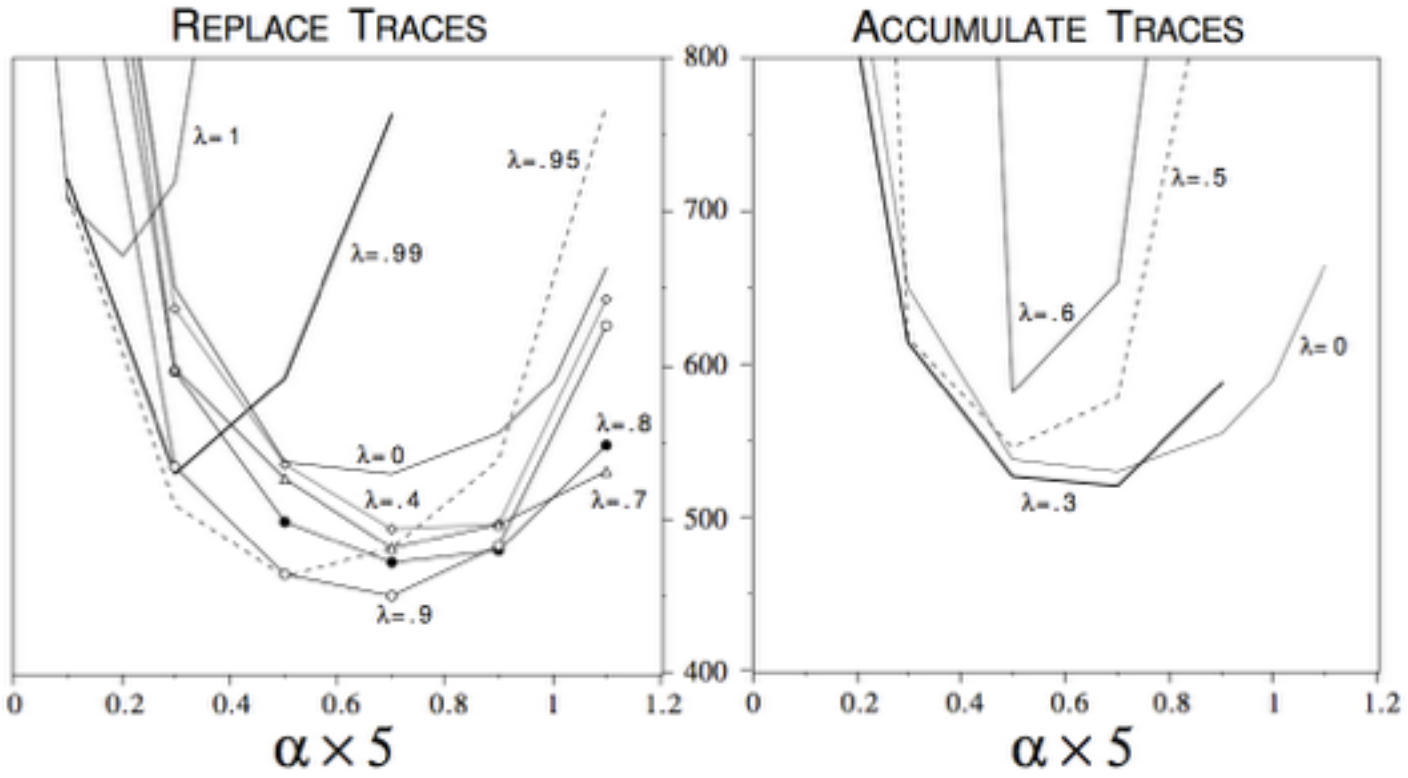
$$\mathbf{e}_t = \gamma\lambda\mathbf{e}_{t-1} + \nabla_{\mathbf{w}_t}\hat{q}(S_t, A_t, \mathbf{w}_t)$$

Approx Value Functions on Mountain-Car Task

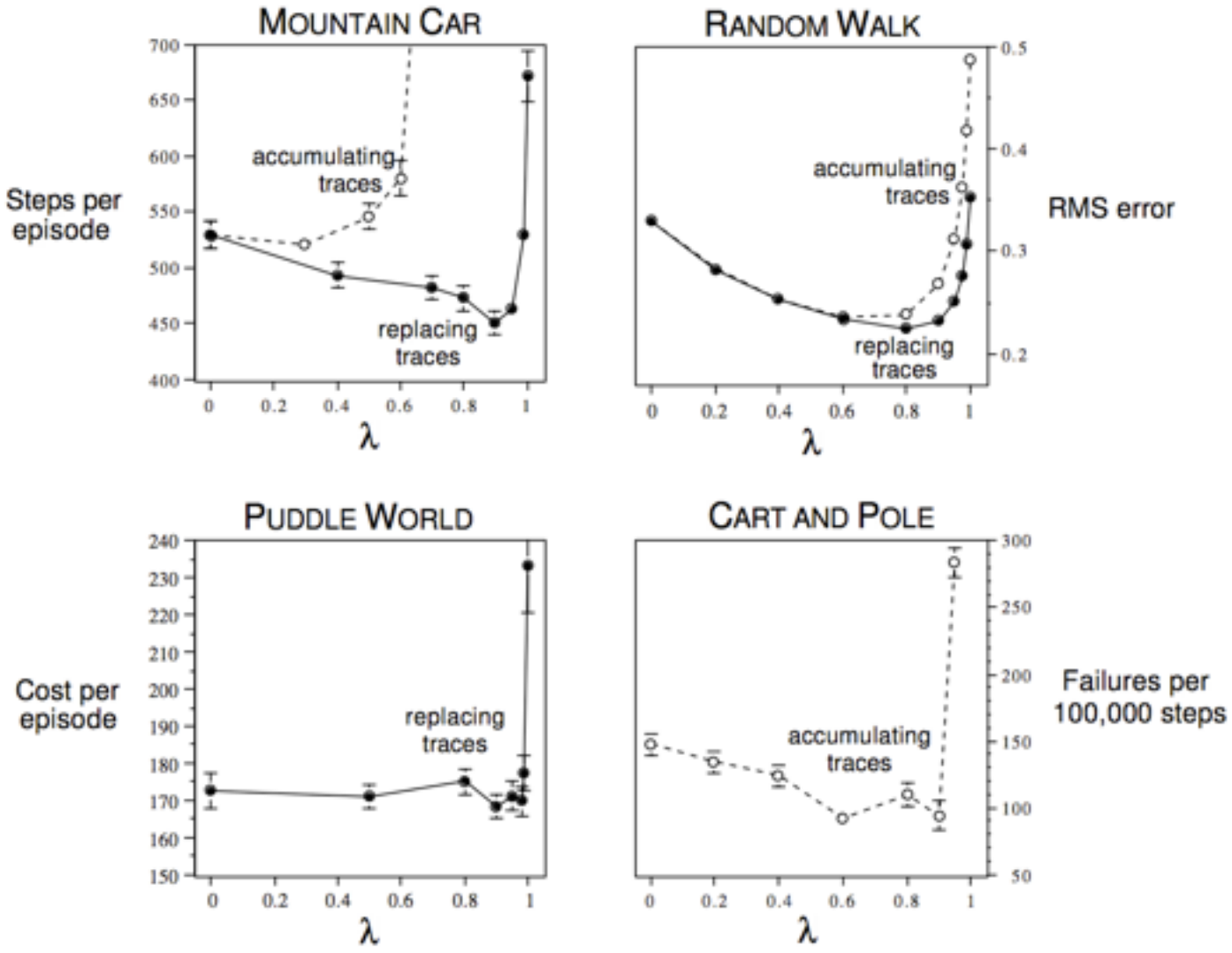


Mountain-Car Results

Steps per episode
averaged over
first 20 trials
and 30 runs



Should We Bootstrap?



Summary

- Generalization
- Adapting supervised-learning function approximation methods
- Linear gradient-descent methods
  - Tile coding