۱) MDP

S	(۱,۱)	(۱,۲)	(۱,۳)	(۲,۱)	(۲,۲)	(۲,۳)
V_0	0	0	-۵	0	0	۵
V_1	0	0	-۵	0	۷,۶	۵
V_2	0	۴,۵۵۲	-۵	۵,۴۷۲	۸,۲۸۴	۵

۲)

S	(۱,۱)	(۱,۲)	(۱,۳)	(۲,۱)	(۲,۲)	(۲,۳)
$\pi^*(s)$	↑	↑	-	→	→	-

۳) Monte Carlo یک روش Model-free می‌باشد که به میانگین گیری نتایج قسمت‌ها برای تخمین توابع action-value تکیه دارد. این روش با Temporal Difference (TD) learning که از bootstrapping برای این تخمین‌ها استفاده می‌کند متفاوت است.

برای خانه (۱,۱) :
$$\frac{0/9 \times (-5) + (0/9)^2 \times (5) \times 2}{3} = 1/2$$

برای خانه (۲,۲) :
$$\frac{1 \times 5 + 1 \times 5}{2} = +5$$

۴)
$$(1,1) \xrightarrow{iter1} (1,2) \xrightarrow{iter2} (1,3)$$

$$iter1: sample = R + \gamma V^{\pi}((1,2)) = 0 + \frac{9}{10} \times 0 = 0 \Rightarrow$$
$$V^{\pi}((1,1)) = (1-\alpha) V^{\pi}((1,1)) + \alpha \times sample = \frac{9}{10} \times 0 + \frac{1}{10} \times 0 = 0$$

$$iter2: sample = R + \gamma V^{\pi}((1,3)) = (-5) + \frac{9}{10} \times (-5) = -9/5 \Rightarrow$$
$$V^{\pi}((1,2)) = (-\alpha) V^{\pi}((1,2)) + \alpha \times sample = \frac{9}{10} \times 0 + \frac{1}{10} \times (-9/5)$$
$$= -0/95$$

1) DQN

DQN از یک شبکه عصبی برای تقریب تابع Q استفاده می‌کند که پاداش‌های مورد انتظار آینده را برای انجام اقدامات مختلف در حالت‌های مختلف تخمین می‌زند. این امر با مدیریت کارآمد فضاهای ورودی با ابعاد بالا، محدودیت‌های Q-learning سنتی را برطرف می‌کند.

هسته DQN شامل ۲ شبکه عصبی است: شبکه Q اصلی و یک شبکه Q هدف. شبکه اصلی مقادیر Q را بر اساس وضعیت فعلی و اقدامات مورد نظر پیش‌بینی می‌کند، در حالی که شبکه هدف آپدیت هدف پایدار برای شبکه اصلی فراهم می‌کند تا از آن یاد بگیرد.

DQN از یک loss function برای به حداقل رساندن تفاوت بین مقادیر Q پیش‌بینی شده و مقادیر Q هدف استفاده می‌کند که با استفاده از شبکه هدف محاسبه می‌شوند. این فرآیند به طور مکرر پیش‌بینی‌های شبکه‌های شبکه Q را اصلاح می‌کند تا با هدف همگرایی در تابع Q بهینه که Reward های تجمعی را به حداکثر برساند.

DQN کاربردهای زیادی از جمله در بازی‌های ویدئویی دارد.