Page 1, Values and policy

Formally it could be written as $\pi(s) = \arg\max_a Q(s, a)$, which means that the result of our policy π at every state s is the action with the largest Q.

In math notation, policy is usually represented as $\pi(s)$, well use this notation as well.

Page 2, Policy gradients

We define policy gradient as $\nabla J \approx \mathbb{E}[Q(s, a)\nabla \log \pi(a|s)]$. Of course, there is a strong proof of this, but its not that important. Which is much more important is the semantic of this expression.

From the practical point of view, policy gradient methods could be implemented as performing optimisation of this loss function: $\mathcal{L} = -Q(s, a) \log \pi(a|s)$.

Page 3, REINFORCE

1. Initialize the network with random weight.
2. Play N full episodes, saving their (s, a, r, s) transitions.
3. For every step t of every episode k calculate discounted total reward for subsequent steps $Q_{k,t} = \sum_{i=0} \gamma^i r_i$
4. Calculate loss function for all transitions $\mathcal{L} = -\sum_{k,t} Q_{k,t} \log(\pi(s_{k,t}, a_{k,t}))$
5. Perform SGD update of weights minimizing the loss.
6. Repeat from step 2 until converged.

Page 9, Full episodes are required

When we talked about DQN weve seen that in practice its fine to replace the exact value for discounted reward with our estimation using 1-step Bellman equation $Q(s, a) = r_a + \gamma V(s')$.

Page 9, High gradient variance

In policy gradients formula $\nabla J \approx \mathbb{E}[Q(s, a)\nabla \log \pi(a|s)]$ we have gradient proportional to the discounter reward from the given state.

Page 10, Exploration

In math notation, entropy of the policy is defined as: $H(\pi) = -\sum \pi(a|s) \log \pi(a|s)$