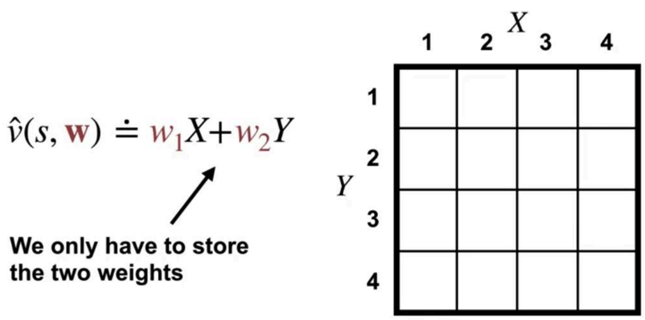
Parameterized Functions

Instead of storing values in a table for each state, we can use parameterized functions to approximate value functions. This is especially useful for large or continuous state spaces.

1. Parameterized Functions in RL



Key Formula:
v̂(s,
w) = w_1x + w_2y
Where:
- v̂ is the approximate value function
- w represents weights
- x,y are state features

2. Linear Value Function Approximation

A fundamental approach to value function approximation using linear combinations of features:

$$\begin{aligned}\hat{v}(s, \mathbf{w}) &= \sum_i w_i x_i(s) \\ &= \langle \mathbf{w}, \mathbf{x}(s) \rangle\end{aligned}$$

- Where:
  - $x(s)$ is the feature vector
  - w is the weight vector
  - $x_i(s)$ represents individual features
  - $\hat{v}(s,\mathbf{w})$ is the inner product of $\mathbf{w}$ and $\mathbf{x}(s)$

Limitations:

- Assumes linear relationships between features and values
- May not capture non-linear complex state-value relationships
- Heavily dependent on feature selection
- Can struggle with high-dimensional state spaces
- May be sensitive to noisy or irrelevant features

3. Generalization vs Discrimination

Generalization

- Ability to perform well on new, unseen data
- Essential for practical RL applications
- Measured through accuracy on test data
- Desirable property

Discrimination

- Ability to distinguish between different states
- Important for precise value estimation
- Balance needed with generalization
- Harmful and should be avoided?

4. Supervised Learning in RL

Supervised learning methods can be integrated into RL in several ways:

Applications in RL:
1. Value function approximation
2. Environment modeling
3. Policy learning from demonstrations
4. Function approximation for Q-learning
5. Transfer learning
6. Data Augmentation

Key Differences from Pure RL:

Aspect	Supervised Learning	Reinforcement Learning
Data Source	Labeled training data	Environment interaction
Feedback	Immediate and direct	Delayed and sparse
Learning Target	Known correct outputs	Estimated values/rewards

5. Practical Implementation Tips

- Start with simple linear approximators before moving to complex ones
- Carefully select and engineer features
- Monitor for overfitting using validation data
- Consider computational efficiency in large state spaces
- Use appropriate learning rates for stability

Common Pitfalls to Avoid:
- Over-complicated feature representations
- Ignoring the curse of dimensionality
- Poor feature scaling
- Inadequate exploration during learning