a. Function Approximation

1. Which Reinforcement Learning methods use tabular representation? Explain how each method employs a tabular representation in brief.

2. What are the drawbacks of tabular implementation?

- Let's consider a tabular representation for Qπ(s, a). Suppose the state-action space is very large, with millions of (s, a) pairs, and at the beginning of training each cell representing a particular Qπ(s, a) is initialized to 0.
- During training, an agent visits (s, a) pairs and the table is updated, but the unvisited (s, a) pairs continue to have Qπ(s, a) = 0. Since the state-action space is large, many (s, a) pairs will remain unvisited and their Q-value estimates will remain at 0 even if (s, a) is desirable, with Qπ(s, a) 0.
- The main issue is that a tabular function representation does not learn anything about how different states and actions relate to each other.

3. What are the solutions to the drawbacks of tabular implementation?

4. What is function approximation? Function approximation attempts to solve the issues of tabular implementation. Justify

We have seen that our estimates of value functions are represented as a table with one entry for each state or for each state-action pair. This is a particularly clear and instructive case, but of course, it is limited to problems with small numbers of states and actions. The problem is not just the memory needed for large tables, but the time and data needed to fill them accurately. In other words, the key issue is that of generalization.

The kind of generalization we require is often called function approximation because it takes examples from a desired function (e.g., a value function) and attempts to generalize from them to construct an approximation of the entire function.
A function is just a mapping from inputs to outputs, and these can take many forms.

Advantages of Function Approximation:

- Deep reinforcement learning replaces tabular methods of estimating state values with function approximation.

- Function approximation not only eliminates the need to store all state and value pairs in a table, but it also enables the agent to generalize the value of states it has never seen before, or has partial information about, by using the values of similar states.
- Much of the exciting advancements in deep reinforcement learning have come about because of the strong ability of neural networks to generalize across enormous state spaces.

5. What are the different function approximators available and which one is widely used in RL?

There are many function approximators:
1. Artificial Neural Networks
2. Decision Tree
3. Nearest Neighbor
4. Fourier/wavelet bases
5. Coarse coding

Artificial Neural Network as Function Approximator:
ANN, comprising many layers, drive deep learning. Deep Neural Networks (DNNs) are such types of networks where each layer can perform complex operations such as representation and abstraction that make sense of images, sound, and text. Considered the fastest-growing field in machine learning, deep learning represents a truly disruptive digital technology, and it is being used by increasingly more companies to create new business models.

6. What are the advantages of function approximation?

- Deep reinforcement learning replaces tabular methods of estimating state values with function approximation.
- Function approximation not only eliminates the need to store all state and value pairs in a table, but it also enables the agent to generalize the value of states it has never seen before, or has partial information about, by using the values of similar states.
- Much of the exciting advancements in deep reinforcement learning have come about because of the strong ability of neural networks to generalize across enormous state spaces.

7. Explain policy gradient with function approximation. What is the consistency/compatibility condition between function and policy parameterization required for convergence?

8. What are the conditions for using policy gradient with function approximation?

b. Deep RL

1. Explain the classification of the deep reinforcement algorithms. Describe them and give an example of each type

2. How is deep learning introduced or used for reinforcement learning tasks?

Deep learning is often used in reinforcement learning tasks to approximate the value or policy functions of the agent. The value function estimates the expected cumulative reward that an agent can obtain from a particular state, while the policy function defines the mapping between states and actions to be taken by the agent.

The most common way to introduce deep learning into reinforcement learning is through the use of deep neural networks to approximate these functions. Specifically, deep neural networks can be used as function approximators to learn the value or policy functions from the observed states and actions of the environment.

3. Compare Deep Learning and Reinforcement Learning.

Deep Learning Vs Reinforcement Learning

DL

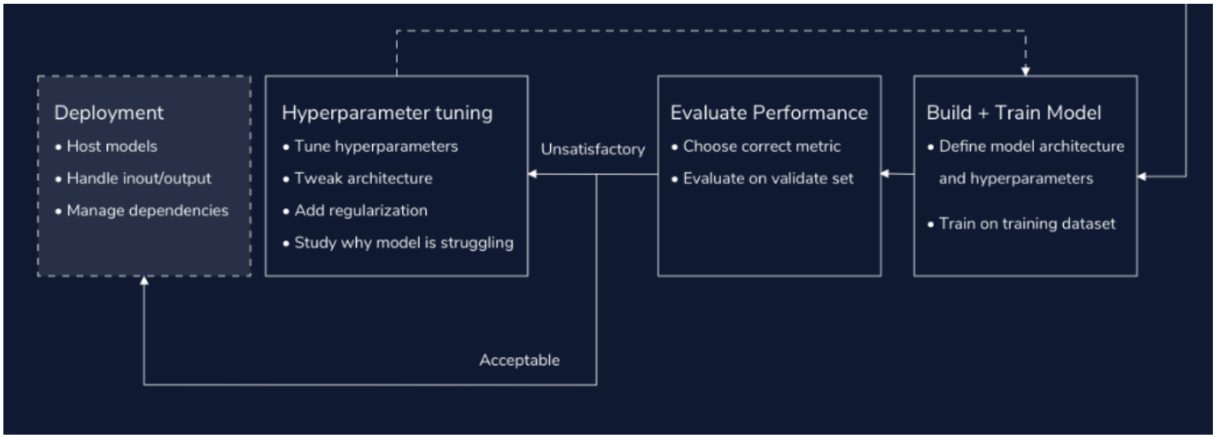
- Deep learning is learning from a training set and then applying that learning to a new data set.
- DL is data driven
- Used in the areas of speech and picture recognition, the dimension reduction task and pretraining for deep networks
- Deep learning is focused mostly on recognition and has a weaker connection to interactive learning

RL

- Reinforcement learning is dynamically learning by adjusting actions based in continuous feedback to maximize a reward.
- It is goal driven
- Used in the fields of robotics, computer gaming, Telecommunications, Healthcare, elevator scheduling
- Reinforcement learning is a type of artificial intelligence that can be enhanced by the use of feedback, making it more comparable to the capabilities of the human brain than deep learning.

4. Explain the training workflow of deep learning.

Deep Learning Training Workflow



1. Build and Train Model
2. Evaluate Performance and Hyperparameter tuning if needed
3. Deployment

DL Training Workflow: Build and Train Model

- Once we have our dataset, it's time to choose our loss function, and our layers.
- For each layer, we also need to select a reasonable number of hidden units. There is no absolute science to choosing the right size for each layer, nor the number of layers – it all depends on your specific data and architecture.
- It's good practice to start with a few layers (2-6).
- Usually, we create each layer with between 32 and 512 hidden units.
- We also tend to decrease the size of hidden layers as we move upwards - through the model.
- We usually try SGD and Adam optimizers first.
- When setting an initial learning rate, a common practice is to default to `0.01`

DL Training Workflow: Performance Evaluation

- Each time we train the model, we evaluate its performance on our validation (Test) set.
- Our performance on the validation set gives us a sense for how our model will perform on new, unseen data.
- When considering performance, it's important to choose the correct metric.
- If our data set is heavily imbalanced, accuracy will be less meaningful. In this case, we likely want to consider metrics like precision and recall.
- F1-score is another useful metric that combines both precision and recall.
- A confusion matrix can help visualize what data-points are misclassified etc.

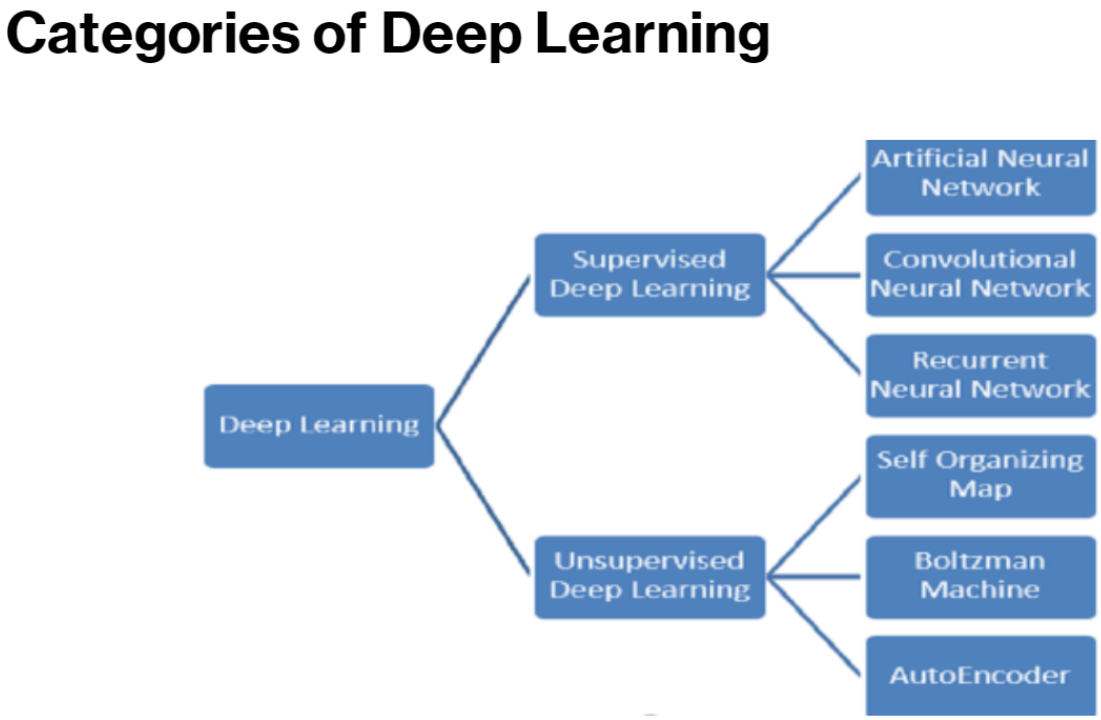
DL Training Workflow: Tuning Parameters

- We will almost always need to iterate upon our initial hyperparameters.
- When training and evaluating our model, we explore different learning rates, batch sizes, architectures, and regularization techniques.
- As we tune our parameters, we should watch our loss and metrics, and be on the lookout for clues as to why our model is struggling.
- Unstable learning means that we likely need to reduce our learning rate and or/increase our batch size. A disparity between performance on the training and evaluation sets means we are overfitting, and should reduce the size of our model
- Poor performance on both the training and the test set means that we are underfitting, and may need a larger model or a different learning rate.
- A common practice is to start with a smaller model and scale up our hyperparameters until we do see training and validation performance diverge, which means we have overfit to our data.

DL Training Workflow: Deployment

- Once you're confident (based on the performance evaluation on the validation set) that you have a model that meets the expectations and requirements set earlier, it is time to bring up the test set and evaluate the model's performance on the test set, which will provide a measure of the model's ability to generalize to previously unseen data.
- At the end of this step, you should have a model whose performance on the test set satisfies an agreed-upon metric of success and is ready for deployment.

5. Name the different categories of Deep Learning.



6. What is Deep Reinforcement Learning?

Deep reinforcement learning (DRL) is a subfield of machine learning that combines reinforcement learning (RL) and deep learning (DL) techniques to enable artificial agents to learn to make decisions in complex and dynamic environments.

In traditional reinforcement learning, an agent learns to interact with an environment by taking actions that maximize a reward signal.

DRL algorithms use deep neural networks as function approximators to learn the value or policy functions from the observed states and actions of the environment. Deep neural networks are

powerful function approximators that can learn to represent complex and high-dimensional input data, such as images and videos.

One of the main advantages of DRL is its ability to learn from high-dimensional and continuous data, which is often encountered in real-world applications such as robotics, gaming, and autonomous driving.

7. What are the different applications of Deep Reinforcement Learning? Explain any one in brief.

Applications of DRL: Robotics, Gaming, NLP

One example of an application of DRL is autonomous driving. In this application, DRL can be used to train an agent to learn to drive a vehicle autonomously. The agent learns to navigate the environment by taking actions that maximize a reward signal, such as staying in the correct lane, obeying traffic signals, and avoiding collisions.

To train an autonomous driving agent using DRL, a simulator is typically used to generate training data. The agent interacts with the simulator by taking actions such as accelerating, braking, and turning the steering wheel. The simulator provides feedback to the agent in the form of a reward signal, which is used to update the agent's policy or value function.

8. Explain the cost function in Deep Q-Network.

$$Cost = \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \max_{a} Q(s',a;\theta)\right)\right]^2$$

DQN cost function

Where θ is the weight vector in the neural network for the current Q value,
r is the reward for the current state and action
This is the MSE function, where the current Q value is the prediction (y), and the immediate and future rewards are the target (y'):

$$MSE = \frac{1}{n}\sum_{1}^{n}(y_i - y_i')^2$$

Mean square error function

This is why Q(s', a;θ) is usually referred to as Q-target.

9. Explain Double Deep Q-Network with the help of a diagram.

Double Deep Q-N

$$Cost = \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \max_{a} Q(s',a;\theta)\right)\right]^2$$

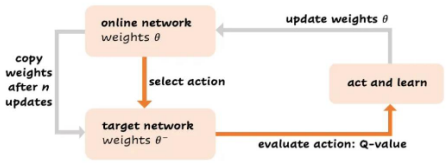
- The target of DQN is given as: $Y_t^{DQN} = r_{t+1} + \gamma \max_{a} Q(S_{t+1}, a;\ \theta_t)$
- The highest Q-value of the next state is selected, and the same network is used for action selection and evaluation. This can lead to an overestimation of action values.
- Instead of using one neural network for action selection and evaluation, it's possible to use two neural networks. One network is called the online network, and the other one the target network. The weights from the online network are slowly copied to the target network, which stabilizes learning. For a Double DQN, the target is defined as:

$$Y_t^{DoubleDQN} = r_{t+1} + \gamma Q(S_{t+1}, \mathbf{argmax}_{a}\, \boldsymbol{Q}(\boldsymbol{S_{t+1}}, \boldsymbol{a};\ \boldsymbol{\theta_t}), \theta_t^-)$$

Double Deep Q-N

$$Y_t^{DoubleDQN} = r_{t+1} + \gamma Q(S_{t+1}, \mathbf{argmax}_{a}\, \boldsymbol{Q}(\boldsymbol{S_{t+1}}, \boldsymbol{a};\ \boldsymbol{\theta_t}), \theta_t^-)$$

- In this case, you select the action with the highest Q-value using the online network (weights θ), and the target network (weights θ-) is used to estimate its value.
- Because the target network slowly updates the Q-values, these values aren't overestimated like when just using the online network.



- A problem in reinforcement learning is **overestimation** of the action values.
- This can cause learning to fail.
- In tabular Q-learning, the Q-values will converge to their true values. The downside of a Q-table is that it does not scale. For more complex problems, we need to *approximate* the Q-values, for example with a DQN.
- This approximation is a problem, because this causes noise on the output production, due to the generalization.
- The noise can cause systematic overestimation of the Q-values. In some situations, this will lead to a suboptimal policy.
- Double DQN solves this by separation of action selection and action evaluation. This leads to more stable learning and an improvement of results.

10. State the DQN algorithm. Explain the working of the algorithm. Describe how the Q-function is learned in the DQN algorithm.

11. State the challenges in Deep Q- learning and explain any one technique to address them.

One of the key challenges in implementing Deep Q-Learning is that the Q- function is typically non-linear and can have many local minima. This can make it difficult for the neural network to converge to the correct Q- function.

To address this, several techniques have been proposed, such as experience replay and target networks.

- **Experience replay** is a technique where the agent stores a subset of its experiences (state, action, reward, next state) in a memory buffer and samples from this buffer to update the Q-function. This helps to decorrelate the data and make the learning process more stable.
- **Target networks**, on the other hand, are used to stabilize the Q-function updates. In this technique, a separate network is used to compute the target Q-values, which are then used to update the Q-function network.

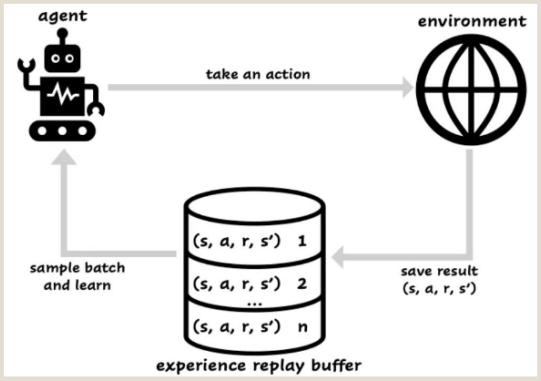
12. Explain the ways in which the DQN algorithm can be improved

Ways to improve DQN

- Prioritized Experience Replay
- Double DQN
- Dueling Network architecture
- Actor Critic
- Noisy Networks
- Distributional RL

Prioritized Experience Replay

- This first technique (Experience Replay) is easy to implement. The idea is simple to implement and most of the deep reinforcement systems makes use of it.
- It works as given in the diagram
- The experience replay buffer is the memory where you store the most recent transitions. Usually, the replay buffer has a fixed size.
- An improved version of experience replay is prioritized experience replay, where you replay important transitions more frequently.
- The importance of a transition is measured by the magnitude of their TD error.



- **Autocorrelation** is a problem in deep reinforcement learning.
- This means that when you train an agent on consecutive samples, the samples are correlated, because there is a relationship between consecutive Q-values. By taking a random sample from the experience replay buffer, this problem is solved.
- Another advantage of this technique is that previous experiences are used efficiently.
- The agent learns multiple times from the same experience and this speeds up the learning process.

Double Deep Q-N

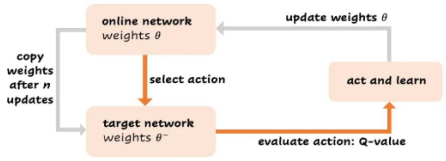
$$Cost = \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \max_a Q(s',a;\theta)\right)\right]^2$$

- The target of DQN is given as: $Y_t^{DQN} = r_{t+1} + \gamma \max_a Q(S_{t+1}, a;\ \theta_t)$
- The highest Q-value of the next state is selected, and the same network is used for action selection and evaluation. This can lead to an overestimation of action values.
- Instead of using one neural network for action selection and evaluation, it's possible to use two neural networks. One network is called the online network, and the other one the target network. The weights from the online network are slowly copied to the target network, which stabilizes learning. For a Double DQN, the target is defined as:

$$Y_t^{DoubleDQN} = r_{t+1} + \gamma Q(S_{t+1}, \mathbf{argmax}_a \boldsymbol{Q}(\boldsymbol{S_{t+1}}, \boldsymbol{a};\ \boldsymbol{\theta_t}), \theta_t^-)$$

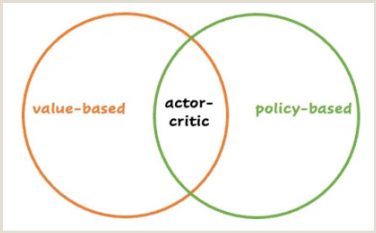
$$Y_t^{DoubleDQN} = r_{t+1} + \gamma Q(S_{t+1}, \mathbf{argmax}_a \boldsymbol{Q}(\boldsymbol{S_{t+1}}, \boldsymbol{a};\ \boldsymbol{\theta_t}), \theta_t^-)$$

- In this case, you select the action with the highest Q-value using the online network (weights θ), and the target network (weights θ-) is used to estimate its value.
- Because the target network slowly updates the Q-values, these values aren't overestimated like when just using the online network.

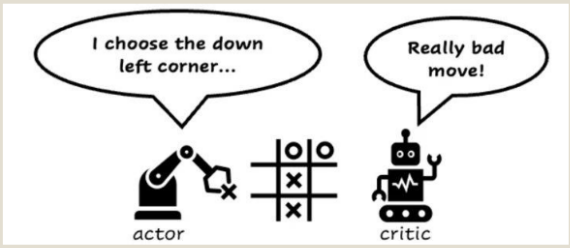


Actor Critic

- So far, the methods discussed calculate the values of state-action pairs and use these action values directly to determine the best policy.
- The best policy is the action with the highest action value for every state.
- Another way to approach RL problems is by representing the policy directly.
- In this case, a neural network outputs a probability for each action.
- The different methods are called value-based methods and policy-based methods, respectively.
- Actor-critic uses a combination of value and policy-based methods. The policy is represented independent of the value function.



Actor Critic



- In actor-critic, there are two neural networks:

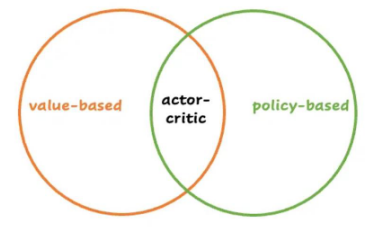
1. A policy network, the *actor*. This network selects the actions.
2. A deep Q network for the action values, the *critic*. The deep Q network is trained normally and learns from the agent's experiences.

- The policy network relies on the action values estimated by the deep Q network. It changes the policy based on these action values. Learning happens on-policy.

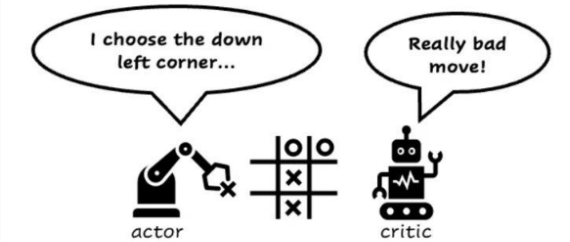
13. Explain the Actor critic algorithm.

Actor Critic

- So far, the methods discussed calculate the values of state-action pairs and use these action values directly to determine the best policy.
- The best policy is the action with the highest action value for every state.
- Another way to approach RL problems is by representing the policy directly.
- In this case, a neural network outputs a probability for each action.
- The different methods are called value-based methods and policy-based methods, respectively.
- Actor-critic uses a combination of value and policy-based methods. The policy is represented independent of the value function.



Actor Critic



- In actor-critic, there are two neural networks:

1. A policy network, the *actor*. This network selects the actions.
2. A deep Q network for the action values, the *critic*. The deep Q network is trained normally and learns from the agent's experiences.

- The policy network relies on the action values estimated by the deep Q network. It changes the policy based on these action values. Learning happens on-policy.

Pseudocode for Actor Critic Algorithm

1.Sample {s_t, a_t}using the policy πθ from the actor-network.

2. Evaluate the advantage function A_t. It can be called as TD error δt. In Actor-critic algorithm, advantage function is produced by the critic-network.

$$A_{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$$

3. Evaluate the gradient using the below expression: $\nabla J(\theta) \approx \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t, s_t) A_{\pi_\theta}(s_t, a_t)$

4. Update the policy parameters, θ $\theta = \theta + \alpha \nabla J(\theta)$

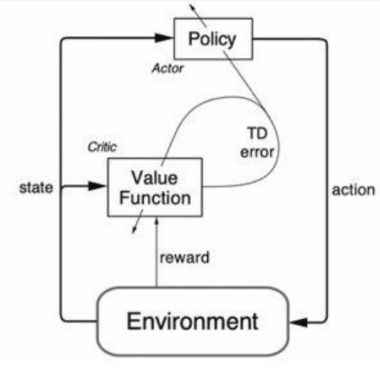
5.Update the weights of the critic-based value-based RL(Q-learning). δt is equivalent to advantage function.

$$w = w + \alpha \delta_t$$

6. Repeat 1 to 5 until we find the optimal policy πθ.

14. Explain Actor-Critic architecture with a well-labeled diagram.

Actor Critic Architecture



TD Error: $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

- Actor-critic methods are TD methods that have a separate memory structure to explicitly represent the policy independent of the value function.
- The policy structure is known as the *actor*, because it is used to select actions, and the estimated value function is known as the *critic*, because it criticizes the actions made by the actor.
- Learning is always on-policy: the critic must learn about and critique whatever policy is currently being followed by the actor.
- The critique takes the form of a TD error. This scalar signal is the sole output of the critic and drives all learning in both actor and critic

15. Write a short note on Alpha Go

16. Explain Deep Deterministic Policy Gradient (DDPG)

DDPG (Deep Deterministic Policy Gradient)

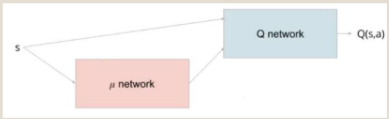
- Deep Deterministic Policy Gradient (DDPG) is a reinforcement learning technique that combines both Q-learning and Policy gradients. DDPG being an actor-critic technique consists of two models: Actor and Critic.
- The actor is a policy network that takes the state as input and outputs the exact action (continuous), instead of a probability distribution over actions. The critic is a Q-value network that takes in state and action as input and outputs the Q-value.
- DDPG is an "off"-policy method. DDPG is used in the continuous action setting and the "deterministic" in DDPG refers to the fact that the actor computes the action directly instead of a probability distribution over actions.
- DDPG is used in a continuous action setting and is an improvement over the vanilla actor-critic.

DDPG (Deep Deterministic Policy Gradient)

- In DQN the optimal action is taken by taking argmax over the Q-values of all actions. In DDPG the actor is a policy network that does exactly this. It outputs the action directly (action can be continuous), bypassing the argmax.

$$Define: \ \mu(s) = \arg\max_a Q(s,a)$$

- The policy is deterministic since it directly outputs the action. In order to promote exploration some Gaussian noise is added to the action determined by the policy.
- To calculate the Q-value of a state, the actor output is fed into the Q-network to calculate the Q-value. This is done only during the calculation of TD-error which we will describe later.



17. Compare Deep RL with Supervised Learning

18. Justify AlpaGo is implemented using reinforcement learning. Write all the elements of RL with respect to AlphaGO