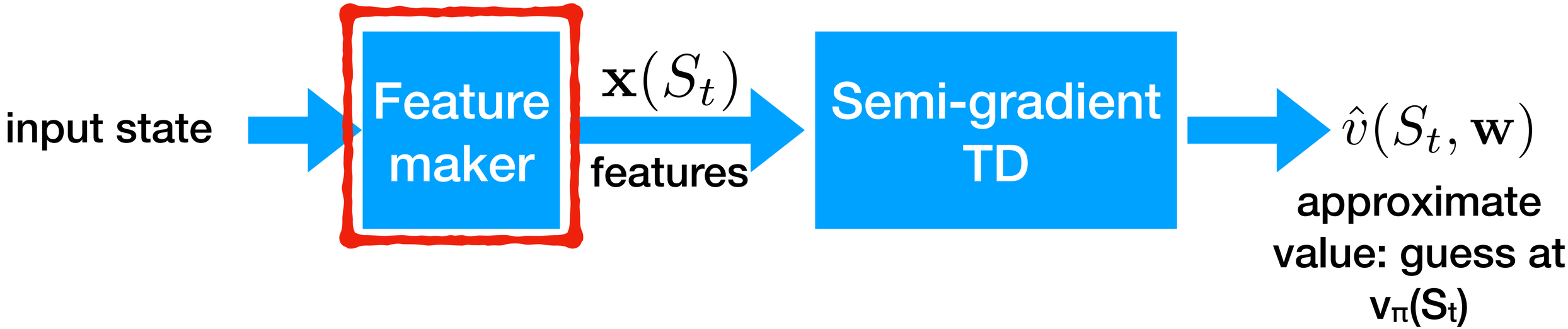
Course 3, Module 2
Constructing Features for Prediction

CMPUT 397
Fall 2020

Announcements

- Remember to keep up to date with your grades on eclass!
- Marking of the midterm will be done this week
  - grades will be released via eclass
  - an important policy: **no arguing for marks**
  - If you want to understand the answers, then come to Martha's office hours

Review of Course 3, Module 2
How to generate features, including Neural Nets


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

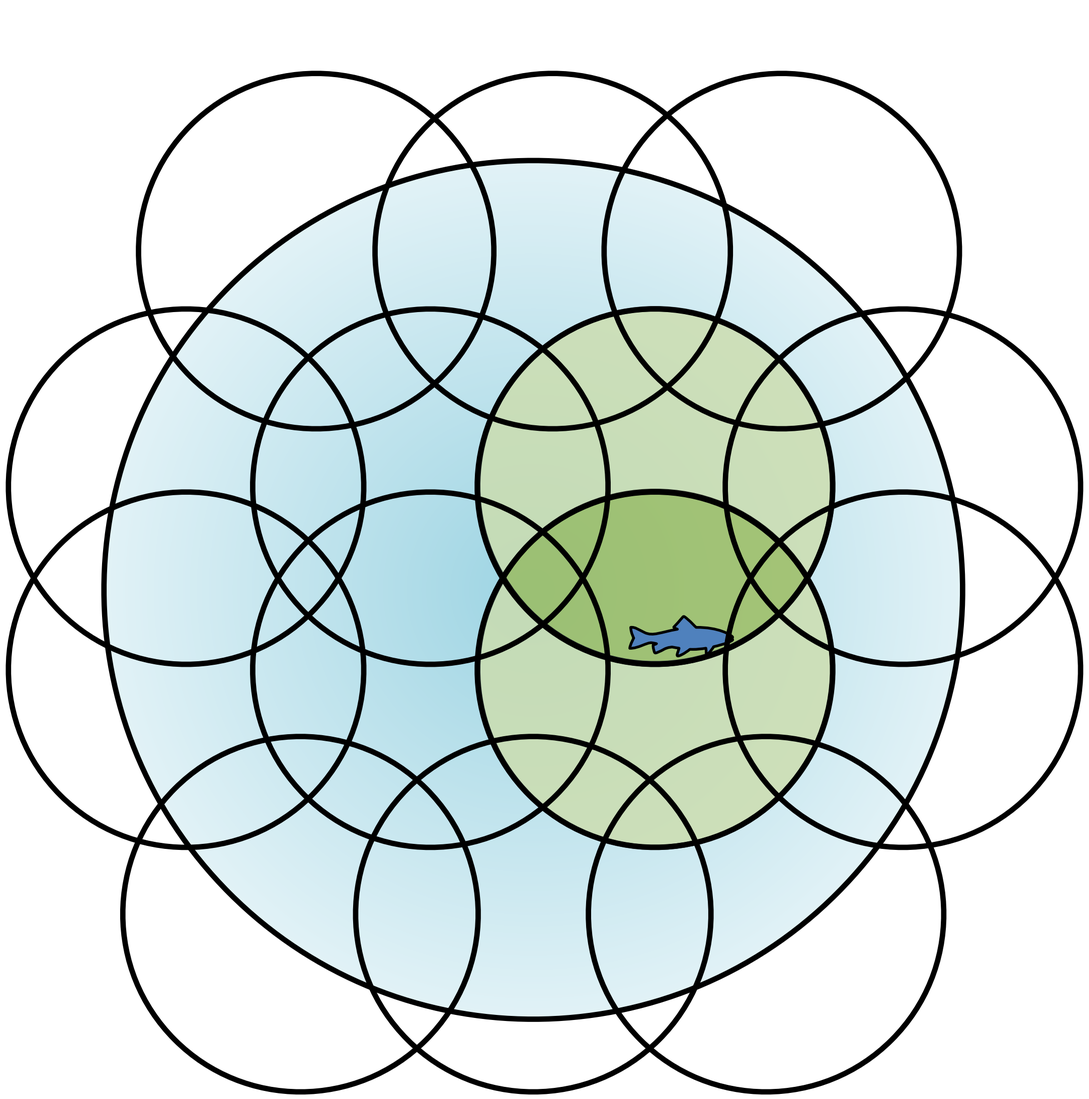
Video 1: Coarse Coding

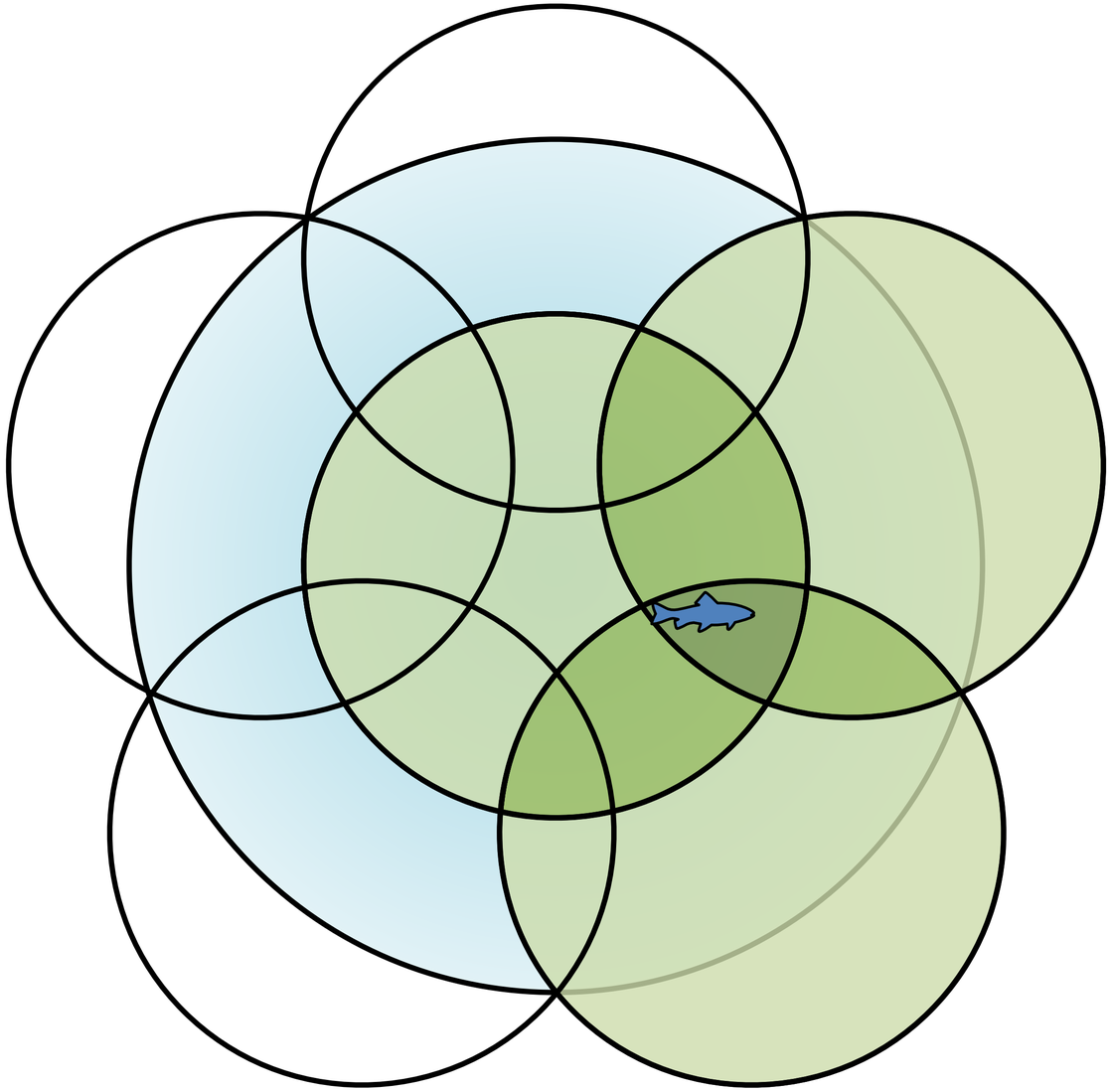
- **Linear function approximation**: how to process the input data to make a collection of features exploiting spacial locality
- Goals:
  - describe coarse coding
  - and describe how it relates to state aggregation.
    - we allow overlapping shapes
    - multiple "active" features

Video 2: Generalization Properties of Coarse Coding

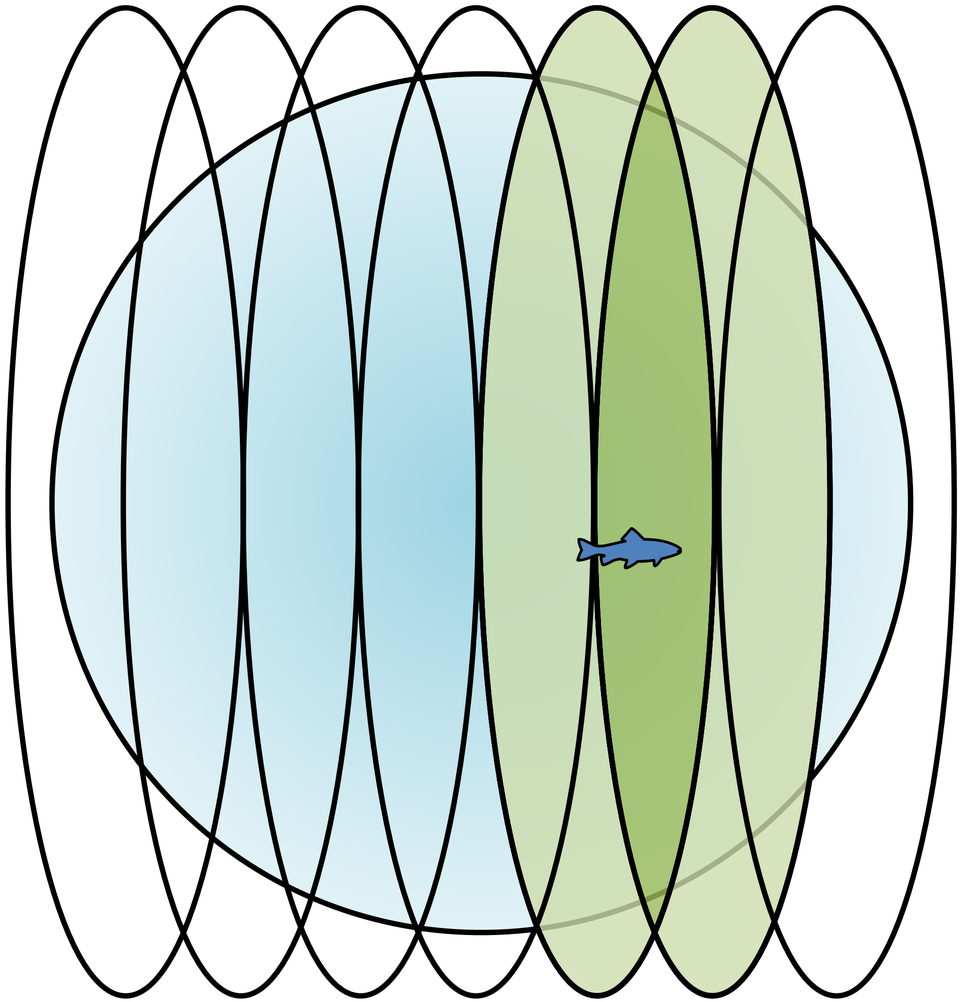
- **What do the shapes mean?**
- Goals:
  - Describe how coarse coding parameters affect generalization and discrimination
  - And understand how that affects learning speed and accuracy.

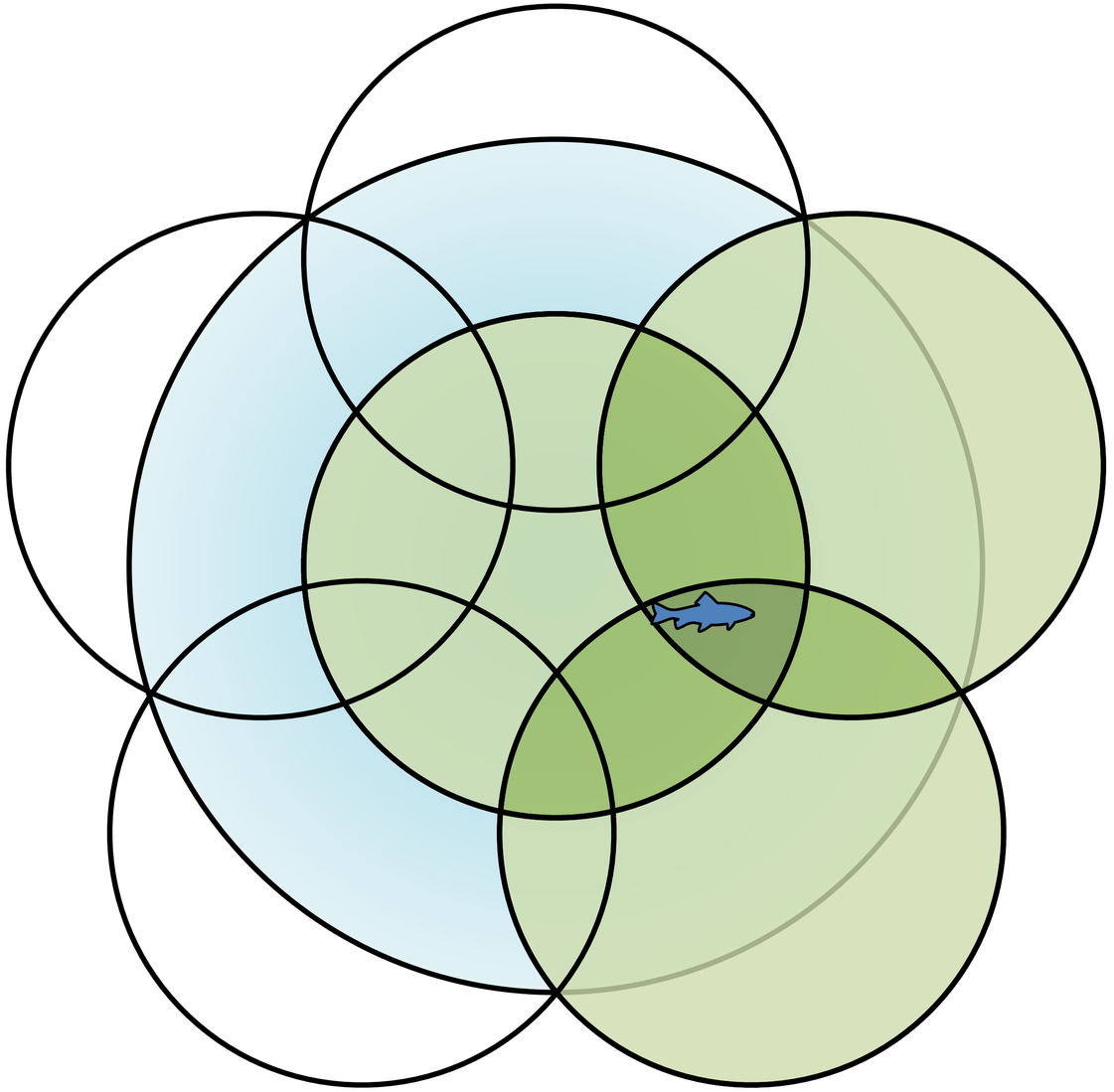
Broadness of generalization





Direction of generalization

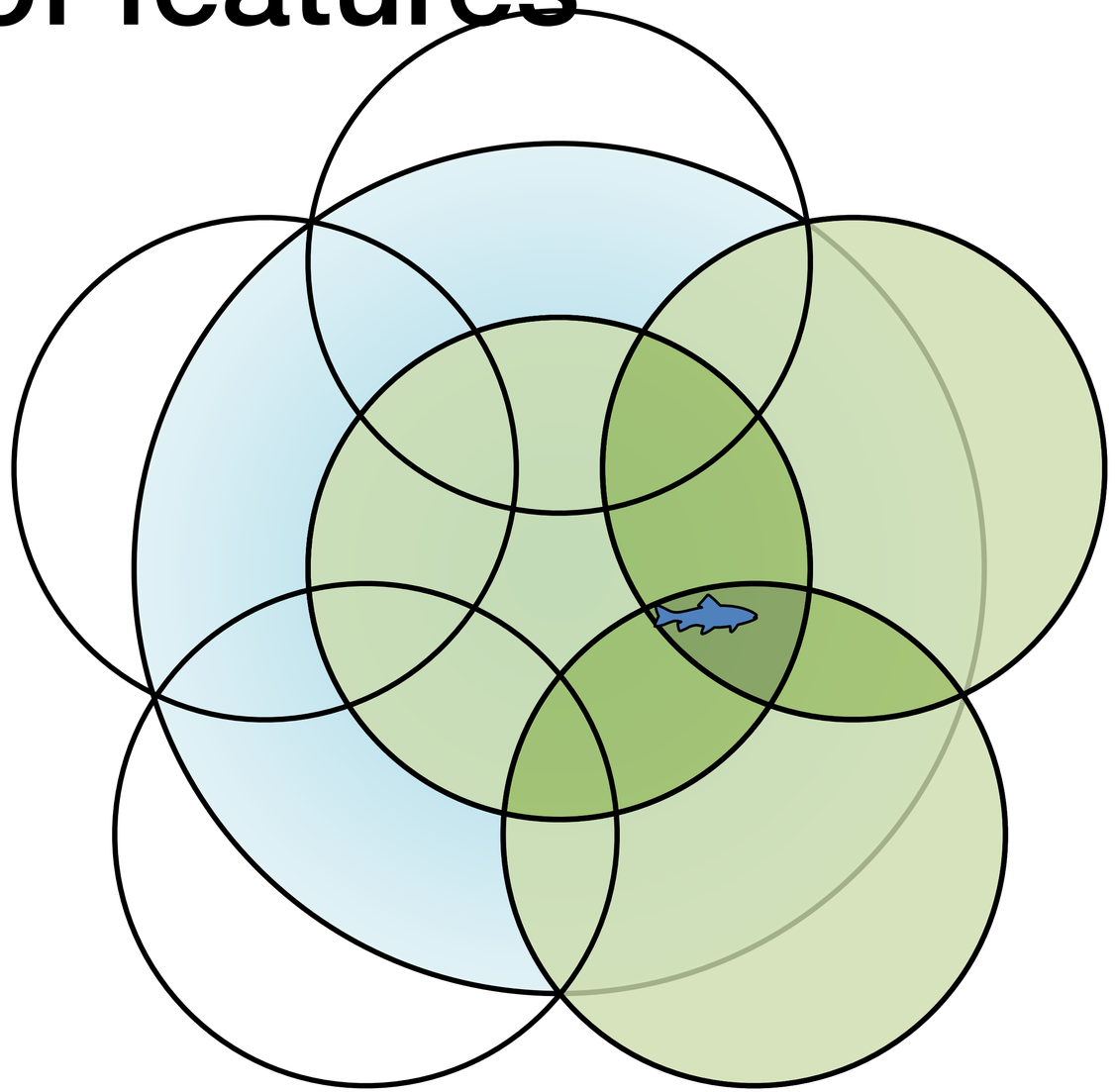




Question from slido about number of features

- “In the videos, we saw the example of how the size of a feature can affect generalization. Do the number and shape of a feature affect generalization in the same way?”

- If we increase the number of circles, then how might this impact generalization and discrimination?



Video 3: Tile Coding

- A computationally efficient coarse coding scheme, that is incredibly useful for situations where the input is low dimensional (<=4). An interesting specific case of coarse coding!

- Goals:

  - explain how tile coding achieves both generalization and discrimination

  - and understand the benefits and limitations of tile coding

Video 4: Using Tile Coding for Prediction

- Tile coding seems pretty great. Gradient Monte Carlo seems pretty good too. Let's put them together. A specific case of linear policy evaluation. We run an experiment on the thousand state chain, to see how tile coding compares to state aggregation

- Goals:

  - Explain how to use tile coding with temporal difference learning

  - And identify important properties of tile-coded representations.

Video 5: What is a Neural Network

- It is a non-linear function approximator, that learns the features. Note tile coding also provides non-linear value estimates in the state, but the value estimate is linear in the tile-coded features, making the gradient update for the weights simpler.

- Goals:

  - Understand feedforward neural networks

  - Define an activation function

  - And understand how a neural network is a parameterized function.

Video 6: Non-linear Approximation with Neural Networks

- In tile coding we had a function to compute the features for an input state. A neural network learns what the features should be based on data. For a different problem it could learn different features. We call this representation learning!

- Goals:

  - Understand how neural networks do feature construction

  - And you will understand how neural networks are a nonlinear function of state.

Video 7: Deep Neural Networks

- It's a neural network with lots of layers. It is a popular network structure that has been very successful in practice. Results in powerful features, but can be more challenging to train.

- Goals:

  - Understand how deep neural networks are composed of many layers.

  - And Understand that depth can facilitate learning features, through composition and abstraction.

Video 8: How to compute the gradient

- Backprop is a procedure for computing the gradient of the NN and updating the weights (avoiding redundant computation). Its Stochastic Gradient Descent for a Neural Network function approximator.

- Goals:

  - derive the gradient of a neural network

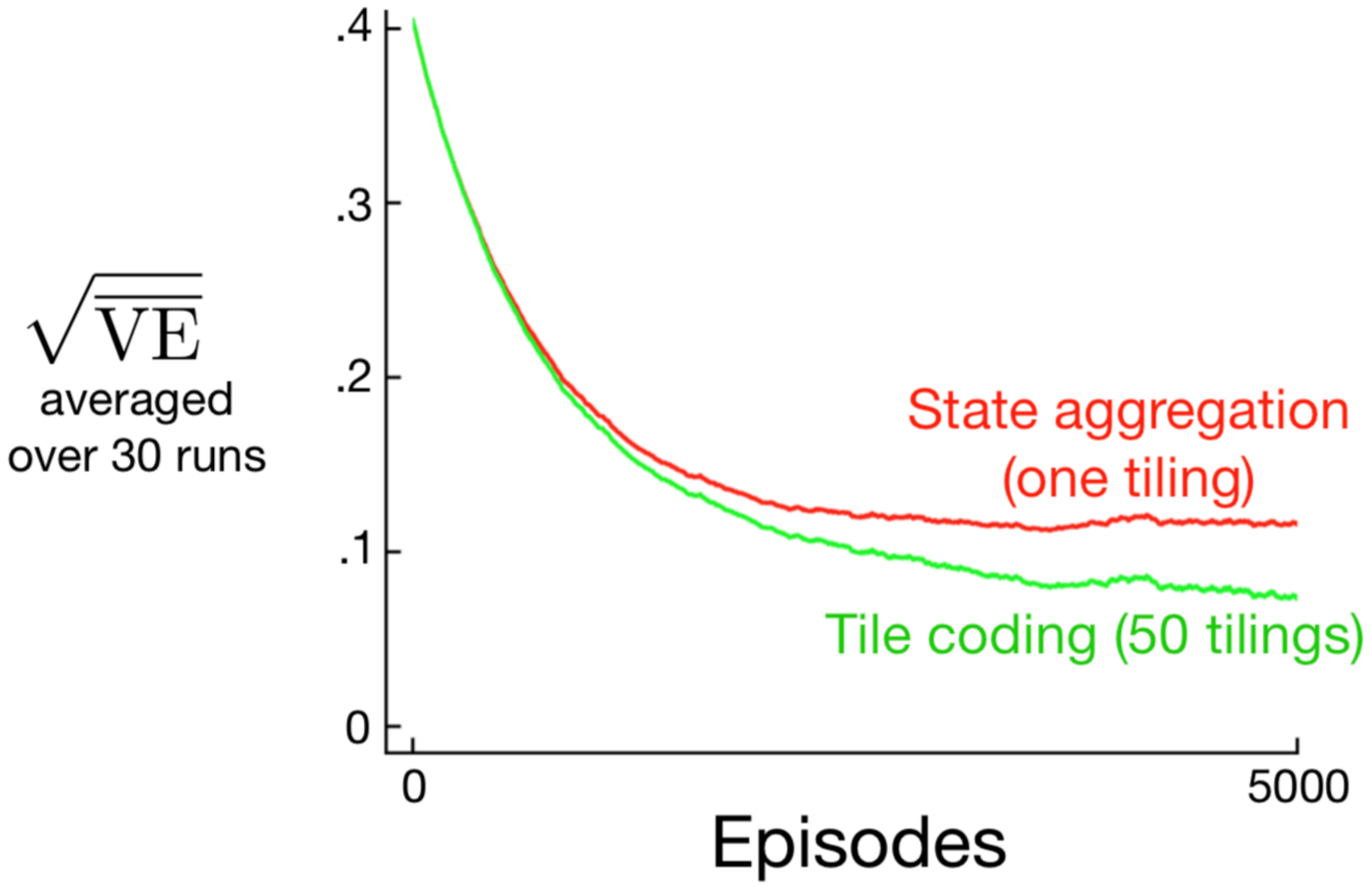
  - and Implement gradient descent on a neural network

Video 9: Optimization Strategies for NNs

- Doing vanilla Stochastic Gradient Descent on a Deep NN does not work very well. In fact, doing SGD on a one layer NN doesn't work very well either.
- Part of all the recent progress is due to massive increases in data and compute
- But there have also been important algorithmic developments
- Goals:
  - Understand the importance of initialization for neural networks.
  - And describe optimization techniques for training neural networks (momentum and vector of step-sizes: adapting the learning rate based on data!)

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

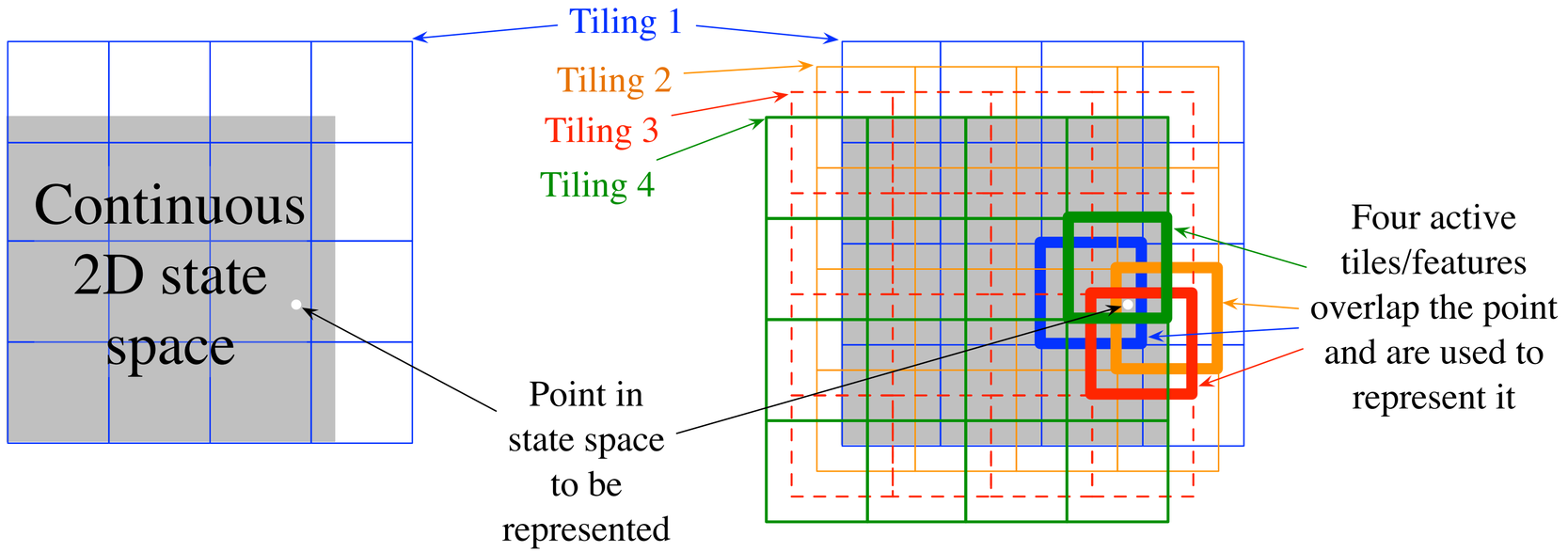
- State aggregation?
- Tile Coding?
- A NN?



Slido questions: Clarification

- “May I understand, the meaning of tile coding method is deal each tile layer at a time, to lower dimension ex, imagine we have n tile has dimension n, we can reduce it to be n separate one dimension problem, which result computing faster, by using neural network, each tile equal to each neuron layer.”

- “What is the input of the neural network for tile coding (the one that outputs tile coding representations)? Is that going to be continuous or discrete?”



Slido questions: Clarification

- Adjustable parameters are just the weights in the network
- “Why was it that in the ex where we used longer intervals in the 1D example (second vid) resulted in better discrimination? We were told that the smaller the region = better discrimination”
- “What is the advantage of using both Linear function and neural networks”

Slido: Clarifications in worksheet

- “When driving the gradient for the loss function of NNs, we split the equation up using the chain rule. Can you further explain how get compute each part of the chain rule expression when deriving the gradient for NNs?”

- “Can we go over question 10 in the quiz, that is finding the partial derivative of the output and a weighted matrix?”

Slide: Clarification using iPad

- “How exactly are activation functions applied? I see that when we get out matrix/ vector output from the neural net we then apply the activation function but I don't see how. Could we do an example of how to do this for the 3 types activation functions mentioned in the videos?”

- “What are the benefits of using 10 10x10 tiles versus 1 tile with dimensions 100x100?”

Slido questions: NN Specification

- “How do you decide what kind of activation function should be used for each neutron? Is there a specific way? can we use different activation function in a neural network or it is usually the case that activation function is unique in a neural network?”

- “The number of hidden layers (depth) and the widths of the layers have a large impact on the neural network's performance. Are there any proven/reliable strategies for selecting these numbers, or is it far too dependent on the training data to generalize?”

- “How should you decide on how many hidden layers and units to use in your Neural Network? Is it similar to deciding the discount factor and step size in previous algorithms?”

- “Other than CNNs, are RLNNs the most commonly used type of neural networks these days? How do they compare in speed and efficiency when compared to Supervised Neural Networks?”

Slido questions: Hyperparams

- “What's the most effcient way to tune hyperparameters nowadays?”

Misc/Advanced

- “Are there any guarantees about a neural network's performance compared to a linear function approximation? How does learning speed compare when considering the computational requirements for each, for example?”

- “In neutral networks, could we update the MDP's state and actions by combining neurons with some other learning methods besides Q-learning? Are there in similar process?”

- “Does increasing the depth of a neural network more computationally expensive? How important is the universal approximation property to us?”

- “We know that increasing the depth of NNs will have benefits like abstraction. Are there any advantages of increasing the width of each layer?”