Deep Q Learning: From Paper to Code

Naive Deep Q Learning in Code

Last Time ...

- Deep neural nets for large / continuous spaces
- Universal function approximators
- Inputs compared to outputs to minimize cost
- Basic example

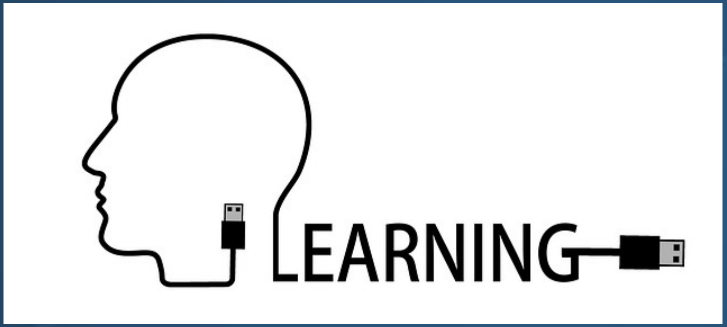
Implementation details

- Split code into 2 classes → Network class is first
- 2 linear layers, hidden layer is 128 x n_actions
- Use Adam optimizer with 0.001 learning rate
- Mean squared error loss → nn.MSELoss()
- Don’t forget device selection
- Relu activation function
- Rest of functionality is in the agent class
  - Init keeps track of gamma, epsilon and action space
  - Epsilon greedy action selection (hint: use tensor.item())
  - Decrement epsilon
  - Don’t forget to zero grad

$$Q(s,a)=Q(s,a)+\alpha(r+\gamma \max_{a} Q(s',a_{max})-Q(s,a))$$

- 10,000 games, gamma 0.99, epsilon goes to 0.01 over 2500 games

(Many) Problems With Our Approach



Learning from one example

Enormous parameter space

Large epsilon enables exploration

(Many) Problems With Our Approach

One network for two tasks

$$Q(s,a)=Q(s,a)+\alpha(r+\gamma \boxed{\max_a Q(s',a_{max}))}-Q(s,a))$$

Could it learn with more games?

Up Next